UNIVERSITAS BHAYANGKARA JAKARTA RAYA
FAKULTAS ILMU KOMPUTER
Kampus I: Jl. Harsono RM No. 67, Ragunan, Pasar Minggu, Jakarta Selatan 12550
Telepon: (021) 27808121 – 27808882
Kampus II: Jl. Raya Perjuangan, Marga Mulya, Bekasi Utara, Jawa Barat, 17142
Telepon: (021) 88955882, Fax.: (021) 88955871
Web: fasilkom.ubharajaya.ac.id, E-mail: fasilkom@ubharajaya.ac.id

SURAT TUGAS

Nomor: ST/1242/X/2023/FASILKOM-UBJ

1. Dasar: Kalender Akademik Universitas Bhayangkara Jakarta Raya Tahun Akademik 2023/2024.'

2. Dalam rangka mewujudkan Tri Dharma Perguruan Tinggi untuk Dosen di Universitas Bhayangkara Jakarta Raya maka dihimbau untuk melakukan Penelitian.

3. Sehubungan dengan hal tersebut di atas, maka Dekan Fakultas Ilmu Komputer Universitas Bhayangkara Jakarta Raya menugaskan:

NO.	NAMA	NIDN	JABATAN
1.	Khairunnisa Fadhilla Ramdhania, S.Si., M.Si.	0328039201	Dosen Tetap Prodi Informatika

Membuat Buku dengan judul **"Matematika Diskrit"**, yang diterbitkan oleh Get Press Indonesia, Cetakan Pertama: November 2023, ISBN: 978-623-198-818-8.

4. Demikian penugasan ini agar dapat dilaksanakan dengan penuh rasa tanggung jawab.

Jakarta, 30 Oktober 2023
DEKAN FAKULTAS ILMU KOMPUTER

Dr. Dra. Tyastuti Sri Lestari, M.M.
NIP. 1408206

MATEMATIKA DISKRIT

Lulut Alfaris, Khairunnisa Fadhilla Ramdhania, Mindo, Ida Fitriana Ambarsari, Suwito Pomalingo, Indah Resti Ayuni Suri, Wirawan Istiono

MATEMATIKA DISKRIT

Penulis :
Lulut Alfaris
Khairunnisa Fadhilla Ramdhania
Mindo
Ida Fitriana Ambarsari
Suwito Pomalingo
Indah Resti Ayuni Suri
Wirawan Istiono

GET PRESS INDONESIA

MATEMATIKA DISKRIT

Penulis :

Lulut Alfaris
Khairunnisa Fadhilla Ramdhania
Mindo
Ida Fitriana Ambarsari
Suwito Pomalingo
Indah Resti Ayuni Suri
Wirawan Istiono

ISBN : 978-623-198-818-8

Editor : Ari Yanto., M.Pd
Penyunting : Dhinie Anjelicha, S.Tr.Kes
Desain Sampul dan Tata Letak : Tri Putri Wahyuni, S.Pd

Penerbit : Get Press Indonesia
Anggota IKAPI No. 033/SBA/2022

Redaksi :
Jl. Palarik Air Pacah RT 001 RW 006
Kelurahan Air Pacah Kecamatan Koto Tangah
Padang Sumatera Barat
Website : www.getpress.co.id
Email : globaleksekutifteknologi@gmail.com

Cetakan pertama, November 2023

KATA PENGANTAR

Segala puji dan syukur atas kehadirat Allah SWT dalam segala kesempatan. Sholawat beriring salam dan doa kita sampaikan kepada Nabi Muhammad SAW. Alhamdulillah atas Rahmat dan Karunia-Nya penulis telah menyelesaikan buku Matematika Diskrit.

Buku ini membahas tentang teori himpunan, logika matematika, relasi dan fungsi, kompleksitas algoritma, penerapan matematika diskrit dalam struktur data, penerapan matematika diskrit dalam bahasa formal dan penerapan matematika diskrit dalam kriptografi.

Proses penulisan buku ini berhasil diselesaikan atas kerjasama tim penulis. Demi kualitas yang lebih baik dan kepuasan para pembaca, saran dan masukan yang membangun dari pembaca sangat kami harapkan.

Penulis ucapkan terima kasih kepada semua pihak yang telah mendukung dalam penyelesaian buku ini. Terutama pihak yang telah membantu terbitnya buku ini dan telah mempercayakan, mendorong, dan menginisiasi terbitnya buku ini. Semoga buku ini dapat bermanfaat bagi masyarakat Indonesia.

Penulis, November 2023

DAFTAR ISI

BAB 1
TEORI HIMPUNAN

Oleh Lulut Alfaris

1.1 Sejarah Teori Himpunan

Perjalanan sejarah teori himpunan merupakan eksplorasi yang menarik dalam pengembangan gagasan matematika dan konsep yang terkait dengan himpunan. Dalam hal ini akan dijelaskan teori himpunan dalam sejarah modern. Pada akhir abad ke-19, matematikawan Jerman Georg Cantor memainkan peran utama dalam membentuk teori himpunan modern. Kontribusinya meliputi pengenalan gagasan himpunan, penerapan himpunan tak terhingga, serta eksplorasi tentang ukuran tak terhingga yang beragam. Cantor diakui sebagai salah satu pendiri teori himpunan dan fondasinya menjadi dasar bagi pengembangan disiplin matematika formal ini (Agustianti et al., 2022).

Sementara itu, tokoh lain yang berpengaruh dalam sejarah awal teori himpunan adalah Richard Dedekind. Ia memperkenalkan konsep pemotongan Dedekind, yang menjadi landasan untuk mendefinisikan bilangan real, serta memperkenalkan gagasan himpunan dalam merinci aspek-aspek dasar dalam matematika (Alfaris, Sarumaha, et al., 2022).

Memasuki awal abad ke-20, Ernst Zermelo dan Abraham Fraenkel merumuskan seperangkat aksioma yang dikenal sebagai teori himpunan Zermelo-Fraenkel (ZF). Aksioma-aksioma ini memberikan kerangka kerja formal yang diperlukan untuk pengembangan teori himpunan, dan secara signifikan mengatasi beberapa masalah mendasar yang pernah muncul dalam usaha sebelumnya dalam mengatur matematika.

Selanjutnya, pengembangan teori himpunan aksiomatik terus berlanjut melalui kontribusi dari matematikawan seperti John von Neumann dan Paul Bernays. Upaya ini menghasilkan penyempurnaan lebih lanjut pada teori himpunan ZF serta pengenalan aksioma tambahan, seperti Aksioma Pilihan.

Sejarah awal teori himpunan juga diwarnai oleh paradoks-paradoks, termasuk Paradoks Russell, yang menyoroti ketidakkonsistenan dalam teori himpunan naif. Matematikawan seperti Bertrand Russell dan Alfred North Whitehead melakukan upaya signifikan dalam menangani paradoks-paradoks ini. Akhirnya, hal ini memunculkan perkembangan teori himpunan aksiomatik yang berhasil menghindari inkonsistensi tersebut.

Saat ini, teori himpunan modern sebagian besar didasarkan pada teori himpunan Zermelo-Fraenkel dengan Aksioma Pilihan (ZF-C). Teori ini menjadi dasar utama bagi sebagian besar cabang matematika dan memiliki aplikasi penting dalam berbagai bidang seperti logika, topologi, dan aljabar.

Sejarah teori himpunan mencerminkan semangat berkelanjutan dalam memahami konsep-konsep dasar dalam matematika dan menjaga konsistensinya secara logis. Seiring berjalannya waktu, teori ini memainkan peran kunci dalam membentuk cara matematika dirumuskan dan diterapkan dalam era modern."

1.2 Dasar Teori Himpunan

Teori himpunan adalah cabang penting dari matematika yang berkaitan dengan studi tentang himpunan. Himpunan itu sendiri adalah kumpulan objek atau elemen-elemen yang tergabung dalam satu kesatuan. Seiring perkembangan waktu, teori himpunan telah menjadi dasar bagi berbagai konsep matematika dan

aplikasinya di berbagai bidang. Sejarah perkembangan teori ini mencakup sumbangan dari banyak matematikawan terkemuka, seperti Georg Cantor, yang membentuk landasan dasar teori himpunan (Stillwell, 2013).

Konsep-konsep kunci dalam teori himpunan meliputi: Notasi Himpunan: Himpunan-himpunan biasanya dilambangkan dengan kurung kerawang, seperti {1, 2, 3}, untuk mewakili himpunan dengan elemen-elemen 1, 2, dan 3. Gabungan dan Irisan: Gabungan dua himpunan menggabungkan semua elemen-elemen dari kedua himpunan tersebut, sedangkan irisan mencari elemen-elemen yang sama di antara keduanya. Himpunan Bagian dan Himpunan Induk (Superset): Sebuah himpunan A disebut sebagai himpunan bagian dari himpunan B jika semua elemen dari A juga terdapat di B. Sebaliknya, himpunan B adalah himpunan induk dari A. Komplement: Komplement dari himpunan A terhadap himpunan universal U berisi semua elemen dari U yang tidak terdapat di dalam A. Kardinalitas: Kardinalitas sebuah himpunan adalah jumlah elemen yang terdapat di dalamnya. Untuk himpunan-himpunan berhingga, ini adalah bilangan bulat biasa, tetapi untuk himpunan-himpunan tak hingga, hal ini bisa menjadi lebih kompleks (Dasgupta, 2014).

1. Himpunan dan Elemen

Himpunan adalah konsep dasar dalam matematika yang mewakili kumpulan objek atau elemen-elemen yang memiliki karakteristik atau sifat yang sama. Himpunan dapat berisi elemen-elemen beragam, seperti angka, huruf, atau objek lainnya. Notasi yang umum digunakan untuk menggambarkan himpunan adalah dengan menggunakan kurung kerawang { }, seperti {1, 2, 3} yang merupakan himpunan bilangan bulat positif 1, 2, dan 3.

Contoh-contoh Himpunan

Himpunan dapat ditemukan di berbagai aspek kehidupan sehari-hari dan matematika. Beberapa contoh umum termasuk:

Himpunan bilangan bulat: {1, 2, 3, 4, ...}

Himpunan huruf dalam alfabet: {a, b, c, ..., z}

Himpunan buah-buahan: {apel, pisang, jeruk}

Himpunan warna dasar: {merah, hijau, biru}

Setiap himpunan dapat memiliki elemen-elemen yang berbeda-beda tergantung pada konteksnya. Misalnya, himpunan buah-buahan berisi elemen-elemen yang merupakan nama buah, sedangkan himpunan warna dasar berisi elemen-elemen yang adalah warna-warna dasar.

2. Elemen dalam Himpunan

Elemen dalam sebuah himpunan adalah objek atau anggota yang termasuk dalam himpunan tersebut. Dalam himpunan {1, 2, 3}, elemen-elemen 1, 2, dan 3 adalah anggota-anggota himpunan tersebut. Dalam notasi matematika, kita menggunakan simbol $\in$ untuk menyatakan bahwa suatu elemen adalah bagian dari himpunan. Misalnya, $1 \in \{1, 2, 3\}$ berarti angka 1 adalah anggota dari himpunan {1, 2, 3}.

3. Himpunan Kosong dan Himpunan Universal

Himpunan kosong, dilambangkan dengan simbol $\emptyset$ atau {}, adalah himpunan yang tidak memiliki elemen sama sekali. Dalam konteks teori himpunan, himpunan kosong merupakan konsep yang penting. Misalnya, jika kita memiliki himpunan A = {bilangan ganjil positif kurang dari 1}, maka himpunan A adalah himpunan kosong, karena tidak ada bilangan yang memenuhi kriteria tersebut.

4. Himpunan Universal dalam Konteks Teori Himpunan

Himpunan universal, dalam teori himpunan, adalah himpunan yang berisi semua elemen yang mungkin dibahas dalam suatu konteks tertentu. Himpunan universal ini biasanya dilambangkan dengan simbol U. Misalnya, jika kita sedang berbicara tentang himpunan bilangan bulat, maka himpunan universalnya adalah himpunan semua bilangan bulat. Himpunan universal ini bergantung pada konteks pembahasan, dan sering digunakan untuk memberikan batasan atau kerangka kerja dalam analisis teori himpunan (Octavianti et al., 2022).

Dalam rangka mengeksplorasi dan memahami teori himpunan lebih lanjut, pemahaman yang kuat tentang konsep dasar seperti himpunan, elemen, himpunan kosong, dan himpunan universal adalah langkah awal yang penting. Konsep-konsep ini membentuk dasar untuk pemahaman lebih lanjut tentang operasi-operasi dan konsep-konsep lanjutan dalam teori himpunan (Alfaris, Dewadi, et al., 2022).

1.3. Operasi Himpunan

Operasi himpunan adalah aspek penting dalam teori himpunan yang memungkinkan kita untuk menggambarkan hubungan, kesamaan, dan perbedaan antara himpunan-himpunan. Dalam konteks teori himpunan, beberapa operasi himpunan yang paling fundamental dan umum digunakan meliputi penggabungan (union), irisan (intersection), komplement (complement), dan beberapa operasi lainnya. Dalam tulisan ini, kita akan menjelaskan masing-masing operasi ini dengan detail.

1. Penggabungan (Union)

Penggabungan dua himpunan A dan B, dilambangkan dengan A ∪ B, adalah operasi yang menghasilkan himpunan yang berisi semua elemen yang ada di A, di B, atau di keduanya. Dalam kata lain, penggabungan menggabungkan semua elemen dari kedua himpunan. Misalnya, jika kita memiliki himpunan A = {1, 2, 3} dan himpunan B = {3, 4, 5}, maka penggabungan A ∪ B akan menghasilkan himpunan {1, 2, 3, 4, 5}. Operasi ini berguna dalam menggabungkan data atau elemen-elemen yang terkait dari dua himpunan yang berbeda.

2. Irisan (Intersection)

Irisan dua himpunan A dan B, dilambangkan dengan A ∩ B, adalah operasi yang menghasilkan himpunan yang berisi semua elemen yang ada di A dan di B secara bersamaan. Dengan kata lain, irisan mencari elemen-elemen yang sama di antara kedua himpunan.Sebagai contoh, jika kita memiliki himpunan A = {1, 2, 3} dan himpunan B = {3, 4, 5}, maka irisan A ∩ B akan menghasilkan himpunan {3}, karena itulah satu-satunya elemen yang ada di kedua himpunan tersebut. Irisan digunakan untuk menemukan elemen-elemen yang bersamaan atau kesamaan dalam dua himpunan yang berbeda.

3. Komplement (Complement)

Komplement sebuah himpunan A terhadap himpunan universal U, dilambangkan dengan A', adalah operasi yang menghasilkan himpunan yang berisi semua elemen yang ada di U tetapi tidak ada di A. Dalam kata lain, komplement mencari elemen-elemen yang tidak termasuk dalam himpunan A. Misalnya, jika kita memiliki himpunan universal U = {1, 2, 3, 4, 5} dan himpunan A = {3, 4}, maka komplement A' akan menghasilkan himpunan {1, 2, 5}, karena itulah elemen-elemen yang ada di U tetapi tidak ada di A.

Operasi komplement digunakan untuk menemukan elemen-elemen yang tidak termasuk dalam suatu himpunan.

4. Operasi Lain

Selain penggabungan, irisan, dan komplement, ada beberapa operasi himpunan lainnya yang digunakan dalam teori himpunan, seperti diferensiasi (difference), gabungan disjungsi (disjoint union), dan simetris diferensiasi (symmetric difference). Operasi-operasi ini memungkinkan kita untuk melakukan manipulasi dan analisis yang lebih kompleks terhadap himpunan-himpunan. Sebagai contoh, diferensiasi antara dua himpunan A dan B, dilambangkan dengan A - B, adalah operasi yang menghasilkan himpunan yang berisi elemen-elemen yang ada di A tetapi tidak ada di B. Operasi ini berguna dalam menemukan elemen-elemen yang ada dalam satu himpunan tetapi tidak dalam himpunan lainnya.

1.4 Diagram Venn

Sebuah diagram Venn adalah cara untuk menggambarkan hubungan antara himpunan. Setiap himpunan ditampilkan sebagai sebuah lingkaran dan lingkaran-lingkaran akan bertumpang tindih jika himpunan-himpunannya beririsan. (Alfaris, Gustian, et al., 2022).Diagram Venn adalah representasi grafis yang digunakan untuk mengilustrasikan semua potensi relasi atau kesamaan antar sekumpulan himpunan atau objek. Dibuat pertama kali oleh John Venn pada 1880, diagram ini digunakan dalam berbagai disiplin seperti matematika, logika, statistik, probabilitas, linguistik, dan ilmu komputer.

Berikut adalah diagram Venn untuk irisan dan gabungan dari dua himpunan. Bagian yang diarsir pada diagram adalah irisan dan gabungan masing-masing.

Membuat Diagram Venn melibatkan beberapa langkah sederhana:

1. Identifikasi Himpunan: Pertama, identifikasi himpunan-himpunan yang akan Anda gambarkan dan tentukan elemen-elemen yang termasuk dalam masing-masing himpunan. Misalnya, jika Anda ingin menggambarkan hubungan antara himpunan A (buah) dan himpunan B (buah-buahan berwarna merah), identifikasi buah-buahan yang termasuk dalam masing-masing himpunan, seperti apel, ceri, dan stroberi.
2. Gambar Lingkaran: Gambar lingkaran pertama untuk mewakili himpunan A dan lingkaran kedua untuk mewakili himpunan B. Anda dapat menggunakan kertas dan pena atau perangkat lunak grafis untuk membuat lingkaran-lin karan ini.
3. Isi Lingkaran dengan Elemen: Letakkan elemen-elemen yang termasuk dalam himpunan A di dalam lingkaran A dan elemen-elemen yang termasuk dalam himpunan B di dalam lingkaran B. Jika ada elemen yang termasuk dalam kedua himpunan, letakkan elemen tersebut di area tumpang tindih antara kedua lingkaran.
4. Label Lingkaran: Beri label masing-masing lingkaran dengan nama himpunan yang sesuai, dalam hal ini, "A" dan "B."
5. Tambahkan Keterangan Tambahan: Jika diperlukan, tambahkan keterangan tambahan untuk menjelaskan hubungan antara himpunan-himpunan tersebut. Misalnya, Anda dapat menambahkan keterangan bahwa area tumpang tindih menunjukkan buah-buahan merah yang termasuk dalam kedua himpunan.

Ada berbagai jenis fungsi, sehingga bagian ini berisi banyak istilah. Ingatlah bahwa dua himpunan hingga berukuran sama jika mereka mengandung jumlah elemen yang sama. Ide ini dapat diformalisasi menggunakan fungsi, dan setelah konsep ini didefinisikan secara formal, ia dapat diterapkan pada himpunan yang tak terbatas.Eksperimen memungkinkan para peneliti untuk menentukan sebab-akibat antara variabel. Dalam bidang seperti psikologi, kedokteran, dan ilmu sosial, eksperimen memberikan dasar yang kuat untuk rekomendasi kebijakan, pengembangan produk, dan intervensi klinis. Metode eksperimental, dengan kontrol ketatnya atas variabel, meminimalkan kemungkinan kesalahan interpretasi (Alfaris, 2023).

Pasangan terurut adalah kumpulan dua elemen dengan sifat tambahan yaitu satu elemen datang pertama dan satu elemen datang kedua. Himpunan yang hanya berisi x dan y (untuk x tidak sama dengan y) ditulis sebagai {x, y}. Pasangan terurut yang mengandung x dan y dengan x di posisi pertama ditulis sebagai (x, y). Perhatikan bahwa {x, x} bukanlah himpunan yang terdefinisi dengan baik, tetapi (x, x) adalah pasangan terurut yang terdefinisi dengan baik karena dua salinan x berbeda berdasarkan posisi pertama dan kedua.

1.5 Penerapan Teori Himpunan

Teori himpunan, meskipun terdengar sebagai konsep matematis yang abstrak, memiliki banyak penerapan praktis dalam berbagai aspek kehidupan sehari-hari. Konsep himpunan memberikan kerangka kerja yang berguna untuk mengorganisir, mengelompokkan, dan menganalisis data dan informasi. Artikel ini akan mengulas beberapa contoh penerapan penting teori himpunan dalam konteks kehidupan sehari-hari (Subakti et al., 2022).

1. Manajemen Data

Salah satu penerapan paling mencolok dari konsep himpunan adalah dalam manajemen data. Himpunan digunakan untuk mengelompokkan dan mengorganisir data menjadi kelompok-kelompok yang lebih teratur dan terstruktur. Sebagai contoh, dalam dunia bisnis, konsep himpunan digunakan dalam basis data untuk mengelompokkan produk berdasarkan jenisnya, seperti "pakaian," "elektronik," atau "makanan." Dengan demikian, manajemen data menjadi lebih efisien dan memudahkan pengambilan informasi.

2. Klasifikasi dan Kategorisasi

Teori himpunan juga penting dalam proses klasifikasi dan kategorisasi. Di perpustakaan, konsep himpunan digunakan untuk mengorganisir buku berdasarkan subjek atau genre. Setiap himpunan mewakili kategori tertentu seperti "sastra klasik," "ilmu pengetahuan," atau "sejarah." Dengan demikian, pengguna perpustakaan dapat dengan mudah menemukan buku berdasarkan minat atau kebutuhan mereka.

3. Probabilitas dan Statistik

Dalam statistik, teori himpunan digunakan untuk memodelkan dan menganalisis data. Konsep himpunan membantu dalam penggambaran himpunan sampel dalam studi statistik. Ini memungkinkan penghitungan probabilitas berbagai peristiwa dalam eksperimen statistik. Selain itu, teori himpunan juga relevan dalam statistik inferensial untuk memahami konsep seperti kesalahan tipe I dan tipe II dalam pengujian hipotesis.

4. Penyelesaian Masalah

Teori himpunan memainkan peran penting dalam penyelesaian masalah sehari-hari. Dalam perencanaan perjalanan, contohnya, konsep himpunan digunakan untuk mengelompokkan destinasi wisata berdasarkan lokasinya, memungkinkan

perencanaan perjalanan yang lebih efisien. Di manajemen proyek, himpunan membantu mengorganisir tugas-tugas proyek menjadi kelompok-kelompok yang lebih mudah dikelola.

5. Keamanan Komputer

Teori himpunan juga diterapkan dalam keamanan komputer. Konsep himpunan digunakan untuk mengelola izin akses dan mengatur hak akses ke data dan sumber daya komputer. Ini memungkinkan pembuatan kelompok pengguna dengan hak akses yang serupa. Dengan demikian, administrator sistem dapat mengatur akses sistem komputer dengan lebih efektif dan menghindari akses yang tidak sah (Dewadi et al., 2023).

6. Perancangan Basis Data

Dalam perancangan basis data, teori himpunan digunakan untuk menggambarkan hubungan antara entitas dan atribut dalam basis data. Himpunan digunakan untuk menggambarkan entitas dan koneksi antara entitas tersebut. Misalnya, dalam perancangan basis data untuk sebuah perusahaan, himpunan digunakan untuk menggambarkan hubungan antara pelanggan, pesanan, produk, dan faktur.

7. Algoritma Pencarian

Teori himpunan memiliki peran penting dalam algoritma pencarian. Dalam pencarian data, algoritma seperti pencarian biner menggunakan konsep himpunan untuk membagi data menjadi dua bagian. Hal ini mempercepat proses pencarian karena setengah data dapat diabaikan pada setiap langkah pencarian.

8. Pengambilan Keputusan

Dalam pengambilan keputusan, konsep himpunan digunakan untuk menggambarkan berbagai pilihan yang tersedia. Dalam analisis keputusan, kita dapat menggunakan himpunan untuk

menggambarkan himpunan opsi yang tersedia dan himpunan kriteria yang digunakan untuk mengevaluasi opsi tersebut. Ini membantu dalam membuat keputusan yang lebih terinformasi dan tepat.

DAFTAR PUSTAKA

Agustianti, R., Nuryami, Fajriah, N. A., Nasruddin, Nay, F. A., Mahmud, R., Kumanireng, L. B., Y, W. N., Faelasofi, R., Prasetyo, A., Alfaris, L., Anim, Asmin, L. O., Nanang, & Sari, M. E. (2022). *Filsafat Pendidikan Matematika*. PT. Global Eksekutif Teknologi.

Alfaris, L. (2023). DERET BERHINGGA DAN TAK HINGGA. In A. Yanto (Ed.), *Aljabar Elementer* (pp. 167–173). PT. Global Eksekutif Teknologi.

Alfaris, L., Dewadi, F. M., Munim, A., Taba, H. T., Khasanah, Maing, C. M. M., Susano, A., & Rukhmana, T. (2022). *Matriks & Ruang Vektor*. Cendikia Mulia Mandiri.

Alfaris, L., Gustian, D., Setyorini, R., Romli, I., Putri, A. Y. P., Herjuna, S. A. S., Syamsiyah, N., Yuniansyah, Aziza, N., Muhammad, A. C., Umar, N., & Wali, M. (2022). *Riset Operasi*. Indie Press.

Alfaris, L., Sarumaha, Y. A., Sitopu, J. W., Agustianti, R., Rizqi, V., Yenni, Jamaludin, Nurvicalesti, N., Murtako, A., Akbar, N., & Abidin, Z. (2022). *Logika dan Struktur Diskrit*. PT. Global Eksekutif Teknologi.

Dasgupta, A. (2014). *Set Theory With an Introduction to Real Point Sets*. Springer.

Dewadi, F. M., Milasari, L. A., A., H., Wibowo, C., Abdi, Alfaris, L., Saputra, A. A., & Gobel, F. F. (2023). *Desain Penelitian Bidang Teknik*. Get Press Indonesia.

Octavianti, C. T., Mulyawati, I., Setiawan, J., Sitopu, J. W., Alfaris, L., Ratuanik, M., Hasanah, N., Agustianti, R., Pangestika, R. R., & Wulandari, Y. O. (2022). *Konsep Dasar Matematika*. PT. Global Eksekutif Teknologi.

Stillwell, J. (2013). *The Real Numbers, An Introduction to Set Theory and Analysis*. Springer.

Subakti, H., Romli, I., Syamsiyah, N., Herianto, A. A. B. H., Alfaris, L., Hasin, M. K., Hadi, A., Farida, F., Rismayani, R., Setiawan, A., &

Khadafi, R. K. H. S. (2022). *Artificial Intelligence.* Media Sains Indonesia.

BAB 2
LOGIKA MATEMATIKA

Oleh Khairunnisa Fadhilla Ramdhania

Logika adalah ilmu yang membantu kita dalam berpikir dan bernalar. Lebih jauh, untuk mencapai suatu kesimpulan dari beberapa pernyataan. Logika adalah dasar untuk mempelajari algoritma dan pemrograman.

Definisi 2.1 *Proposisi adalah pernyataan yang bernilai benar (true) atau salah (false), tetapi tidak keduanya. Proposisi dilambangkan dengan huruf kecil* (Munir, 2016).

Contoh 2.1

1. "Paus adalah mamalia" merupakan pernyataan dan sebuah proposisi yang memiliki nilai kebenaran "BENAR".
2. "$-3 > -2$" juga merupakan pernyataan dan proposisi, tetapi memiliki nilai kebenaran "SALAH"
3. Bagaimana dengan:

   $$"2x + 1 \geq 5"$$

   Apakah sebuah pernyataan? Ya.
   Apakah sebuah proposisi? Bukan, karena nilai kebenarannya bergantung pada x. Pernyataan tersebut adalah fungsi proposisi atau kalimat terbuka
4. Setelah Kamis adalah Jumat. (Proposisi)
5. Untuk setiap x dan y bilangan real berlaku $x + y = y + x$ (Proposisi)
6. Berapakah 3+2? Bukan proposisi, melainkan berupa kalimat tanya.
7. $y - 7 = 12$ (Bukan proposisi, karena bisa bernilai benar sekaligus salah

Proposisi dapat disajikan dalam bentuk simbolik untuk mempermudah. Simbol untuk proposisi biasanya ditulis dengan menggunakan huruf kecil seperti p, q, r, dll.

Definisi 2.2 *Operator logika merupakan simbol-simbol yang memiliki makna tertentu yang dapat digunakan untuk mengkombinasikan proposisi, yakni sebagai berikut* (Rachmat, 2004)*:*

$\wedge$ *: 'konjungsi', 'dan', 'and'*

$\vee$ *: 'disjungsi', 'atau', 'or'*

$\sim$ *: 'negasi', 'tidak', 'not'*

$\oplus$ *: 'disjungsi eksklusif', 'or exclusive'*

$\rightarrow$ *: 'implikasi', 'jika..., maka...', 'if..., then...'*

$\leftrightarrow$ *: 'biimplikasi', 'jika dan hanya jika', 'if only if (iff)'*

Definisi 2.3 *Kombinasi beberapa proposisi yang dihubungkan oleh operator logika disebut sebagai formula atau dapat dikatakan proposisi majemuk* (Rachmat, 2004)*.*

2.1 Kalkulus Proposisi

Saat menentukan nilai kebenaran dari proposisi yang sudah disajikan dalam bentuk formula, proses kalkulasi dilibatkan. Akibatnya, proses penentuan nilai kebenaran dari suatu formula kita sebut sebagai kalkulus proposisi. Dalam buku ini, nilai kebenaran direpresentasikan dengan simbol 'T' untuk benar dan 'F' untuk salah.

Misalkan p, q adalah proposisi tunggal, ada 4 macam proposisi:

1. Konjungsi: p dan q, dinotasikan $p \wedge q$
2. Disjungsi: p atau q atau keduanya, dinotasikan $p \vee q$
3. Negasi: tidak p, dinotasikan $\sim p$
4. Disjungsi eksklusif: p atau q tapi bukan keduanya, dinotasikan $p \oplus q$

Keempat proposisi tersebut memiliki tabel kebenaran masing-masing, seperti yang disajikan dalam Tabel 2.1

Tabel 2.1 Tabel Kebenaran 4 Proposisi

p	q	$\sim p$	$p \wedge q$	$p \vee q$	$p \oplus q$
T	T	F	T	T	F
T	F	F	F	T	T
F	T	T	F	T	T
F	F	T	F	F	F

Contoh 2.2

1. p : Hari ini tanggal 17

   q : Mahasiswa Ubhara upacara

   $p \wedge q$: Hari ini tanggal 17 **dan** mahasiswa Ubhara upacara.

2. p : Hari ini tanggal 17

   $\sim p$: Hari ini **bukan** tanggal 17.

3. p : Ketua HIMTIF harus mahasiswa Informatika

   q : Ketua HIMTIF sudah menempuh 2 semester

   $p \vee q$: Ketua HIMTIF harus mahasiswa Informatika **atau** sudah menempuh 2 semester.

4. p : Bahri sedang berenang di sungai

   q : Bahri sedang makan nasi padang

$p \oplus q$: Bahri sedang berenang di sungai **atau** sedang makan nasi padang (**hanya bisa dilakukan salah satu**).

Definisi 2.4 *Proposisi majemuk disebut* ***tautologi*** *jika ia benar untuk semua kasus, sedangkan jika ia salah untuk semua kasus, disebut* ***kontradiksi*** (Munir, 2016).

Dua buah proposisi majemuk dikatakan **ekuivalen** (" $\equiv$ ") jika keduanya memiliki table kebenaran yang identik.

Hukum De Morgan: $\sim(p \wedge q) \equiv \sim p \vee \sim q$

Tabel 2.2 Tabel Kebenaran Hukum De Morgan

p	q	$p \wedge q$	$\sim(p \wedge q)$	$\sim p$	$\sim q$	$\sim p \vee \sim q$
T	T	T	F	F	F	F
T	F	F	T	F	T	T
F	T	F	T	T	F	T
F	F	F	T	T	T	T

2.2 Proposisi Bersyarat

Implikasi disebut juga proposisi bersyarat, bentuknya adalah "jika p, maka q", dinotasikan sebagai $p \rightarrow q$, p disebut hipotesis, premis, atau kondisi, sedangkan q disebut konklusi.

Tabel 2.3 Tabel Kebenaran Implikasi

p	q	$p \rightarrow q$
T	T	T
T	F	F
F	T	T
F	F	T

Contoh 2.3

1. Jika hari ini hujan, maka saya memakai payung.

2. Jika anda ingin lulus Struktur Diskrit, maka anda harus mendapat nilai minimal C.
3. Jika anda mahasiswa Teknik Informatika, maka anda harus mengambil mata kuliah Algoritma.

Catat bahwa dalam implikasi yang menjadi **poin penting adalah nilai kebenaran premis dan konklusi**, **BUKAN hubungan sebab akibat di antara keduanya**. Apa maksudnya? Mari lihat contoh berikut.

"Jika 2+8=3, maka anda merokok"

"Jika hari ini cerah, maka Bandung termasuk Provinsi Jawa Barat"

Ada beberapa cara untuk mengekspesikan implikasi $p \rightarrow q$:

a. Jika p, maka q
b. Jika p, q
c. p mengakibatkan q
d. q jika p
e. p hanya jika q
f. p syarat cukup untuk q
g. q syarat perlu bagi p
h. q bilamana p

Contoh 2.4

1. Ubahlah proposisi berikut ke bentuk standard "jika p, maka q"

   Syarat cukup agar pom bensin meledak adalah percikan api dari rokok.

   Jawab:

   Ingat: $p \rightarrow q$ dapat dibaca p syarat cukup untuk q

Kita susun sesuai format:

Percikan api dari rokok adalah syarat cukup agar pom bensin meledak."

Kita bentuk menjadi:

p : Api memercik dari rokok

q : Pom bensin meledak

Bentuk standar: "Jika Api memercik dari rokok, maka Pom bensin meledak"

2. Dua pedagang barang kelontong mengeluarkan moto jitu untuk menarik pembeli. Pedagang pertama mengumbar moto "Barang bagus tidak murah" sedangkan pedagang kedua mempunyai moto "Barang murah tidak bagus".

   Apakah kedua moto pedagang tersebut menyatakan hal yang sama?

   Jawab:

   p : Barang itu bagus

   q : Barang itu murah

   Kita notasikan moto pedagang 1 sebagai P_1 & pedagang 2 sebagai P_2.

   P_1: "Jika barang bagus maka barang itu tidak murah" atau $p \rightarrow \sim q$

   P_2: "Jika barang itu murah maka barang itu tidak bagus" atau $q \rightarrow \sim p$

Tabel 2.4 Tabel Kebenaran Jawaban Nomor 2

p	q	$\sim p$	$\sim q$	$p \rightarrow \sim q$	$q \rightarrow \sim p$
T	T	F	F	F	F
T	F	F	T	T	T
F	T	T	F	T	T
F	F	T	T	T	T

Dari tabel tersebut dapat kita lihat pernyataan kedua pedagang tersebut ekuivalen, atau mereka menyatakan hal yang sama.

Catat: Implikasi $p \rightarrow q$ ekuivalen $\sim p \vee q$

Tabel 2.5 Tabel Kebenaran Ekuivalensi Implikasi

p	q	$\sim p$	$p \rightarrow q$	$\sim p \vee q$
T	T	F	T	T
T	F	F	F	F
F	T	T	T	T
F	F	T	T	T

Varian Proposisi Bersyarat

- Konvers: $q \rightarrow p$
- Invers: $\sim p \rightarrow \sim q$
- Kontraposisi: $\sim q \rightarrow \sim p$

Tabel 2.6 Tabel Kebenaran Ekuivalensi Implikasi

p	q	$\sim p$	$\sim q$	$p \rightarrow q$	$q \rightarrow p$	$\sim p \rightarrow \sim q$	$\sim q \rightarrow \sim p$
T	T	F	F	T	T	T	T
T	F	F	T	F	T	T	F
F	T	T	F	T	F	F	T
F	F	T	T	T	T	T	T

Contoh 2.5

Tentukan konvers, invers, dan kontraposisi dari:

"Jika Amir mempunyai mobil, maka ia orang kaya"

Jawab:

Konvers: Jika Amir orang kaya, maka ia mempunyai mobil

Invers: Jika Amir tidak mempunyai mobil, maka ia bukan orang kaya

Kontraposisi: Jika Amir bukan orang kaya, maka ia tidak mempunyai mobil

2.3 Bi-implikasi

Bi-implikasi berbentuk "p jika dan hanya jika q", dinotasikan sebagai $p \leftrightarrow q$,

Tabel 2.7 Tabel Kebenaran Bi-implikasi

p	q	$p \leftrightarrow q$
T	T	T
T	F	F
F	T	F
F	F	T

Catatan:

Bi-impikasi $p \leftrightarrow q$ ekuivalen dengan $(p \rightarrow q) \wedge (q \rightarrow p)$ atau ditulis sebagai:

$$p \leftrightarrow q \equiv (p \rightarrow q) \wedge (q \rightarrow p)$$

Artinya, "p jika dan hanya jika q" dapat dibaca "Jika p maka q dan jika q maka p".

Tabel 2.8 Tabel Kebenaran Ekuivalensi Bi-implikasi

p	q	$p \leftrightarrow q$	$p \rightarrow q$	$q \rightarrow p$	$(p \rightarrow q) \wedge (q \rightarrow p)$
T	T	T	T	T	T
T	F	F	F	T	F
F	T	F	T	F	F
F	F	T	T	T	T

Ada beberapa cara untuk mengekspesikan bi-implikasi $p \leftrightarrow q$:

a. p jika dan hanya jika q

b. Jika p, maka q, dan sebaliknya

c. p adalah syarat perlu dan cukup untuk q

d. p *iff* q

Contoh 2.6

1. Untuk $a > 0$, $|x| < a$ jika dan hanya jika $-a < x < a$.
2. 1 + 1 = 2 jika dan hanya jika 2 + 2 = 4.
3. Syarat cukup dan syarat perlu agar hari hujan adalah kelembaban udara tinggi.
4. Jika anda orang kaya maka anda mempunyai banyak uang, dan sebaliknya.
5. Bandung terletak di Jawa Barat *iff* Jawa Barat adalah sebuah propinsi di Indonesia.

Berikut ini merupakan ekivalensi logika yang bisa kita gunakan untuk melakukan normalisasi formula dalam pembuktian logika.

Tabel 2.9 Ekivalensi Logika

Bentuk Simbolik	Istilah
$\sim(\sim p) \equiv p$	Hukum Involusi
$p \vee \sim p \equiv T$ $p \wedge \sim p \equiv F$	Hukum Negasi
$p \vee F \equiv p$ $p \wedge T \equiv p$	Hukum Identitas
$p \wedge F \equiv F$ $p \vee T \equiv T$	Hukum Dominasi
$p \vee p \equiv p$ $p \wedge q \equiv p$	Hukum Idempoten
$p \vee (p \wedge q) \equiv p$ $p \wedge (p \vee q) \equiv p$	Hukum Absorpsi
$p \vee q \equiv q \vee p$ $p \wedge q \equiv q \wedge p$	Hukum Komutatif
$p \vee (q \vee r) \equiv (p \vee q) \vee r$ $p \wedge (q \wedge r) \equiv (p \wedge q) \wedge r$	Hukum Asosiatif
$p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$ $p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$	Hukum Distributif
$\sim(p \wedge q) \equiv \sim p \vee \sim q$ $\sim(p \vee q) \equiv \sim p \wedge \sim q$	Hukum De Morgan
$(p \rightarrow q) \equiv (\sim q \rightarrow \sim p)$	Kontrapositif
$(p \rightarrow q) \equiv (\sim p \vee q)$ $(p \rightarrow q) \equiv \sim(p \wedge \sim q)$	Implikasi
$(p \leftrightarrow q) \equiv [(p \rightarrow q) \wedge (q \rightarrow p)]$	Ekuivalensi
$[(p \wedge q) \rightarrow r] \equiv [p \rightarrow (q \rightarrow r)]$	Hukum Eksportasi
$(p \rightarrow q) \equiv [(p \wedge \sim q) \rightarrow p]$	Reductio ad absurdum

Sumber: (Rachmat, 2004)

Implikasi logis merupakan dasar bagi aturan inferensi yang amatlah berguna dalam pembuktian kesimpulan.

Tabel 2.10 Implikasi Logis

Bentuk Simbolik	Istilah
$p \Rightarrow (p \vee q)$	Adisi
$(p \wedge q) \Rightarrow p$	Simplifikasi
$(p \rightarrow 0) \Rightarrow \sim p$	Absurditas
$(p \wedge (p \rightarrow q)) \Rightarrow q$	Modus Ponen
$\big((p \rightarrow q) \wedge \sim q\big) \Rightarrow \sim p$	Modus Tollen
$((p \vee q) \wedge \sim p) \Rightarrow q$	Silogisme Disjungtif
$p \Rightarrow (q \rightarrow (p \wedge q))$	
$\big((p \rightarrow q) \wedge (q \rightarrow r)\big) \Rightarrow (p \rightarrow r)$	Transifitas Implikasi
$\big((p \leftrightarrow q) \wedge (q \leftrightarrow r)\big) \Rightarrow (p \leftrightarrow r)$	Transifitas Biimplikasi
a. $(p \rightarrow q) \Rightarrow \big((p \vee r) \rightarrow (q \vee r)\big)$ b. $(p \rightarrow q) \Rightarrow \big((p \wedge r) \rightarrow (q \wedge r)\big)$ c. $(p \rightarrow q) \Rightarrow \big((q \rightarrow r) \rightarrow (p \rightarrow r)\big)$	
a. $((p \rightarrow q) \wedge (r \rightarrow s) \Rightarrow \big((p \vee r) \rightarrow (q \vee s)\big)$ b. $((p \rightarrow q) \wedge (r \rightarrow s) \Rightarrow \big((p \wedge r) \rightarrow (q \wedge s)\big)$	Dilema Konstruktif
$((p \rightarrow q) \wedge (r \rightarrow s) \Rightarrow \big((\sim q \vee \sim s) \rightarrow (\sim p \vee \sim r)\big)$ $((p \rightarrow q) \wedge (r \rightarrow s) \Rightarrow \big((\sim q \wedge \sim s) \rightarrow (\sim p \wedge \sim r)\big)$	Dilema Destruktif

Sumber: (Lembang et al., 2023)

2.4 Bentuk Normal Formula

Suatu formula tersusun dari kombinasi proposisi-proposisi yang dihubungkan oleh operator logika. Kita bisa menyusun formula yang ekuivalen dengan formula implikasi atau biimplikasi. Formula yang hanya dihubungkan dengan operator $\wedge$, $\vee$, dan $\sim$ disebut formula

bentuk normal, yang terbagi menjadi dua, yakni normal konjungsi dan normal disjungsi.

Definisi 2.5 *Formula F dikatakan bentuk normal konjungsi jika F berbentuk $F_1 \wedge F_2 \wedge \ldots \wedge F_n$, sedangkan bentuk normal disjungsi jika F berbentuk $F_1 \vee F_2 \vee \ldots \vee F_n$ dengan $n \in \mathbb{Z}^+$* (Rachmat, 2004).

Jika ingin mengubah formula ke bentuk normal, kita bisa menggunakan Tabel 2.9 dan 2.10, perhatikan contoh berikut.

Contoh 2.7 Ubahlah formula $(p \wedge (q \rightarrow r)) \rightarrow s$ ke dalam bentuk normal konjungsi.

$$
\begin{aligned}
(p \wedge (q \rightarrow r)) \rightarrow s &= (p \wedge (\sim q \vee r)) \rightarrow s && \because \text{implikasi} \\
&= \sim(p \wedge (\sim q \vee r)) \vee s && \because \text{implikasi} \\
&= (\sim p \vee \sim(\sim q \vee r)) \vee s && \because \text{negasi} \\
&= (\sim p \vee (q \wedge \sim r)) \vee s && \because \text{negasi} \\
&= ((\sim p \vee q) \wedge (\sim p \vee \sim r)) \vee s && \because \text{distributif} \\
&= (\sim p \vee q \vee s) \wedge (\sim p \vee \sim r \vee s) && \because \text{distributif}
\end{aligned}
$$

Contoh 2.8 Ubahlah formula $(p \vee \sim q) \rightarrow r$ ke dalam bentuk normal konjungsi.

$$
\begin{aligned}
(p \vee \sim q) \rightarrow r &= \sim(p \vee \sim q) \vee r && \because \text{implikasi} \\
&= (\sim p \wedge q) \vee r && \because \text{negasi}
\end{aligned}
$$

2.5 Pembuktian Logika

Pada subbab ini dijelaskan mengenai proses membuktian benar atau salahnya dari penarikan suatu kesimpulan secara logika. Kita dapat menggunakan fakta-fakta yang dinyatakan dalam proposisi-proposisi yang diasumsikan benar untuk membuktikan kesimpulan. Fakta-fakta ini dapat disebut premis/aksioma/hipotesis dan

penarikan kesimpulan dari premis-premis disebut konsekuensi logis. Premis dapat dinotasikan sebagai $P_1, P_2, \dots, P_n$ dan kesimpulan C.

Metode pembuktian terdapat 2 macam, yaitu:

1. Pembuktian langsung *(direct proof)*
   Formula yang digunakan:
   $$P_1 \wedge P_2 \wedge P_3 \wedge \dots \wedge P_n \Rightarrow C$$
2. Pembuktian taklangsung *(indirect proof)*
   a. Dengan kontrapositif, formula yang digunakan:
   $$\sim C \Rightarrow \sim(P_1 \wedge P_2 \wedge P_3 \wedge \dots \wedge P_n)$$
   b. Dengan kontradiksi, formula yang digunakan:
   $$P_1 \wedge P_2 \wedge P_3 \wedge \dots \wedge P_n \wedge \sim C \Rightarrow 0$$

Selanjutnya, teknik/cara yang dapat dilakukan dalam membuktikan kesimpulan ada 3 sebagai berikut:

1. Tabel kebenaran, merupakan cara yang paling mudah dan sederhana dalam melakukan pembuktian kebenaran dari suatu kesimpulan. Namun ingat, pembuktian ini harus tetap mengacu pada 2 metode pembuktian yang telah disebutkan di atas.
   Langkah yang dilakukan dalam cara ini cukup mudah, yakni kita harus menentukan formula dari premis-premis yang diketahui. Selanjutnya menentukan metode yang ingin digunakan, metode pembuktian langsung atau tidak langsung. Apapun metodenya, kelak akan mendapatkan hasil/kesimpulan yang sama. Langkah terakhir adalah membuat tabel kebenaran berdasarkan metode yang telah kita pilih. Mari kita perhatikan contoh berikut agar dapat memahami dengan jelas teknik ini.
   Contoh 2.9 Periksalah kesimpulan dari premis berikut dengan menggunakan kedua metode pembuktian dan tabel kebenaran.

P_1: Jika ada CCTV di dalam rumah maka rumah tidak akan kemalingan.
P_2: CCTV tidak ada di dalam rumah.
Kesimpulan: Rumah pasti akan kemalingan. (*Apakah benar demikian?*)
Solusi:
Misalkan p adalah 'CCTV ada di dalam rumah', r adalah 'rumah tidak akan kemalingan', sehingga untuk metode pembuktian langsung kita bisa memformulasikan:
$\big((p \to q) \wedge {\sim}p\big) \to {\sim}q$
dan membuat tabel kebenaran sebagai berikut:

Tabel 2.11 Tabel Kebenaran untuk Pembuktian Langsung

p	q	${\sim}p$	${\sim}q$	$p \to q$	$(p \to q) \wedge {\sim}p$	$(p \to q) \wedge {\sim}p \to {\sim}q$
T	T	F	F	T	F	T
T	F	F	T	F	F	T
F	T	T	F	T	T	F
F	F	T	T	T	T	T

Berdasarkan Tabel 2.11 dapat dikatakan bahwa interpretasi dari argumen $\big((p \to q) \wedge {\sim}p\big) \to {\sim}q$ tidak semuanya benar. Dengan demikian, kesimpulan 'Rumah pasti akan kemalingan' adalah salah.

Selanjutnya, kita akan coba kerjakan dengan metode pembuktian tidak langsung (dengan kontradiksi). Formula yang bisa kita susun adalah:
$\big((p \to q) \wedge {\sim}p\big) \wedge q$
dengan tabel kebenaran:

Tabel 2.12 Tabel Kebenaran untuk Pembuktian Tidak Langsung

	q	$\sim p$	$p \to q$	$(p \to q) \wedge \sim p$	$(p \to q) \wedge \sim p \wedge q$
T	T	F	T	F	F
T	F	F	F	F	F
F	T	T	T	T	T
F	F	T	T	T	F

Karena dibuktikan menggunakan pembuktian kontradiksi, maka hasil inkonsisten yang kita harapkan, tetapi nyatanya berdasarkan Tabel 2.12 interpretasi $\big((p \to q) \wedge \sim p\big) \wedge q$ tidak salah semuanya (bukan kontradiksi), sehingga kesimpulan salah.

2. Normalisasi

Pembuktian dengan cara normalisasi dilakukan dengan menghubungkan premis-premis yang diketahui dengan menggunakan operator logika $\wedge$ lalu disederhanakan ke bentuk normal konjungsi atau disjungsi. Dalam proses ini, diperbolehkan melakukan penggantian atau substitusi formula dengan formula lain yang ekuivalen.

Contoh 2.10 Harga saham turun jika apabila suku bunga naik. Banyak investor kecewa jika harga saham turun. Saat ini, suku bunga naik. Tunjukkanlah bahwa dapat ditarik kesimpulan banyak investor kecewa (Rachmat, 2004).

Solusi:

Dari argumen yang diketahui dan kesimpulan dapat ditulis:

P_1: jika apabila suku bunga naik (p) maka harga saham turun (s), dapat dinyatakan dengan formula $p \to s$.

P_2: jika harga saham turun (s) maka banyak investor kecewa (u), dapat dinyatakan dengan formula $s \to u$.

P_3: suku bunga naik (p).

C: banyak investor kecewa (u).

Dengan mengacu pada pembuktian langsung maka kita dapat hubungkan semua premis dengan operator $\wedge$, akibatnya:

$$\begin{aligned}(p \to s) \wedge (s \to u) \wedge p &\Leftrightarrow (\sim p \vee s) \wedge (\sim s \vee u) \wedge p \\ &\Leftrightarrow p \wedge (\sim p \vee s) \wedge (\sim s \vee u) \\ &\Leftrightarrow (p \wedge \sim p) \vee (p \wedge s) \wedge (\sim s \vee u) \\ &\Leftrightarrow F \vee (p \wedge s) \wedge (\sim s \vee u) \\ &\Leftrightarrow (p \wedge s) \wedge (\sim s \vee u) \\ &\Leftrightarrow (p \wedge s \wedge \sim s) \vee (p \wedge s \wedge u) \\ &\Leftrightarrow (p \wedge F) \vee (p \wedge s \wedge u) \\ &\Leftrightarrow (p \wedge s \wedge u)\end{aligned}$$

Karena semua premis adalah benar maka $p \wedge s \wedge u$ benar.

Formula $p \wedge s \wedge u$ adalah benar hanya jika p, s, u benar.

Dengan demikian, u adalah benar, artinya kesimpulan benar.

3. Aturan Inferensi

Berdasarkan metode pembuktian langsung, kita dapat menarik kesimpulan atau **konklusi** C dari proposisi-proposisi dan bisa ditulis sebagai berikut:

$$P_1 \wedge P_2 \wedge P_3 \wedge \ldots \wedge P_n \Rightarrow C$$

Proses penarikan kesimpulan ini disebut **inferensi** dengan menggunakan kaidah/aturan inferensi dalam Tabel 2.13.

Tabel 2.13 Aturan Inferensi

No.	Bentuk Simbolik	Istilah
1.	$p \to q$ p $\therefore q$	Modus Ponen
2.	$p \to q$ $\sim q$ $\therefore \sim p$	Modus Tollen
3.	$p \to q$ $q \to r$ $\therefore p \to r$	Modus Hipotetis
4.	$p \vee q$ $\sim p$ $\therefore q$	Silogisme Disjungtif
5.	$p \wedge q$ $\therefore p$	Simplifikasi
6.	p $\therefore p \vee q$	Penjumlahan Disjungtif
7.	p q $\therefore p \wedge q$	Konjungsi

Sumber: (Lembang et al., 2023)

Contoh 2.11 Buktikan bahwa $s \to r$ adalah kesimpulan dari premis $p \to (q \to r), p \vee \sim s$, dan q!

Penyelesaian: Kita gunakan pembuktian langsung yakni

$P_1 \wedge P_2 \wedge P_3 \wedge \ldots \wedge P_n \Rightarrow C$

menggunakan aturan inferensi, ekivalensi logika, dan implikasi logis.

Langkah		Alasan
1.	$p \to (q \to r)$	Premis 1
2.	$p \vee \sim s$	Premis 2
3.	q	Premis 3
4.	$\sim s \vee p$	Langkah 2: Hukum Komutatif
5.	$s \to p$	Langkah 4: Implikasi
6.	$s \to (q \to r)$	Langkah 1 dan 5
7.	$(s \wedge q) \to r$	Langkah 6: Hukum Eksportasi
8.	$q \to (s \to (q \wedge s))$	Tautologi
9.	$s \to (q \wedge s)$	Langkah 3 & 8: Modus Ponen
10.	$s \to (s \wedge q)$	Langkah 9: Hukum Komutatif
11.	$s \to r$	Langkah 7 & 10; Modus Hipotetis

$\therefore$ Terbukti bahwa $s \to r$ merupakan kesimpulan dari premis $p \to (q \to r), p \vee \sim s, q$.

2.6 Logika Predikat

Subbab terakhir ini merupakan penutup sekaligus pelengkap dari subbab sebelumnya. Pada bab tersebut, kita bisa melihat bahwa kalkulus proposisi hanya menangani premis-premis yang terbatas. Faktanya, dalam kehidupan sehari-hari kerap kali kita menghadapi suatu premis yang banyaknya tidak terbatas. Untuk jelasnya mari kita perhatikan contoh berikut.

Contoh 2.12 Setiap manusia pasti mati. Karena Amir adalah manusia, maka Amir pasti mati. Secara intuisi, kesimpulan tersebut

terasa benar. Jika kita buat formula dan menggunakan kalkulus proposisi,

diperoleh:

p: Setiap manusia pasti mati

q: Amir adalah manusia

r: Amir pasti mati

Namun, kita tidak bisa menyatakan r sebagai konsekuensi logis dari p dan q. Mengapa? Karena pernyataan 'setiap manusia' memberikan arti kumpulan dari manusia, sedangkan Amir adalah individu yang merupakan bagian dari kumpulan manusia. Hubungan pernyataan seperti ini tidak dikenali dalam kalkulus proposisi. Jika kita ingin membuktikan kebenaran dari 'Setiap manusia pasti mati' maka kita harus membuktikan kebenaran dari semua manusia (anggota kumpulan manusia). Hal tersebut tidak mungkin dilakukan dibuktikan dalam kalkulus proposisi, untuk itu pada bab ini akan dibahas mengenai logika predikat.

Definisi 2.6 *Predikat adalah suatu pernyataan yang memiliki satu atau lebih variabel, dinotasikan dengan simbol huruf kapital dan variabel huruf kecil seperti G(x). Jika nilai-nilainya diberikan ke dalam semua variabel dalam suatu predikat, akan menghasilkan pernyataan dalam sebuat proposisi(Gunawan, 2022).*

Sebagai contoh, $x \leq 10$ merupakan predikat karena x merupakan suatu variabel, yakni bilangan real. Jika kita substitusikan sebuah bilangan real ke dalam x, maka akan didapat sebuah proposisi. Misal $x = 5$ dan $x = 12$, sehingga $5 < 10$ dan $12 < 10$yang secara berurutan merupakan pernyataan proposisi yang bernilai benar dan salah. Proposisi logika dapat mengandung kuantor, yang terbagi menjadi 2 yakni universal dan eksistensial.

Definisi 2.7 *Simbol kuantor universal adalah* $\forall$ *yang memberikan arti 'untuk setiap', 'semua', sedangkan untuk kuantor eksistensial disimbolkan dengan* $\exists$ *yang berarti 'sebagian','terdapat', 'beberapa'.*

Pernyataan terdapat x sehingga $x > 4$, dinyatakan $G(x): x > 4$, ditulis $\exists x, G(x)$. Pernyataan untuk setiap $x, x < 3$ atau $x > 10$, dapat dinyatakan $P(x): x < 3$, $Q(x): x > 10$, ditulis $\forall x\ (P(x) \vee Q(x))$.

DAFTAR PUSTAKA

Gunawan, P. H. (2022). *Logika Matematika untuk Analisis Algoritma* (L. mayasari, Ed.). ANDI.

Munir, R. (2016). *Matematika Diskrit* (6th ed.). Informatika Bandung.

Rachmat, S. (2004). *Pengantar Logika Matematika*. Informatika Bandung.

Lembang, S. T., Langi', E. L., Sari, R., Dairoh, Sriwahyuningrum, A., Palayukan, H., Lestari, I., Damayanti, N. P. R., Ramdhania, K. F., Hartati, L., Palengka, I., Remme, B. V., Herlinawati, E., Damayanti, I. D., & Ayu, R. W. S. (2023). *Logika Matematika* (S. Haryanti, Ed.). Media Sains Indonesia.

BAB 3
RELASI DAN FUNGSI

Oleh Mindo H. Sinambela

3.1 RELASI

3.1.1 Definisi Relasi

1. Diketahui A dan B adalah dua buah Himpunan, maka diperoleh hasil kali silang (*Cross Product)* dari A dan B adalah $A \times B = \{(a, b) | a \in A \, dan \, b \in B$
2. Jika A dan B merupakan dua himpunan tak kosong, maka suatu relasi T biner dari A ke B adalah suatu himpunan bagian dari A x B. Jika A = B, **maka T disebut relasi biner** pada A.
3. Sebuah Relasi dari *A* ke *B* adalah subhimpunan dari *A* x *B* yang jika dinotasikan

Notasinva : $R \subseteq A \, X \, B$

Keterangan:
A : Domain (Daerah Asal) dari R
B : Kodomain (daerah lawan/kawan) dan bisa juga merupakan Range dari *R*

Contoh 1:
Misalkan $A = \{1,2,3\}, \text{dan } B = \{a, b\}$. Tentukanlah hasil kali silang antara $\times B$!
Jawab:
Maka hasil kali silang antara $A \times B$ adalah:
$A \times B = \{(1, a), (1, b), (2, a), (2, b), (3, a), 3, b)\}$

4. Jika A dan B merupakan himpunan berhingga, maka $|\boldsymbol{A \times B}| = |\boldsymbol{A}| \times |\boldsymbol{B}|$
5. Pasangan berurutan (a, b) berbeda dengan (b, a) dapat dituliskan $(a, b) \neq (b, a)$
6. Perkalian Kartesian tidak komutatif, yaitu $A \times B \neq B \times A$, dengan syarat A $atau$ B ***bukan*** merupakan himpunan kosong
7. Jika $A = \emptyset \; atau \; B = \emptyset, maka \; A \times B = B \times A = \emptyset$

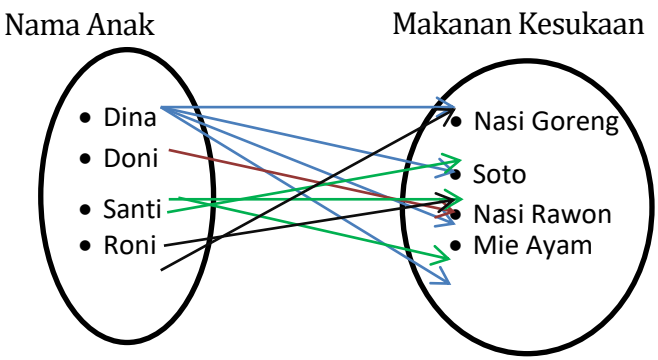
3.1.2 Cara Menyatakan Relasi

Relasi dapat dinyatakan dengan beberapa cara

- Menggunakan Diagram Panah

  Contohnya:

  Sebuah keluarga terdiri dari anak yang bernama Dina, Doni, Santi dan Roni. Mereka mempunyai makanan kesukaan masing. Dina menyukai Nasi goreng, Soto, Nasi Rawon dan Mie Ayam. Doni menyukai Nasi Rawon. Santi menyukai Soto, Nasi Rawon dan Mie Ayam. Sedangkan Roni menyukai Nasi Goreng dan Nasi Rawon. Buatlah relasinya dengan menggunakan diagram Panah!



Gambar 1. Relasi Makanan Kesukaan

- Menggunakan Tabel

  Kolom pertama tabel menyatakan daerah Asal, sedangkan kolom kedua menyatakan daerah hasil

  Tabel 1. Makanan kesukaan

Nama	Makanan Kesukaan
Dina	Nasi goreng
Dina	Soto
Dina	Nasi Rawon
Dina	Mie Ayam
Doni	Nasi Rawon
Santi	Soto
Santi	Nasi Rawon
Santi	Mie Ayam
Roni	Nasi Goreng
Roni	Nasi Rawon

- Dengan pasangan Berurutan

  Contoh :
  {(Dina, Nasi Goreng), (Dina, Soto), (Dina, Nasi Rawon),(Dina, Mie Ayam),(Doni, Nasi Rawon), (Santi Soto), (Santi,Nasi Rawon), (Santi, Mie Ayam),(Roni, Nassi Goreng), (Roni, Nasi Rawon)}

- Dengan Matriks

  Misalkan R adalah relasi dari $A = \{a_1, a_2, \dots, a_m\}$ dan $B = \{b_1, b_2, \dots, b_n\}$, maka Relasi R dapat disajikan dengan matriks $M = [m_{ij}]$,

$$\begin{array}{c} \\ a_1 \\ a_2 \\ \vdots \\ a_m \end{array} \begin{array}{c} \begin{array}{cccc} b_1 & b_2 & \dots & b_n \end{array} \\ \begin{bmatrix} m_{11} & m_{12} & \dots & m_{1n} \\ m_{21} & m_{22} & \dots & m_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ m_{m1} & m_{m2} & \dots & m_{mn} \end{bmatrix} \end{array}$$

Yang dalam hal ini

$$m_{ij} = \begin{cases} 1, (a_i, b_j) \in R \\ 0, (a_i, b_j) \notin R \end{cases}$$

Dengan kata lain elemen matriks pada posisi (i,j) bernilai 1 jika a_i dihubungkan dengan b_j dan bernilai 0 jika a_i tidak dihubungkan dengan b_j. Relasi matriks merupakan contoh matriks *zero-one*

Relasi pada contoh penggunaan digram panah dapat dinyatakan dengan matriks

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \end{pmatrix}$$

Yang dalam hal ini $a_1 = Dina$, a_2= Doni, $a_3 = Santi$, $a_4 = Roni$ dan b_1 =Nasi Goreng, b_2=Soto, $b_3 = Nasi\ Rawon$, dan b_4= Mie Ayam

3.1.3 Sifat-sifat Relasi

1. **Refleksif**

   Misalkan S adalah himpunan tak kosong, relasi R pada S disebut Refleksif jika

   (a,a) ∈ **S**, hal ini berarti bahwa di dalam refleksif setiap elemen di dalam S berhubungan dengan dirinya sendiri dan memiliki arti juga bahwa relasi R pada himpunan S tidak refleksif jika ada a ϵ S sedemikian sehingga (a,a) ∉ R.

   Contoh 1 :

   Misalkan A ={a,b,c,d}. Dan relasi R di bawah ini didefinisikan pada himpunan A, maka

a. Relasi R ={(a,a,),(a,c),(b,a),(b,b),(c,c),(d,b),(d,c),(d,d)} bersifat refleksif karena terdapat elemen relasi yang berbentuk Refleksif (a,a) yaitu (a,a),(b,b), (c,c) dan (d,d)
b. Relasi R ={(a,a), (b,b),(b,c), (d,b),(d,c), (d,d) tidak bersifat refleksif karena (c,c) $\notin R$

Contoh 2. :

Relasi "Habis membagi" pada himpunan bilangan bulat positif bersifat refleksif karena setiap bilangan bulat positif selalu habis membagi dirinya sendiri (a,a) ϵR ϵ untuk setiap a ϵA

Contoh 3.

Tiga buah relasi dibawah ini menyatakan relasi pada himpunan bilangan bulat positif N

$R: x \; lebih \; kecil \; dari \; y$

$S: x + y = 5$

$T: 3x + y = 10$

Penyelesaiannya : Tidak satupun dari ketiga relasi di atas yang termaksud refleksif karena, misalkan kita ambil bilangan positif N nya (2,2) maka

(2,2) $\notin S \notin R \; maupuan \; \notin T$.

2. Simetris

Relasi R pada himpunan A disebut simetris jika (a,b) ϵR maka (b,a) ϵR, untuk semua a,b ϵA

Contoh :

1. Misalkan A adalah himpunana mahasiswa di sebuah kampus dan R adalah relasi pada A sedemikian sehingga (a,b) ϵ R jika dan hanya jika a

satu prodi dengan b. Jelas bahwa b satu prodi dengan a

2. T adalah relasi pada himpunan bilangan bulat positif sedemikian sehingga (a,b) ϵT jika dan hanya jika $a \geq b$
3. Misalkan A={1,2,3,4}, dan relasi R didefeniskan sebagai berikut pada himpunan A, Maka R={(1,1),(1,2), (2,1),(2,2),(2,4),(4,2),(4,4)} bersifat simetris karena jika (A,b) ϵR Maka (b,a) juga ϵR. Dalam hal ini bisa diperhatikan (1,2) dan (2,1) ϵR
4. Relasi "Habis membagi" pada himpunan bilangan bulat bersifat tidak simetris karena jika a habis membagi b , b tidak habis membagi a, kecuali a=b

3. Transitif

Relasi R pada himpunan A disebut transitif jika (a,b) ϵR dan (b,c) ϵR, maka (a,c) ϵR untuk semua a,b,c ϵA

Contoh :

a. Misalkan A adalah himpunan orang, dan R adalah relasi pada A sedemikian sehingga (a,b) ϵR jika dan hanya jika b adalah keturunan a. Jelas , jika b adalah keturunan a yaitu (a,b) ϵR, dan c adalah keturunan b yaitu (b,c) ϵR, maka c juga keturunan a yaitu (a,c) ϵR
b. Misalkan A={1,2,3,4} dan relasi R didefinisikan pada himpunan A, maka R={(2,1),(3,1),(3,2),(4,1),(4,2),(4,3)} adalah bersifat transitif

Dibuktikan dengan penggunaan tabel

(a,b)	(b,c)	(a,c)
(3,2)	(2,1)	(3,1)
(4,2)	(2,1)	(4,1)
(4,3)	(3,1)	(4,1)

Tetapi, jika R ={(1,1), (2,3),(2,4),(4,2)} maka himpunan tersebut tidak bisa dikatakan relasi transitif karena (2,4) dan (4,2) $\epsilon\ R$ tetapi (2,2) $\notin\ R$
(2,3) dan (4,2) $\epsilon\ R$ tetapi (2,2) $\notin R$

Relasi keterbagian pada bilangan bulat (disimbolkan dengan |) dengan defenisi untuk $a, b \in Z\{0\}$.
Suatu Relasi pada S disebut Relasi Ekivalen, apabila memenuhi sifat refleksif, simetris dan transitif.

Contoh :

1. Misalkan $Q = \left\{\frac{p}{q} | p, q \in Z; q \neq 0\right\}$. Buktikan bahwa relasi $\sim$ pada Q dengan aturan $\frac{p}{q} \sim \frac{m}{n}$ jika dan hanya jika $pn = mq$. Sehingga relasi $\sim$ merupakan relasi ekuivalen

Bukti :

Untuk membuktikan bahwa relasi $\sim$ merupakan relasi ekuivalen, maka akan dibuktikan bahwa $\sim$ memiliki 3 sifat relasi.

a) Sifat Refleksif yaitu $\frac{p}{q} \sim \frac{p}{q} \Leftrightarrow pq = pq$

b) Sifat Simetris yaitu $\frac{p}{q} \sim \frac{m}{n} \longrightarrow \frac{m}{n} \sim \frac{p}{q}$

$$\frac{p}{q} \sim \frac{m}{n} \Leftrightarrow pn = mq \ldots\ldots\ldots (1)$$
$$\frac{m}{n} \sim \frac{p}{q} \Leftrightarrow mq = pn \ldots\ldots\ldots (2)$$

Dari persamaan (1) dan (2) dapaat dinyatakan bahwa $pn = mq \equiv mq = pn$, dengan kata lain bahwa $mq = pn$

c) Sifat Transitif yaitu $\frac{p}{q} \sim \frac{m}{n}, \frac{m}{n} \sim \frac{s}{t} \longrightarrow \frac{p}{q} \sim \frac{s}{t}$

$\frac{p}{q} \sim \frac{m}{n}$ $artinya\, pn = mq$ …… (1)

$\frac{m}{n} \sim \frac{s}{t}$ $artinya\, mt = sn$ …… (2)

Dengan mensubsitusikan $m = \frac{pn}{q}$ sehingga dari persaman (2) diperoleh :

$\frac{pn}{q} \times t = sn$

$pt = sq$

$pt = sq$ yang berarti $\frac{p}{q} \sim \frac{s}{t}$

Berdasarkan, pembuktian a),b),c) maka terbukti bahwa "$\sim$" merupakan relasi ekivalen

2. Selidiki relasi $a|b \iff b = ac$ untuk c $\in \mathbb{Z}|\{0\}$

Pembuktian :

a. Sifat relasi Refleksif $(a|a)$

Ambil $a \in \mathbb{Z}|\{0\}$ sembarang, sehingga

$a = a.1, \exists\, c = 1 \in \mathbb{Z}|\{0\}$

$a = a.1\, maka\, a|a$ …….. (Refleksif)

b. Sifat Simetris $(a|b \Rightarrow b|a)$

Ambil $a, b \in \mathbb{Z}|\{0\}$ sembarang,

Misalkan kita ambil $a = 2\, dan\, b = 4$

Sehingga $a|b$ menjadi $2|4$ tapi $4 \nmid 2$

$4 \nmid 2$ karena $2 = 4.c$, Tidak ada $c \in \mathbb{Z}|\{0\}$ yang memenuhi $4|2$……(Tidak Simetris)

c. Sifat Transitif ($a|b, b|c \Rightarrow a|c$)

Ambil $a, b, c \in \mathbb{Z}|\{0\}$ sembarang

$a|b \Leftrightarrow b = a.n$

$b|c \Leftrightarrow c = b.m$

$a|c \Leftrightarrow c = b.m$

$c = a.n.m$

$c = a.r; r = nm$

$c = ar \Longrightarrow a|c$……(Tidak Transitif)

Berdasarkan pembuktian di atas maka disimpulkan bahwa relasi '|" bukan relasi ekivalen.

3.1.4 Relasi Inversi

Misalkan R adalah relasi dari himpunan A ke himpunan B. Invers dari relasi R, dilambangkan dengan R^{-1}, adalah relasi dari B ke A yang didefinisikan oleh:

$\boldsymbol{R^{-1}}$ **= {(b, a) | (a, b)** $\in \mathbb{R}$

Contoh:

Misalkan P = {3, 6, 8} dan Q = {3, 6, 8, 12, 16}. Jika kita definisikan relasi R dari P ke Q dengan (p,q) ∈ R jika p habis membagi q maka kita peroleh:

R = {(3, 3), (3, 6), (3,12), (6, 6), (6, 12), (8, 8), (8, 16)}

$\boldsymbol{R^{-1}}$adalah invers dari relasi R, yaitu relasi dari Q ke P dengan (q, p) ∈ $\boldsymbol{R^{-1}}$ jika q adalah kelipatan dari p maka kita peroleh

$\boldsymbol{R^{-1}}$ = {(2, 2), (6, 3), (12, 3), (6, 6), (12, 6), (8, 8), (16, 8)}

3.1.5 Mengkombinasikan Relasi

Karena relasi biner merupakan himpunan pasangan terurut, maka operasi himpunan seperti irisan, gabungan,

selisih, dan beda setangkup antara dua relasi atau lebih juga berlaku.

Jika R1 dan R2 masing-masing adalah relasi dari himpunan A ke himpunan B, maka

$R1 \cap R2$, $R1 \cup R2$, $R1 - R2$, dan $R1 \oplus R2$ juga adalah relasi dari A ke B.

$R1 \cap R2 = \{(x, y) | (x, y) \in R \text{ dan } (x, y) \in R\}$

$R1 \cup R2 = \{(x, y) | (x, y) \in R \text{ atau } (x, y) \in R\}$

$R1 - R2 = \{(x, y) | (x, y) \in R \text{ dan } (x, y) \notin R\}$

$R2 - R1 = \{(x, y) | (x, y) \in R \text{ dan } (x, y) \notin R\}$

$R1 \oplus R2 = (R1 \cup R2) - (R1 \cap R2)$

$R1 \oplus R2 = (R1 - R2) \cup (R2 - R1)$

Contoh:

Misalkan A = {1, 2, 3} dan B = {1, 2, 3, 4}.

Relasi R1 = {(1, 1), (2, 2), (3, 3)}

Relasi R2 = {(1, 1), (1, 2), (1, 3), (1, 4)}

$R1 \cap R2$ = {(1,1)}

$R1 \cup R2$ = {(1, 1), (2, 2), (3, 3), (1, 2), (1, 3), (1, 4)}

$R1 - R2$ = {(2, 2), (3, 3)}

$R2 - R1$ = {(1, 2), (1, 3), (1, 4)}

$R1 \oplus R2$ = {(2, 2), (3, 3), (1, 2), 1, 3), (1, 4)}

3.2 FUNGSI (PEMETAAN)

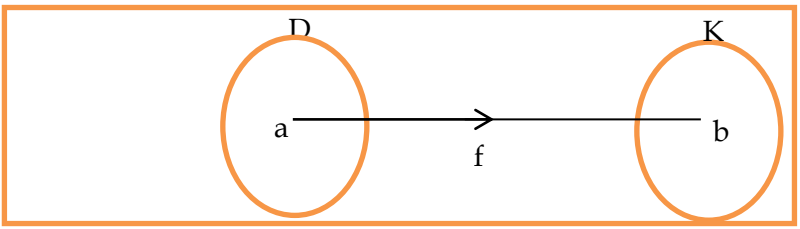
3.2.1 Defenisi

Pemetaan adalah suatu himpunan Domain (daerah asal) ke himpunan Kodomain (daerah lawan/kawan) adalah aturan pengawanan (korespondensi) yang mengawankan setiap elemen dari Domain dengan tepat satu elemen pada Kodomain. Jika dibentuk dalam bentuk notasi himpunan

$$f: D \rightarrow K$$

$$x \rightarrow f(x), \forall x \in D$$

Jika dalam bentuk diagram venn



Gambar 2. Fungsi

Contoh fungsi :

a. Misalkan A= {1,2,3} dan B ={u,v,w}. Jika A dipetakan ke B maka, $f = \{(1,u),(2,v),(3,w)\}$

b. Diberikan suatu relasi α berikut :

$\boldsymbol{\alpha}: \mathbb{R} \to \mathbb{R}, x \mapsto \alpha(x) = x^2 + 3x - 1$

Buktikan bahwa α merupakan pemetaan

Penyelesaian:

Langkah pertama: Ambil sembarang $x_1, x_2 \epsilon\ \mathbb{R}\ \ni\ x_1 = x_2$

Langkah kedua:

$$\alpha(x_1) = x_1^2 + 3x_1 - 1 = x_2^2 + 3x_2 - 1 = \alpha(x_2)$$

Ternyata $\alpha(x_1) = \alpha(x_2)$, sehingga terbukti bahwa α merupakan pemetaan

3.2.2. Ciri-Ciri Fungsi

a. Setiap elemen domain dipasangkan dengan suatu elemen kodomain. Dengan kata lain tidak boleh ada satupun elemen dari domain yang tidak mempunyai kawan di kodomain

b. Setiap elemen domain dipasangkan dengan tepat satu elemen di kodomain. Artinya bahwa satu elemen di

domain hanya berpasangan dengan satu elemen di kodomain dan beberapa elemen di domain boleh dipasangakan dengan satu kodomain tetapi tidak sebaliknya.

3.2.3.Jenis-Jenis Fungsi

Bergantung pada bayangan, fungsi dibedakan

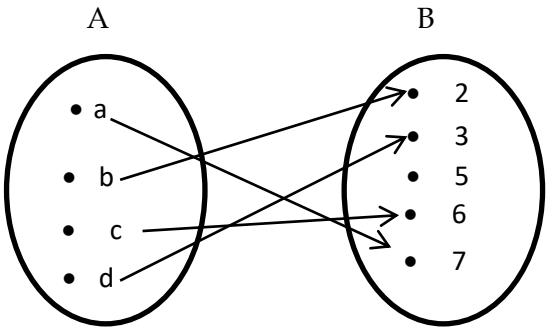
a. Fungsi **satu-ke-satu** (*one-to-one*) atau **injektif** (*injective*) jika tidak ada dua elemen himpunan A yang memiliki bayangan sama.

$$f(a_1) = f(a_2) \text{ maka } a_1 = a_2$$

Atau

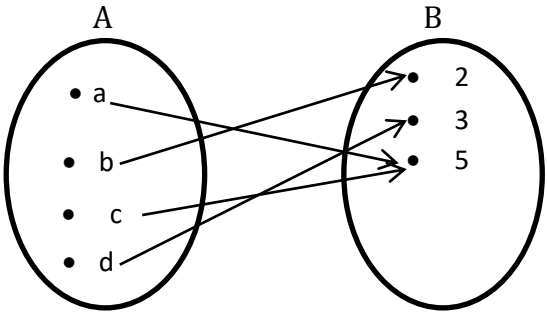
$$a_1 \neq a_2 \; maka \; f(a_1) \neq f(a_2)$$

Dapat dilihat pada diagram panah di bawah ini :



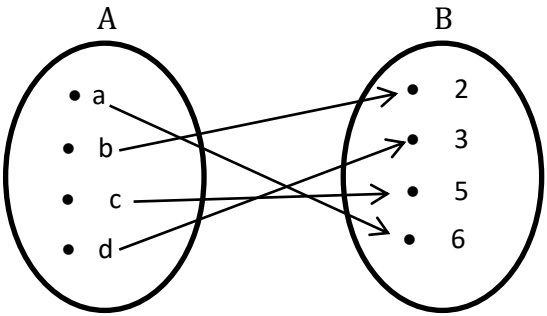
Gambar 3. Fungsi Injektif

b. Fungsi **pada** (*onto*) atau **surjektif** (*surjective*) jika setiap elemen himpunan B merupakan bayangan dari satu atau lebih elemen himpunan A.Dengan kata lain seluruh elemen B merupakan jelajah dari f atau untuk setiap $b \in B$ terdapat $a \in A$, sehingga berlaku $b = f(a)$.



Gambar 4. Fungsi Surjektif

c. Fungsi **Bijektif** yaitu fungsi yang bersifat injektif dan surjektif



Gambar 5. Fungsi Bijektif

Contoh :

Misalkan $f:\mathbb{Z} \to \mathbb{Z}$. Tentukan apakah $f(x) = x^2 + 1$ merupakan fungsi injektif, surjektif dan bijektif!

Jawab :

a. Untuk mengetahui apakah $f(x)$ merupakan injektif maka $f(x_1) = f(x_2)$ maka $x_1 = x_2$ atau $x_1 \neq x_2$ *maka* $f(x_1) \neq f(x_2)$.

Ambil sebarang $x_1, x_2 \in \mathbb{Z}$. Misalkan kita ambil nilai $x_1 = 2$ dan $x_2 = -2$, maka

$f(x_1) = f(x_2)$

${x_1}^2 + 1 = {x_2}^2 + 1$

$$2^2 + 1 = (-2)^2 + 1$$
$$5 = 5$$

Terbukti bahwa $f(x) = x^2 + 1$ bukan fungsi injektif karena tidak memenuhi syaratnya yaitu $f(x_1) = f(x_2)$ maka $x_1 = x_2$

b. Untuk mengetahui apakah $f(x)$ merupakan Surjektif maka $y = f(x)$

$f(x) = x^2 + 1$ bukanlah fungsi pada karena tidak semua nilai bilangan bulat merupakan jelajah dari *f.* Misalkan $y = 0 \in \mathbb{Z}$. Tidak ada nilai x yang memenuhi $x^2 + 1 = 0$

c. Bukan merupan fungsi Bijektif karena tidak memenuhi sifat injektif dan surjektif

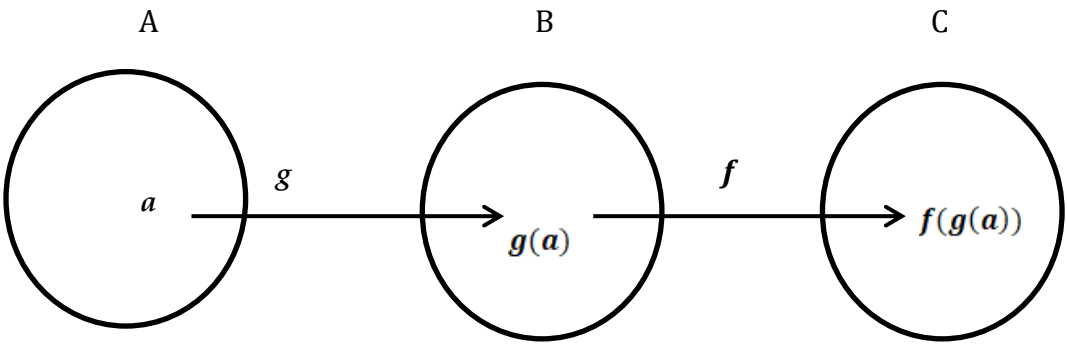
3.2.4 Fungsi Invers Fungsi

a. Jika *f* adalah fungsi berkoresponden satu-ke-satu dari *A* ke *B*, maka kita dapat menemukan **balikan** (*invers*) dari *f*.
b. Balikan fungsi dilambangkan dengan f –1. Misalkan *a* adalah anggota himpunan *A* dan *b* adalah anggota himpunan *B*, maka f -1(*b*) = *a* jika *f*(*a*) = *b*.
c. Fungsi yang berkoresponden satu-ke-satu sering dinamakan juga fungsi yang *invertible* (dapat dibalikkan), karena kita dapat mendefinisikan fungsi balikannya. Sebuah fungsi dikatakan *not invertible* (tidak dapat dibalikkan) jika ia bukan fungsi yang berkoresponden satu-ke-satu, karena fungsi balikannya tidak ada.

3.2.5 Pemetaan Komposisi

Misalkan g adalah fungsi dari himpunan A ke himpunan B, dan f adalah fungsi dari himpunan B ke himpunan C. Komposisi f dan g dinotasikan dengan $f \circ g$. Sehingga fungsi dari A ke C didefenisikan sebagai berikut :

$$(f \circ g)(a) = f(g(a))$$



Gambar 6. Fungsi Komposisi

Contoh :

Didefenisikan fungsi sebagai berikut :

$f : \mathbb{R} \longrightarrow \mathbb{R}$, dan $g : \mathbb{R} \longrightarrow \mathbb{R}$

Jika diketahui

$f(x) = 2x^3 - 3x + 1$ *dan* $g(x) = 3x + 2, \forall x \in \mathbb{R}$.Tentukanlah :

a. $(f \circ g)(x) = \cdots$

b. $(g \circ f)(x) = \cdots$

Penyelesaian

a. $(f \circ g)(x) = f\big(g(x)\big)$

$= f(3x + 2)$

$= 2(3x - 3)^3 - 3(3x + 2) + 1$

$$= 2(27x^3 - 32x^2 + 81x - 27) - 3(3x + 2) + 1$$
$$= 54x^3 - 64x^2 + 162x - 54 - 9x - 6 + 1$$
$$= 54x^3 - 64x^2 - 153x - 59$$

b. $(g \circ f)(x) = g(f(x)$

$$= g(2x^3 - 3x + 1)$$
$$= 3(2x^3 - 3x + 1) + 2$$
$$= 6x^3 - 9x + 3 + 2$$
$$= 6x^3 - 9x + 5$$

Dari contoh soal di atas menunjukkan bahwa
$(f \circ g)(x) \neq (g \circ f)(x)$

DAFTAR PUSTAKA

Munir, R., 2010. Matematika Diskrit Edisi 3. *Bandung: Informatika Bandung* .

Muhsetyo,G.2010. *Matematika Diskrit.*Jakarta.Universita Terbuka

Jek Siang, J., 2006. Matematika Diskrit dan Aplikasinya Pada Komputer.

Wibisono, S.2008.*Matematika Diskri Edisi 2.* Yogyakarta. Graha Ilmu

BAB 4
KOMPLEKSITAS ALGORITMA

Oleh Ida Fitriana Ambarsari

4.1 Algoritma

Algoritma adalah langkah-langkah yang disusun secara sistematis untuk menyelesaikan suatu masalah. Suatu algoritma tidak hanya harus benar, tetapi juga harus mangkus (*efisien*). Algoritma yang benar sekalipun mungkin tidak berguna untuk jenis dan ukuran masukan tertentu karena waktu yang diperlukan untuk menjalankan algoritma tersebut atau ruang memori yang diperlukan untuk struktur datanya terlalu besar. Misalnya kita ingin menentukan berapa banyak himpunan bagian dari suatu himpunan yang mengandung elemen x. Jika kita tulis algoritmanya, maka algoritma harus menguji semua himpunan bagian dan memeriksa apakah x merupakan anggota himpunan bagian tersebut. Jika kardinalitas himpunan adalah n, maka algoritma harus menguji sebanyak 2^n himpunan bagian. Semakin besar nilai n, waktu yang diperlukan oleh algoritma tersebut tumbuh sangat cepat. Sehingga, untuk ukuran masukan yang besar algoritma tersebut menjadi tidak mangkus. Masalah kemangkusan (*efficiency*) algoritma merupakan pokok bahasan bab ini.

4.2 Kemangkusan Algoritma

Algoritma yang bagus adalah algoritma yang mangkus. Kemangkusan algoritma diukur dari berapa jumlah waktu dan ruang (*space*) memori yang dibutuhkan untuk menjalankan. Algoritma yang mangkus adalah algoritma yang meminimumkan kebutuhan waktu dan ruang (Munir, 2010).

Kebutuhan waktu dan ruang suatu algoritma bergantung pada ukuran masukan, yang secara khas adalah jumlah data, yang diproses. Ukuran masukan itu disimbolkan dengan n. Misalnya, jika mengurutkan 1000 elemen larik, maka n adalah 1000; menghitung 6! maka $n = 6$; dan lain-lain. Waktu/ruang yang dibutuhkan oleh algoritma dinyatakan sebagai fungsi dari n. Jika n meningkat, maka sumber daya waktu/ruang yang dibutuhkan juga meningkat. Seberapa besar peningkatan sumber daya itu menentukan apakah algoritmanya mangkus atau tidak.

Kemangkusan algoritma juga berguna dalam membanding-bandingkan algoritma. Suatu masalah dapat memiliki lebih dari satu algoritma penyelesaian. Misalnya, untuk mengurutkan elemen larik (*array*) terdapat beberapa algoritma, seperti algoritma pengurutan apung (*bubble sort*), pengurutan sisipan (*insertion sort*), pengurutan seleksi (*selection sort*), pengurutan gabung (*merge sort*), pengurutan cepat (*quick sort*) dan masih banyak lagi algoritma pengurutan lainnya. Jika algoritma-algoritma tersebut akan dipertimbangkan untuk mengurutkan n data, pertanyaan yang sering muncul adalah

bagaimana memilih algoritma yang terbaik untuk diimplementasikan. Lebih lanjut, jawaban untuk pertanyaan tersebut adalah kita memerlukan kriteria formal yang digunakan untuk menilai algoritma yang terbaik. Kriteria itu tidak lain adalah kemangkusan algoritma.

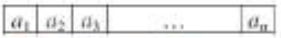
4.3 Kebutuhan Waktu dan Ruang

Kebutuhan waktu suatu algoritma biasanya dihitung dalam satuan detik, mikrodetik, dan sebagainya. Sedangkan, ruang memori yang digunakan dapat dihitung dalam satuan *byte* atau *kilobyte*.

Pada umumnya, orang mengukur kebutuhan waktu suatu algoritma dengan mengeksekusi langsung algoritma tersebut pada suatu komputer, lalu dihitung berapa lama durasi waktu yang dibutuhkan untuk menyelesaikan suatu persoalan dengan n yang berbeda-beda. Keakuratan waktu eksekusi algoritma dapat diperoleh dengan tidak menghitung kebutuhan waktu untuk menampilkan antarmuka program, operasi masukan/keluaran (*read*, *write*) dan sebagainya. Jadi, yang dihitung hanya kebutuhan waktu untuk bagian algoritma yang inti saja.

Perhatikan ilustrasi di bawah ini. Berikut adalah algoritma untuk menghitung rata-rata dari n data bilangan bulat. Asumsikan data masukan sudah dibaca dan disimpan di dalam elemen-elemen larik (tabel). Jadi, kita hanya memperhatikan bagian perhitungan rata-rata saja seperti yang ditunjukkan di bawah ini. Selanjutnya,

jalankan program yang mengandung prosedur ini pada suatu komputer. Berikutnya, hitung selisih waktu antara sebelum pemanggilan prosedur dan sesudah pemanggilan prosedur. Selisih kedua waktu ini adalah kebutuhan waktu aktual untuk menghitung rata-rata n data.



Larik bilangan bulat

```
procedure HitungMerata(input a1, a2, ..., an : integer, output r : real)
{ Menghitung nilai rata-rata dari sekumpulan elemen larik integer a1, a2,
..., an.
  Nilai rata-rata akan disimpan di dalam peubah r.
  Masukan: a1, a2, ..., an
  Keluaran: r (nilai rata-rata)
}
Deklarasi
   k : integer
   jumlah : real

Algoritma
   jumlah←0
   k←1
   while k ≤ n do
     jumlah←jumlah + ak
     k←k+1
   endwhile
   { k > n }
   r ← jumlah/n   { nilai rata-rata }
```

Model perhitungan kebutuhan waktu seperti di atas kurang dapat diterima karena masing-masing komputer memiliki spesifikasi yang berbeda-beda, sehingga menghasilkan waktu yang berbeda juga untuk melaksanakan operasi-operasi dasar. Sedemikian sehingga, tidak memungkinkan untuk menentukan ukuran kebutuhan waktu untuk masing-masing algoritma. Alasan selanjutnya, kebutuhan waktu suatu algoritma bergantung pada *compiler* bahasa pemrograman yang digunakan. *Compiler* yang berbeda akan menerjemahkan program ke dalam kode mesin yang berbeda juga. Sedemikian sehingga, kode mesin yang berbeda akan

menggunakan ruang memori dan memerlukan waktu pelaksanaan program yang berbeda juga.

4.4 Kompleksitas Waktu dan Ruang

Secara teoritis, model abstrak pengukuran waktu/ruang harus independen dari pertimbangan mesin dan *compiler* apapun. Model abstrak seperti ini dapat digunakan untuk membandingkan algoritma yang berbeda. Besaran yang dipakai untuk menerangkan model abstrak pengukuran waktu/ruang ini adalah **kompleksitas algoritma**. Ada dua macam kompleksitas algoritma, yaitu **kompleksitas waktu dan kompleksitas ruang**.

Kompleksitas waktu dinotasikan dengan $T(n)$, yaitu jumlah tahapan komputasi yang dibutuhkan untuk menjalankan algoritma sebagai fungsi dari ukuran masukan n (Tjaru, 2009). Sedangkan, **kompleksitas ruang** dinotasikan dengan $S(n)$, yaitu jumlah memori yang digunakan oleh struktur data yang terdapat di dalam algoritma sebagai fungsi dari ukuran masukan n. Lebih lanjut, dengan menggunakan besaran kompleksitas waktu/ruang algoritma, kita dapat menentukan laju peningkatan waktu (ruang) yang diperlukan algoritma dengan meningkatnya ukuran masukan n.

Pada Bab 8 ini, kita akan fokus untuk membahas kompleksitas waktu, karena kompleksitas ruang dikaitkan dengan struktur data dan kapasitas memori, sementara ukuran memori

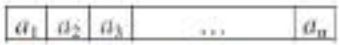
tidak menjadi masalah karena komputer sekarang sudah mempunyai ukuran yang besar, dan meskipun kurang, masih ada memori sekunder. Namun, kompleksitas ruang tetap penting, tetapi kompleksitas waktu akan menjadi fokus utama dalam merancang suatu algoritma.

1. Kompleksitas Waktu

   Langkah-langkah untuk mengukur kompleksitas waktu adalah sebagai berikut:

   a) Menetapkan ukuran masukan, dan
   b) Menghitung banyaknya operasi (operasi penjumlahan (termasuk pengurangan), operasi perbandingan, operasi pembagian, operasi pembacaan, pemanggilan prosedur, dan sebagainya) yang dilakukan oleh algoritma.

Contoh 8.1



Larik bilangan bulat

```
procedure HitungMerata2(input a1, a2, ..., an : integer, output r : real)
{ Menghitung nilai rata-rata dari sekumpulan elemen larik integer a1, a2,
..., an.
  Nilai rata-rata akan disimpan di dalam peubah r.
  Masukan: a1, a2, ..., an
  Keluaran: r (nilai rata-rata)
}
Deklarasi
   k : integer
   jumlah : real

Algoritma
   jumlah←0
   k←1
   while k ≤ n do
     jumlah←jumlah + ak
     k←k+1
   endwhile
   { k > n }
   r ← jumlah/n   { nilai rata-rata }
```

Perhatikan algoritma di atas disusun untuk menghitung rata-rata n data. Tentukan kompleksitas waktu algoritma tersebut.

Penyelesaian :

Identifikasi jenis-jenis operasi yang terdapat di dalam algoritma, lalu hitung masing-masing jumlah operasi. Jika operasi tersebut berada di dalam suatu kalang (*loop*), maka jumlah operasinya bergantung berapa kali kalang tersebut diulang.

(i) Operasi pengisian nilai

Jumlah ← 0,	1 kali
k ← 1,	1 kali
Jumlah ← Jumlah + a_k,	n kali

k ← k + 1, n kali

r ← Jumlah/n, 1 kali

Sehingga, jumlah seluruh operasi pengisian nilai adalah $t_1 = 1 + 1 + n + n + 1 = 3 + 2n$.

(ii) Operasi penjumlahan

Jumlah + a_k, n kali

k + 1, n kali

Sehingga, jumlah seluruh operasi penjumlahan adalah $t_2 = n + n = 2n$.

(iii) Operasi pembagian

Jumlah/n, 1 kali

Sehingga, jumlah seluruh operasi pembagian adalah $t_3 = 1$.

Jadi, kompleksitas waktu algoritma dihitung berdasarkan jumlah operasi aritmetika dan operasi pengisian nilai adalah

$$T(n) = t_1 + t_2 + t_3 = 3 + 2n + 2n + 1 = 4n + 4.$$

Lebih lanjut, hal lain yang harus diperhatikan dalam menentukan kompleksitas waktu adalah parameter yang mencirikan ukuran masukan. Pada algoritma pencarian misalnya, waktu pencarian tidak hanya bergantung pada ukuran larik (n), tetapi juga bergantung pada nilai elemen yang dicari (x). Sebagai contoh, diketahui larik bilangan bulat yang beranggotakan 128 elemen, $a_1, a_2, \dots, a_n$. Asumsikan elemen-elemen larik sudah

berurutan. Jika $a_1 = x$, maka waktu pencariannya 128 kali lebih cepat daripada saat $a_{128} = x$ atau jika x tidak terdapat di dalam larik. Demikian juga, jika $a_{64} = x$, maka waktu pencariannya ½ kali lebih cepat daripada saat $a_{128} = x$. Sedemikian sehingga, kompleksitas waktu dibedakan atas tiga macam:

1) $T_{max}(n)$: kompleksitas waktu untuk kasus terburuk (*worst case*), yaitu kebutuhan waktu maksimum yang diperlukan suatu algoritma sebagai fungsi dari n.
2) $T_{min}(n)$: kompleksitas waktu untuk kasus terbaik (*best case*), yaitu kebutuhan waktu minimum yang diperlukan suatu algoritma sebagai fungsi dari n.
3) $T_{avg}(n)$: kompleksitas waktu untuk kasus rata-rata (*average case*), yaitu kebutuhan waktu rata-rata yang diperlukan algoritma sebagai fungsi dari n. Untuk kasus rata-rata ini, biasanya dibuat asumsi bahwa semua barisan masukan bersifat sama. Misalnya, pada masalah pencarian dimisalkan bahwa data yang dicari mempunyai peluang yang sama untuk terletak di dalam larik.

Contoh 8.2

Diberikan larik bilangan bulat $a_1, a_2, \ldots, a_n$ yang telah terurut dan tidak ada elemen ganda. Tentukan kompleksitas waktu terbaik,

terburuk, dan rata-rata dari algoritma pencarian beruntun (*sequential search*) di bawah ini.

```
procedure PencarianBeruntun(input a1, a2, ..., an : integer, x : integer,
                            output idx : integer)
Deklarasi
  k : integer
  ketemu : boolean    { bernilai true jika x ditemukan atau false jika x
tidak ditemukan }

Algoritma:
  k←1
  ketemu ← false
  while (k ≤ n) and (not ketemu) do
    if ak = x then
      ketemu←true
    else
      k ← k + 1
    endif
  endwhile
  { k > n or ketemu }

  if ketemu then   { x ditemukan }
    idx←k
  else
    idx← 0        { x tidak ditemukan }
  endif
```

Penyelesaian :

Algoritma "Pencarian Beruntun" membandingkan setiap elemen larik dengan x, mulai dari elemen pertama sampai x ditemukan atau sampai elemen terakhir. Jika x ditemukan, maka proses pencarian dihentikan. Kita akan menghitung jumlah operasi perbandingan elemen larik yang terjadi selama pencarian ($\mathrm{a_i = x}$). Operasi perbandingan yang lain, seperti $\mathrm{i \leq n}$ tidak akan dihitung. Operasi perbandingan elemen-elemen larik adalah operasi abstrak yang mendasari algoritma pencarian.

1) *Kasus terbaik*: ini terjadi jika $a_1 = x$.

Operasi perbandingan elemen ($a_i = x$) hanya dilakukan satu kali, maka

$T_{min}(n) = 1$

2) *Kasus terburuk* : jika $a_n = x$ atau x tidak ditemukan. Seluruh elemen larik dibandingkan, maka jumlah perbandingan elemen larik ($a_i = x$) adalah

$T_{max}(n) = n$

3) *Kasus rata-rata* : jika x ditemukan pada posisi ke-*j*, maka operasi perbandingan ($a_i = x$) dilakukan sebanyak *j* kali. Jadi, kebutuhan waktu rata-rata algoritma pencarian beruntun adalah

$$T_{avg}(n) = \frac{(1 + 2 + 3 + \cdots + n)}{n} = \frac{\frac{1}{2}n(n+1)}{n} = \frac{(n+1)}{2}$$

4.5 Beberapa Analisis Kompleksitas Algoritma

Penelitian ini bertujuan untuk mengukur tingkat kompleksitas waktu dan ruang dari algoritma filter IIR Shen-Castan untuk deteksi tepi pada citra digital (Hapsari *et al.*, 2019). Hasil dari penelitian tersebut menjelaskan bahwa algoritma Shen-Castan yang dapat menghasilkan filter yang lebih baik dari filter turunan pertama Gaussian dan LoG. Sedangkan, pada Rahayuningsih (2016) mendiskusikan tentang analisis perbandingan kompleksitas algoritma pengurutan nilai (*sorting*). Pada penelitian ini hanya membahas tentang analisis kompleksitas waktu untuk setiap jenis

algoritma pengurutan nilai. Dijelaskan juga bahwa masing-masing teknik pengurutan memiliki kelebihan dan kekurangan. Pada penelitian ini dijelaskan terkait program tersebut memiliki akses yang cepat atau lambat serta kapasitas memori yang dibutuhkan.

Lebih lanjut, pada Dwiyanto (2017) dibahas analisis kompleksitas algoritma dalam algoritma pengurutan. Kesimpulan dari penelitian ini adalah suatu algoritma tidak dapat dikatakan lebih mangkus (efisien) dari algoritma yang lain. Kemangkusan algoritma didasarkan pada banyak atau sedikitnya data dan penerapannya. Selanjutnya, pemilihan algoritma yang tepat adalah salah satu keahlian yang harus dimiliki oleh seorang programer, sehingga akan lebih baik jika seorang programer memiliki banyak pengetahuan terkait berbagai macam algoritma dan mampu menerapkan suatu algoritma dengan baik dan benar. Sedemikian sehingga, berdasarkan penjelasan di atas dapat disimpulkan bahwa suatu algoritma memiliki kelebihan dan kekurangan masing-masing. Sehingga, keahlian seorang programer dalam hal memilih dan menganalisis algoritma yang mangkus dan efektif sangat penting dalam menjalankan suatu program.

Latihan Soal

1) Perhatikan algoritma untuk mencari elemen terbesar di dalam suatu larik (*array*) yang berukuran n elemen.

Tentukan kompleksitas waktu terbaik, terburuk, dan rata-rata dari algoritma tersebut.

2) Perhatikan algoritma di bawah ini disusun untuk pencarian biner (*bynary search*).

Tentukan kompleksitas waktu algoritma tersebut.

3) Perhatikan algoritma berikut disusun untuk pengurutan pilih (*selection sort*). Tentukan kompleksitas waktu terbaik, terburuk, dan rata-rata dari algoritma tersebut.

```
procedure Urut(input/output a1, a2, ..., an : integer)
Deklarasi
   i, j, imaks, temp : integer

Algoritma
  for i←n downto 2 do   { pass sebanyak n - 1 kali }
    imaks←1
    for j←2 to i do
      if aj > aimaks then
        imaks←j
      endif
    endfor
    { pertukarkan aimaks dengan ai }
    temp←ai
    ai←aimaks
    aimaks←temp

  endfor
```

DAFTAR PUSTAKA

Dwiyanto, R. C. (2017) 'Kompleksitas Algoritma Dalam Algoritma Pengurutan'.

Hapsari, D. *et al.* (2019) 'Analisis Kompleksitas Algoritma Filter IRR Shen-Castan untuk Deteksi Tepi pada Citra Digital', 12(2), pp. 218–228.

Munir, R. (2010) 'Matematika Diskrit'.

Rahayuningsih, P. A. (2016) 'Analisis Perbandingan Kompleksitas Algoritma Pengurutan Nilai (*Sorting*)', 4(2), pp. 64-75.

Tjaru, S. N. B. (2009) 'Kompleksitas Algoritma Pengurutan Selection Sort dan Insertion Sort'.

BAB 5
PENERAPAN MATEMATIKA DISKRIT DALAM STRUKTUR DATA

Oleh Suwito Pomalingo

5.1. Pendahuluan

Matematika Diskrit, sebagai salah satu cabang matematika, memiliki peran yang sangat penting dalam pengembangan dan analisis struktur data. Pada buku ini, kita akan mempelajari tentang apa dan bagaimana Matematika Diskrit diterapkan dalam berbagai jenis struktur data. Mulai dari konsep dasar seperti himpunan dan relasi, fungsi dan pemetaan, hingga logika dan aljabar Boolean, serta teori graf dan pohon. Tujuannya adalah untuk mendalami dan memahami bagaimana prinsip-prinsip ini digunakan untuk meningkatkan efisiensi dan efektivitas dalam manipulasi dan penyimpanan data. Dengan memahami hubungan erat antara Matematika Diskrit dan struktur data, diharapkan kita dapat memanfaatkan pengetahuan ini untuk menciptakan solusi yang lebih baik dan inovatif dalam berbagai aplikasi komputasi.

5.1.1 Definisi dan Sejarah Matematika Diskrit

Matematika Diskrit, juga dikenal sebagai matematika berhitung, adalah cabang dari matematika yang mempelajari objek-objek diskrit. Berbeda dengan matematika kontinu yang berfokus pada objek-objek yang dapat dibagi dan dipecah menjadi bagian

yang sangat kecil atau bahkan tak terbatas, matematika diskrit justru mempelajari objek-objek yang terpisah dan berbeda satu sama lain. Objek-objek ini biasanya dapat dihitung satu per satu, misalnya seperti himpunan, graf, dan pohon. Matematika diskrit memiliki peran yang sangat penting dalam berbagai disiplin ilmu, khususnya dalam ilmu komputer dan teknologi informasi (Rosen, 2011).

Sejarah matematika diskrit tidak dapat dipisahkan dari sejarah matematika itu sendiri. Namun, perkembangan signifikan dalam matematika diskrit dimulai pada abad ke-20, seiring dengan kemajuan dalam bidang ilmu komputer. Banyak konsep dalam matematika diskrit, seperti teori graf dan teori himpunan, menjadi dasar dari algoritma dan struktur data dalam ilmu komputer. Selain itu, konsep logika dan aljabar Boolean dalam matematika diskrit juga menjadi fondasi dari pemrosesan digital dan desain rangkaian digital (Biggs, 2002).

Sebenarnya matematika diskrit sudah ada sejak zaman kuno. Misalnya, penggunaan sistem bilangan bulat dan bilangan prima dalam matematika Mesir kuno dan Yunani kuno. Begitu pula dengan konsep-konsep awal dalam aljabar Boolean dan logika, yang telah ada sejak zaman filsuf Yunani kuno seperti Aristoteles. Ini menunjukkan bahwa walaupun istilah "matematika diskrit" mungkin baru digunakan dalam beberapa dekade terakhir, konsep-konsep yang dipelajari dalam matematika diskrit sebenarnya telah ada sejak ribuan tahun yang lalu (Gallian, 2012).

5.1.2 Definisi dan Peran Struktur Data

Struktur data adalah cara khusus untuk menyimpan dan mengatur data dalam komputer sehingga data tersebut dapat digunakan secara efisien. Struktur data menyediakan suatu cara untuk mengelola sejumlah besar data secara efisien untuk penggunaan seperti database atau sistem indexing internet (Goodrich, Tamassia and Goldwasser, 2014). Beberapa jenis struktur data yang umum digunakan adalah array, linked list, stack, queue, tree, dan graph.

Peran struktur data sangat krusial dalam pemrograman dan pengembangan software. Struktur data memungkinkan kita untuk mengelola dan mengorganisir data, memungkinkan penanganan data secara efisien dalam hal waktu dan memori. Struktur data memberikan fondasi bagi algoritma komputer, dan berperan penting dalam desain sistem komputer dan aplikasi. Misalnya, dalam pencarian informasi, jenis struktur data yang digunakan akan sangat mempengaruhi seberapa cepat informasi dapat ditemukan. Selain itu, dalam pengolahan data skala besar atau big data, struktur data seperti tree dan hash table sangat penting untuk memungkinkan proses yang efisien.

Selain itu, pemilihan struktur data yang tepat juga sangat penting dalam pengembangan algoritma. Misalnya, dalam pengurutan data, struktur data seperti heap dapat digunakan untuk membuat algoritma pengurutan yang efisien. Begitu juga dalam

grafika komputer, struktur data seperti pohon ruang-partisi (space-partitioning trees) digunakan untuk melakukan operasi seperti deteksi tabrakan dan pencahayaan.

5.1.3 Keterkaitan dan Pentingnya Matematika Diskrit dalam Struktur Data

Keterkaitan antara Matematika Diskrit dan Struktur Data sangat erat dan saling melengkapi. Konsep-konsep dalam Matematika Diskrit, seperti teori himpunan, logika, aljabar Boolean, teori graf, dan lainnya, menjadi dasar yang sangat penting dalam pengembangan dan analisis Struktur Data.

Pertama, Struktur Data seringkali dianalisis dan dikembangkan berdasarkan konsep-konsep dalam teori himpunan. Misalnya, array dan linked list pada dasarnya adalah penerapan dari konsep himpunan dalam Struktur Data. Kedua, logika dan aljabar Boolean juga sangat penting dalam Struktur Data. Misalnya, struktur data seperti stack dan queue seringkali dianalisis dan dikembangkan berdasarkan prinsip-prinsip dalam logika dan aljabar Boolean. Selain itu, operasi-operasi dalam Struktur Data, seperti penambahan, penghapusan, dan pencarian data, seringkali dianalisis dengan menggunakan logika dan aljabar Boolean. Ketiga, teori graf dan pohon juga menjadi dasar yang sangat penting dalam Struktur Data. Misalnya, struktur data seperti graph dan tree pada dasarnya adalah penerapan dari teori graf dan pohon dalam Matematika Diskrit. Selain itu, banyak algoritma dalam ilmu komputer, seperti algoritma

pencarian dan pengurutan, dikembangkan berdasarkan konsep-konsep dalam teori graf dan pohon.

Dengan demikian, Matematika Diskrit memainkan peran yang sangat penting dalam Struktur Data. Tanpa pemahaman yang baik tentang Matematika Diskrit, pengembangan dan analisis Struktur Data akan menjadi sangat sulit. Oleh karena itu, penting bagi setiap ilmuwan komputer dan programmer untuk memahami dan menguasai Matematika Diskrit.

5.2 Konsep Dasar Matematika Diskrit dalam Struktur Data

Konsep-konsep dasar Matematika Diskrit memegang peran penting dalam pemahaman dan pengembangan Struktur Data. Pada bagian ini, kita akan mendalami berbagai konsep dasar tersebut dan bagaimana mereka diterapkan dalam berbagai jenis struktur data. Pemahaman yang kuat terhadap konsep-konsep ini akan membantu kita dalam merancang dan menganalisis struktur data serta algoritma yang efisien dan efektif.

Pertama, kita akan membahas tentang himpunan dan relasi. Dalam konteks struktur data, himpunan dapat dianggap sebagai koleksi item atau elemen, sementara relasi mendefinisikan bagaimana elemen-elemen tersebut saling berhubungan satu sama lain. Konsep ini sangat relevan, misalnya dalam struktur data seperti array dan linked list.

Kedua, kita akan mempelajari tentang fungsi dan pemetaan. Dalam struktur data, fungsi dan pemetaan digunakan untuk mengubah dan mengelola data. Misalnya, dalam teknik hashing, fungsi hash digunakan untuk memetakan data ke posisi tertentu dalam struktur data.

Ketiga, kita akan melihat bagaimana logika dan aljabar Boolean diterapkan dalam struktur data. Misalnya, dalam struktur data seperti stack dan queue, operasi penambahan dan penghapusan data dapat dijelaskan dengan menggunakan prinsip-prinsip logika dan aljabar Boolean.

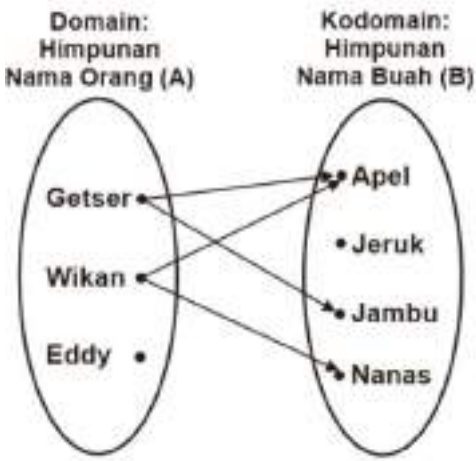
Terakhir, kita akan membahas tentang teori graf dan pohon. Dalam struktur data, teori ini digunakan untuk mendefinisikan dan menganalisis struktur data yang kompleks seperti graph dan tree.

Melalui pemahaman konsep dasar ini, kita dapat memahami bagaimana Matematika Diskrit berkontribusi terhadap desain dan implementasi struktur data. Kita akan melihat bahwa tanpa konsep-konsep ini, struktur data yang kita gunakan sehari-hari mungkin tidak akan efisien atau bahkan mungkin tidak mungkin ada.

5.2.1 Himpunan dan Relasi

Himpunan dan relasi adalah dua konsep dasar dalam Matematika Diskrit yang digunakan secara luas dalam pembuatan dan analisis struktur data. Himpunan, dalam pengertian yang paling dasar, adalah kumpulan objek atau elemen yang berbeda dan tidak berurutan. Misalnya, jika kita memiliki kumpulan bilangan bulat {1,

2, 3}, ini bisa kita sebut sebagai sebuah himpunan. Konsep ini digunakan secara luas dalam struktur data. Sebagai contoh, dalam struktur data seperti array atau linked list, setiap elemen atau data disimpan dalam suatu struktur yang pada dasarnya bisa dianggap sebagai himpunan. Misalnya, array [2, 3, 5, 7, 11] pada dasarnya adalah himpunan bilangan prima.



Gambar 9.1 Himpunan dan Relasi

(Sumber : www.advernesia.com)

Relasi, di sisi lain, adalah konsep yang menggambarkan hubungan antara elemen-elemen dari satu atau lebih himpunan. Sebagai contoh, kita bisa memiliki relasi "lebih besar dari" antara himpunan bilangan bulat. Dalam konteks struktur data, kita bisa melihat bagaimana relasi digunakan dalam struktur data seperti pohon atau graf, di mana relasi mendefinisikan hubungan antara

node atau titik. Misalnya, dalam struktur data tree, node 'A' bisa memiliki relasi "parent dari" dengan node 'B' dan 'C'.

Selain itu, dalam basis data relasional, konsep relasi digunakan untuk mendefinisikan hubungan antara tabel. Misalnya, dalam basis data penjualan, kita bisa memiliki tabel 'Pelanggan' dan 'Pesanan', dan relasi "membuat" antara pelanggan dan pesanan mereka (Date, 2004).

5.2.2 Fungsi, Pemetaan, dan Peranannya dalam Struktur Data

Fungsi dan pemetaan adalah dua konsep lain dari Matematika Diskrit yang sering digunakan dalam struktur data.

Fungsi dalam matematika adalah suatu aturan yang mengaitkan setiap elemen dari satu himpunan (domain) ke tepat satu elemen dari himpunan lainnya (codomain). Fungsi digunakan dalam banyak cara dalam struktur data. Sebagai contoh, dalam struktur data seperti array, kita bisa memahami indeks array sebagai fungsi yang memetakan setiap indeks ke nilai yang sesuai dalam array.

Pemetaan adalah istilah lain untuk fungsi, dan keduanya digunakan secara bergantian. Pemetaan sering digunakan dalam struktur data seperti tabel hash atau hashmap, di mana fungsi hash digunakan untuk memetakan kunci ke nilai yang sesuai. Fungsi hash adalah contoh spesifik dari fungsi atau pemetaan, yang mengambil input (kunci) dan menghasilkan output (indeks dalam array) sehingga data dapat dicari atau disimpan dengan efisien.

Fungsi dan pemetaan juga berperan penting dalam operasi lain pada struktur data, seperti penambahan, penghapusan, dan pencarian data. Misalnya, dalam pohon biner pencarian, fungsi digunakan untuk memetakan setiap kunci ke posisi yang sesuai dalam pohon berdasarkan relasi urutan.

Jadi, konsep fungsi dan pemetaan sangat penting dalam desain dan analisis struktur data. Mereka memberikan cara untuk mengorganisir dan memanipulasi data dengan efisien dan efektif.

5.2.3 Logika Matematika dan Aljabar Boolean

Logika Matematika dan Aljabar Boolean adalah dua cabang penting dari Matematika Diskrit yang digunakan secara luas dalam bidang komputasi dan struktur data.

Logika matematika, yang juga dikenal sebagai logika formal, adalah studi tentang proposisi dan deduksi logis. Dalam konteks struktur data dan algoritma, logika digunakan dalam penulisan dan evaluasi kondisi, serta dalam pengambilan keputusan. Misalnya, dalam algoritma pencarian atau pengurutan, logika digunakan untuk menentukan kondisi yang harus dipenuhi untuk mencari atau mengurutkan data.

Aljabar Boolean, di sisi lain, adalah struktur aljabar yang digunakan untuk analisis dan desain rangkaian logika dalam komputer. Dalam struktur data dan algoritma, Aljabar Boolean digunakan dalam operasi dengan tipe data boolean, seperti AND, OR, dan NOT. Sebagai contoh, dalam struktur data seperti tree atau graf,

operasi boolean dapat digunakan untuk menentukan apakah suatu elemen ada dalam struktur data atau tidak.

Pada tingkat yang lebih tinggi, Aljabar Boolean juga digunakan dalam desain dan optimasi basis data, serta dalam pencarian dan pengambilan data dari basis data. Misalnya, dalam basis data relasional, operasi boolean digunakan dalam penulisan query SQL untuk memilih, mengupdate, atau menghapus data.

Secara keseluruhan, Logika Matematika dan Aljabar Boolean memainkan peran penting dalam desain dan analisis struktur data. Tanpa pemahaman yang baik tentang konsep-konsep ini, akan sulit untuk merancang dan menganalisis struktur data dan algoritma secara efektif dan efisien.

5.2.4 Teori Graf dan Pohon

Teori Graf dan Pohon adalah dua bagian dari Matematika Diskrit yang sangat penting dan banyak digunakan dalam berbagai bidang komputasi dan struktur data.

Teori Graf adalah studi tentang graf, yang adalah kumpulan titik (atau simpul) dan garis (atau tepi) yang menghubungkan titik-titik tersebut. Graf digunakan dalam berbagai struktur data dan algoritma. Misalnya, dalam struktur data graf, kita dapat mewakili objek sebagai simpul dan hubungan antara objek sebagai tepi. Graf juga digunakan dalam algoritma seperti algoritma pencarian jalur terpendek, seperti algoritma Dijkstra atau algoritma A*, yang

digunakan dalam bidang seperti jaringan komputer, optimasi transportasi, dan permainan.

Sebuah graf G dapat diartikulasikan sebagai kombinasi dua set (V, E), yang dinyatakan dalam bentuk G=(V,E). Di sini, V melambangkan kumpulan simpul yang tidak kosong, sementara E mewakili kumpulan sisi yang menghubungkan dua simpul. Sebuah sisi yang hanya terkait dengan satu simpul disebut sebagai loop. Jika ada dua sisi yang berbeda tetapi menghubungkan simpul yang sama, mereka disebut sebagai sisi paralel. Dua simpul dianggap bersebelahan (adjacent) jika ada sisi yang menghubungkan antara satu dengan yang lainnya. Simpul yang tidak terhubung dengan sisi manapun dikenal sebagai simpul terisolasi. Apabila semua sisinya memiliki arah, maka graf tersebut dikenal sebagai graf berarah atau sering disingkat sebagai di-graf. Sebaliknya, jika semua sisinya tidak memiliki arah, graf tersebut disebut sebagai graf tak berarah. Dengan demikian, berdasarkan orientasi sisinya, graf dapat dikategorikan menjadi dua jenis: graf berarah dan graf tak berarah.

Jenis-jenis graf:

1. Graf Berarah (Directed Graf = Digraph)

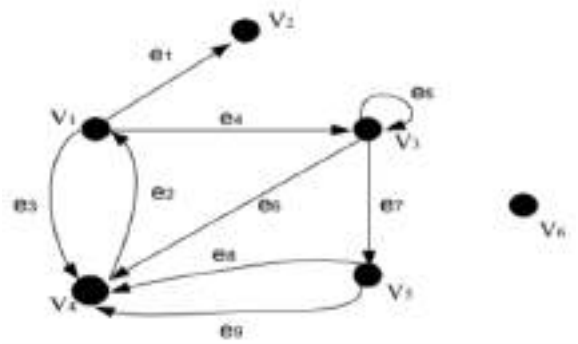
   Dalam graf yang memiliki arah, urutan dan orientasi sisi sangat penting. Sebuah lintasan dalam graf adalah sekuen simpul atau sisi yang digunakan untuk berpindah dari satu simpul ke simpul lainnya. Pada graf dengan arah, simpul tujuan dari satu busur akan menjadi simpul asal untuk busur

yang mengikutinya. Sebuah sirkuit adalah jenis lintasan di mana simpul awal dan akhir identik. Jumlah sisi yang dilewati oleh lintasan tersebut menentukan panjangnya.

Graf berarah, yang disimbolkan sebagai G, terdiri dari kumpulan simpul V(G) {$v1$, $v2$, ...}, kumpulan sisi E(G) {$e1$, $e2$, ...}, serta fungsi yang memetakan setiap sisi dalam E(G) ke sebuah pasangan simpul yang berurutan (vi, vj). Jika ek = (vi, vj) adalah sebuah sisi dalam G, maka vi dikenal sebagai simpul asal dari ek, dan vj adalah simpul tujuan dari ek. Orientasi sisi bergerak dari vi ke vj.

Jumlah sisi yang berangkat dari simpul vi dikenal sebagai derajat keluar dari simpul tersebut, yang dinotasikan dengan $d+(vi)$. Sementara itu, jumlah sisi yang mengarah ke simpul vi disebut sebagai derajat masuk dari simpul tersebut, yang dinotasikan dengan $d-(vi)$.

Simpul yang terisolasi adalah simpul dalam G yang memiliki derajat keluar dan masuk sama dengan nol. Simpul yang tergantung adalah simpul dalam G di mana derajat masuk dan keluarnya sama dengan satu. Dua sisi berarah disebut paralel jika mereka memiliki simpul asal dan tujuan yang sama.



Gambar 9.2 Contoh Graf Berarah

(Sumber : Munir, 2005)

2. Graf Tak Berarah (Undirected Graph)

   Graf tak berarah adalah jenis graf di mana orientasi atau arah sisi tidak diperhatikan. Dalam notasi matematis, graf tak berarah biasanya didefinisikan sebagai pasangan $G=(V,E)$, di mana V adalah himpunan simpul dan E adalah himpunan sisi. Dalam konteks ini, setiap sisi adalah pasangan tidak berurutan dari simpul, yaitu $(u,v)=(v,u)$. Ini berarti bahwa sisi antara dua simpul u dan v identik, tidak peduli dari mana ke mana arahnya.

   Graf tak berarah memiliki berbagai jenis dan properti. Misalnya, graf yang semua simpulnya terhubung disebut graf terhubung, sedangkan yang tidak disebut graf tidak terhubung. Selain itu, ada konsep lintasan, yaitu urutan simpul yang dilewati melalui sisi-sisi graf. Dalam graf tak

berarah, lintasan dari simpul *A* ke *B* sama dengan lintasan dari *B* ke *A*.

Loop adalah sisi yang menghubungkan simpul ke dirinya sendiri, dan dalam graf tak berarah, loop biasanya dihindari kecuali dalam kasus tertentu. Sisi paralel adalah dua atau lebih sisi yang menghubungkan simpul yang sama. Dalam graf tak berarah, sisi paralel juga biasanya dihindari untuk menyederhanakan graf.

Graf tak berarah banyak digunakan dalam berbagai aplikasi praktis. Misalnya, dalam jaringan sosial, graf tak berarah bisa digunakan untuk merepresentasikan hubungan mutual antara individu. Dalam biologi, graf ini bisa digunakan untuk menunjukkan hubungan antara berbagai spesies atau gen. Dalam ilmu komputer, graf tak berarah sering digunakan dalam algoritma seperti pencarian jalur terpendek, penyebaran informasi, dan lain-lain.

Pohon adalah jenis khusus dari graf yang tidak memiliki siklus (yaitu, Anda tidak dapat memulai dari suatu simpul dan mengikuti serangkaian tepi untuk kembali ke simpul yang sama). Pohon sering digunakan dalam struktur data dan algoritma. Misalnya, pohon biner digunakan dalam struktur data seperti pohon pencarian biner dan heaps, yang memungkinkan pencarian, penambahan, dan penghapusan data secara efisien. Pohon juga digunakan dalam

algoritma seperti algoritma penelusuran pohon, seperti penelusuran pre-order, in-order, dan post-order.

Selain itu, pohon dan graf juga digunakan dalam bidang lain seperti ilmu komputer teoritis (misalnya, dalam penentuan jadwal optimal), jaringan komputer (misalnya, dalam penentuan rute terpendek antara dua titik dalam jaringan), dan bahkan dalam ilmu biologi (misalnya, dalam konstruksi pohon filogenetik dalam biologi evolusioner).

5.3 Penerapan Matematika Diskrit dalam Struktur Data

Penerapan Matematika Diskrit dalam struktur data tidak hanya memungkinkan kita untuk memahami karakteristik dan batasan dari berbagai jenis struktur data, tetapi juga membantu dalam pengembangan algoritma untuk manipulasi data tersebut. Dengan memahami sifat matematis dari struktur data, kita bisa merancang algoritma yang lebih efisien, yang pada gilirannya memungkinkan performa yang lebih baik dan penggunaan sumber daya yang lebih optimal.

5.3.1 Himpunan dan Aplikasinya dalam Struktur Data (contoh: Array, Linked List)

Himpunan dalam Matematika Diskrit adalah kumpulan objek atau elemen yang tidak berurutan dan unik. Konsep ini memiliki aplikasi langsung dalam struktur data komputer, terutama dalam desain dan implementasi array dan linked list. Array adalah struktur

data yang memungkinkan penyimpanan elemen-elemen dalam indeks yang berurutan, mirip dengan konsep himpunan dalam matematika tetapi dengan elemen yang berurutan. Di sisi lain, linked list adalah struktur data yang terdiri dari elemen-elemen yang disimpan dalam node, di mana setiap node memiliki referensi ke node berikutnya, membentuk sebuah "rantai". Meskipun elemen dalam linked list tidak harus unik atau berurutan, struktur ini bisa dianggap sebagai representasi dari himpunan dalam kasus tertentu, terutama ketika elemen-elemen di dalamnya unik.

Contoh sederhana dari aplikasi himpunan dalam array adalah operasi seperti pencarian elemen, penghapusan elemen, dan penambahan elemen. Misalnya, kita bisa memiliki sebuah array yang berisi angka-angka [1, 2, 3, 4, 5], dan kita ingin mengetahui apakah angka 3 ada dalam array tersebut. Ini mirip dengan operasi "keanggotaan" dalam teori himpunan. Dalam konteks linked list, kita bisa memiliki sebuah linked list yang berisi elemen-elemen seperti "A" -> "B" -> "C", dan kita ingin menambahkan elemen "D" ke dalam list. Ini bisa dianggap sebagai operasi "penyatuan" dalam teori himpunan, di mana kita menyatukan himpunan {A, B, C} dengan {D} untuk membentuk himpunan baru {A, B, C, D}.

Dalam keadaan lebih kompleks, konsep himpunan juga digunakan dalam operasi seperti "irisan" dan "gabungan" antar array atau linked list. Misalnya, jika kita memiliki dua array [1, 2, 3] dan [3, 4, 5], operasi irisan akan menghasilkan array baru [3],

sedangkan operasi gabungan akan menghasilkan array [1, 2, 3, 4, 5]. Dalam konteks linked list, operasi ini bisa sedikit lebih kompleks karena perlu menangani referensi antar node, tetapi prinsip dasarnya tetap sama. Dengan memahami konsep himpunan dari Matematika Diskrit, kita bisa merancang algoritma yang lebih efisien dan efektif untuk manipulasi struktur data ini.

Berikut adalah contoh implementasi sederhana dalam bahasa pemrograman Python yang menunjukkan bagaimana konsep himpunan dari Matematika Diskrit bisa diterapkan dalam struktur data array dan linked list.

Array dan Operasi Himpunan

```
# Fungsi untuk mencari irisan antara dua array
def array_intersection(arr1, arr2):
    return [x for x in arr1 if x in arr2]

# Fungsi untuk mencari gabungan antara dua array
def array_union(arr1, arr2):
    return list(set(arr1 + arr2))

arr1 = [1, 2, 3]
arr2 = [3, 4, 5]

print("Irisan:", array_intersection(arr1, arr2))  # Output: [3]
print("Gabungan:", array_union(arr1, arr2))  # Output: [1, 2, 3,
4, 5]
```

```
class Node:
    def __init__(self, data):
        self.data = data
        self.next = None

class LinkedList:
    def __init__(self):
        self.head = None

    # Menambahkan elemen ke linked list
    def append(self, data):
        new_node = Node(data)
        if self.head is None:
            self.head = new_node
            return
        last_node = self.head
        while last_node.next:
            last_node = last_node.next
        last_node.next = new_node

    # Menampilkan elemen-elemen dalam linked list
    def print_list(self):
        cur_node = self.head
        while cur_node:
            print(cur_node.data, end=" -> ")
            cur_node = cur_node.next
        print("None")

# Membuat linked list dari himpunan {A, B, C}
llist = LinkedList()
llist.append("A")
llist.append("B")
```

```
llist.append("C")

# Menampilkan linked list
llist.print_list()  # Output: A -> B -> C -> None

# Menambahkan elemen "D" ke linked list (Operasi Penyatuan)
llist.append("D")
llist.print_list()  # Output: A -> B -> C -> D -> None
```

Dalam contoh array, kita menggunakan fungsi bawaan Python untuk set untuk melakukan operasi gabungan, yang pada dasarnya adalah representasi dari himpunan. Untuk linked list, kita membuat sebuah kelas Node untuk merepresentasikan elemen dan sebuah kelas LinkedList untuk merepresentasikan himpunan itu sendiri. Operasi penambahan elemen ke linked list mirip dengan operasi "penyatuan" dalam teori himpunan.

5.3.2 Logika Matematika dan Aljabar Boolean dalam Struktur Data (contoh: Stack, Queue)

Logika Matematika dan Aljabar Boolean memainkan peran penting dalam desain dan operasi dari beberapa struktur data, terutama stack dan queue. Dalam konteks stack, yang merupakan struktur data berbasis prinsip Last-In-First-Out (LIFO), operasi seperti "push" dan "pop" seringkali diatur oleh kondisi logis. Misalnya, sebelum melakukan operasi "pop", kita perlu memeriksa apakah stack sudah kosong atau belum. Demikian pula, dalam queue yang mengikuti prinsip First-In-First-Out (FIFO), operasi "enqueue" dan "dequeue" juga diatur oleh kondisi logis, seperti apakah queue sudah penuh atau masih memiliki ruang.

Contoh penerapan logika dalam stack adalah penggunaan algoritma untuk memeriksa kevalidan dari ekspresi matematika yang menggunakan tanda kurung. Dalam kasus ini, kita bisa menggunakan stack untuk menyimpan tanda kurung pembuka dan memeriksa dengan tanda kurung penutup menggunakan logika Boolean. Untuk queue, contoh penerapannya bisa dilihat dalam algoritma simulasi antrian, di mana logika digunakan untuk menentukan apakah elemen bisa ditambahkan ke antrian atau dihapus dari antrian berdasarkan kondisi tertentu.

```
# Implementasi Stack dengan Logika
class Stack:
    def __init__(self):
        self.items = []

    def is_empty(self):
        return len(self.items) == 0

    def push(self, item):
        self.items.append(item)

    def pop(self):
        if not self.is_empty():
            return self.items.pop()

# Memeriksa kevalidan ekspresi dengan tanda kurung
def check_expression(expression):
    stack = Stack()
    for char in expression:
        if char == "(":
            stack.push(char)
```

```
        elif char == ")":
            if stack.is_empty():
                return False
            stack.pop()
    return stack.is_empty()

print(check_expression("(1+2)"))  # Output: True
print(check_expression("(1+2"))   # Output: False

# Implementasi Queue dengan Logika
class Queue:
    def __init__(self, size):
        self.items = []
        self.size = size

    def is_full(self):
        return len(self.items) == self.size

    def enqueue(self, item):
        if not self.is_full():
            self.items.append(item)

    def dequeue(self):
        if len(self.items) > 0:
            return self.items.pop(0)

# Simulasi Antrian
queue = Queue(3)
queue.enqueue(1)
queue.enqueue(2)
print(queue.is_full())  # Output: False
queue.enqueue(3)
print(queue.is_full())  # Output: True
```

Dalam contoh di atas, metode is_empty dan is_full menggunakan logika untuk menentukan apakah operasi pop pada stack atau enqueue pada queue bisa dilakukan. Ini adalah contoh nyata dari bagaimana logika matematika dan aljabar Boolean digunakan dalam struktur data.

5.3.3 Teori Graf dan Pohon dalam Struktur Data (contoh: Graph, Tree)

Teori Graf dan Pohon adalah salah satu aspek paling penting dari Matematika Diskrit yang memiliki aplikasi langsung dalam struktur data. Graf adalah struktur data yang terdiri dari simpul (nodes) dan sisi (edges) yang menghubungkan simpul-simpul tersebut. Graf digunakan dalam berbagai aplikasi, mulai dari representasi jaringan sosial, rute antar kota, hingga algoritma pathfinding dalam game. Pohon adalah jenis graf yang spesial, di mana setiap simpul memiliki tepat satu "parent" kecuali simpul akar, dan tidak ada siklus. Pohon sering digunakan dalam struktur data seperti pohon pencarian biner, pohon AVL, dan pohon merah-hitam, yang memungkinkan operasi pencarian, penyisipan, dan penghapusan data secara efisien.

Contoh penerapan teori graf dalam struktur data adalah dalam algoritma shortest path seperti Dijkstra atau A*. Dalam kasus ini, simpul merepresentasikan titik-titik dalam jaringan atau grid, dan sisi merepresentasikan jarak atau biaya antar titik. Untuk pohon, contoh penerapannya adalah dalam pohon sintaksis abstrak yang digunakan dalam kompilasi kode program, di mana setiap simpul

merepresentasikan operasi atau variabel, dan struktur pohon membantu dalam evaluasi ekspresi atau pernyataan.

```
# Implementasi sederhana dari Graf menggunakan adjacency list
class Graph:
    def __init__(self):
        self.graph = {}

    def add_edge(self, u, v):
        if u not in self.graph:
            self.graph[u] = []
        self.graph[u].append(v)

    def show_edges(self):
        for node in self.graph:
            for neighbour in self.graph[node]:
                print(f"{node} -> {neighbour}")

# Membuat graf
g = Graph()
g.add_edge('A', 'B')
g.add_edge('A', 'C')
g.add_edge('B', 'D')
g.add_edge('C', 'D')
g.add_edge('D', 'A')

# Menampilkan sisi-sisi dalam graf
g.show_edges()  # Output: A -> B, A -> C, B -> D, C -> D, D -> A

# Implementasi sederhana dari Pohon menggunakan class Node
class Node:
    def __init__(self, key):
        self.left = None
        self.right = None
```

```
        self.val = key

# Menambahkan simpul ke pohon pencarian biner
def insert(root, key):
    if root is None:
        return Node(key)
    else:
        if root.val < key:
            root.right = insert(root.right, key)
        else:
            root.left = insert(root.left, key)
    return root

# Menampilkan elemen-elemen dalam pohon (inorder traversal)
def inorder(root):
    if root:
        inorder(root.left)
        print(root.val, end=" ")
        inorder(root.right)

# Membuat pohon
r = Node(50)
r = insert(r, 30)
r = insert(r, 70)
r = insert(r, 20)
r = insert(r, 40)
r = insert(r, 60)
r = insert(r, 80)

# Menampilkan elemen-elemen dalam pohon
inorder(r)  # Output: 20 30 40 50 60 70 80
```

Dalam contoh di atas, kelas Graph menggunakan adjacency list untuk merepresentasikan graf, dan metode add_edge digunakan

untuk menambahkan sisi antar simpul. Untuk pohon, kita menggunakan kelas Node untuk merepresentasikan simpul dan fungsi insert untuk menambahkan simpul ke pohon pencarian biner. Fungsi inorder digunakan untuk menampilkan elemen-elemen dalam pohon.

5.3.4 Fungsi dan Pemetaan dalam Struktur Data (contoh: Hashing, Sorting)

Fungsi dan pemetaan adalah konsep matematika yang sering digunakan dalam struktur data, terutama dalam konteks hashing dan sorting. Dalam hashing, sebuah fungsi hash digunakan untuk memetakan data ke sebuah indeks dalam sebuah array atau tabel hash. Fungsi ini dirancang untuk meminimalkan jumlah bentrokan, di mana dua atau lebih item berbeda dipetakan ke indeks yang sama. Dengan demikian, hashing memungkinkan akses data yang sangat cepat, biasanya dalam waktu konstan $O(1)$. Di sisi lain, sorting adalah proses pengaturan elemen data dalam urutan tertentu (biasanya menaik atau menurun), dan berbagai algoritma sorting seringkali menggunakan fungsi pemetaan atau perbandingan untuk menentukan posisi relatif antar elemen.

Contoh penerapan fungsi dalam hashing adalah dalam implementasi tabel hash atau kamus dalam bahasa pemrograman. Fungsi hash mengambil kunci sebagai input dan menghasilkan indeks di mana nilai yang sesuai dengan kunci tersebut disimpan. Dalam sorting, fungsi pemetaan atau perbandingan bisa digunakan dalam algoritma seperti quicksort atau mergesort untuk

membandingkan elemen dan menentukan bagaimana mereka harus diurutkan.

```
# Implementasi Hash Table sederhana
class HashTable:
    def __init__(self, size):
        self.size = size
        self.table = [None] * size

    def hash_function(self, key):
        return key % self.size

    def insert(self, key, value):
        index = self.hash_function(key)
        self.table[index] = value

    def search(self, key):
        index = self.hash_function(key)
        return self.table[index]

# Membuat Hash Table
hash_table = HashTable(10)
hash_table.insert(1, 'Apple')
hash_table.insert(11, 'Banana')
print(hash_table.search(1))  # Output: Apple
print(hash_table.search(11))  # Output: Banana

# Implementasi Sorting menggunakan Quick Sort
def quick_sort(arr):
    if len(arr) <= 1:
        return arr
    pivot = arr[len(arr) // 2]
    left = [x for x in arr if x < pivot]
    middle = [x for x in arr if x == pivot]
```

```
    right = [x for x in arr if x > pivot]
    return quick_sort(left) + middle + quick_sort(right)

# Menggunakan Quick Sort
arr = [3, 6, 2, 7, 1, 9]
sorted_arr = quick_sort(arr)
print(sorted_arr)  # Output: [1, 2, 3, 6, 7, 9]
```

Dalam contoh di atas, kelas HashTable menggunakan fungsi hash sederhana berdasarkan modulus untuk memetakan kunci ke indeks dalam tabel. Metode insert dan search digunakan untuk menyisipkan dan mencari elemen berdasarkan kunci. Untuk sorting, kita menggunakan algoritma quick sort, yang memilih sebuah "pivot" dan mempartisi array menjadi tiga bagian: elemen yang lebih kecil dari pivot, elemen yang sama dengan pivot, dan elemen yang lebih besar dari pivot. Fungsi quick_sort kemudian dipanggil secara rekursif pada partisi ini. Kedua contoh ini menunjukkan bagaimana fungsi dan pemetaan digunakan dalam struktur data untuk memfasilitasi operasi seperti penyimpanan dan pengurutan data.

5.4 Studi Kasus dan Analisis

5.4.1 Studi Kasus Penggunaan Matematika Diskrit dalam Struktur Data di Dunia Nyata

1. Sistem Penyimpanan Data

   Dalam dunia penyimpanan data, Matematika Diskrit sering digunakan untuk memaksimalkan efisiensi dan kecepatan akses data. Salah satu contoh paling nyata adalah penggunaan tabel hash. Dalam tabel hash, sebuah fungsi hash

digunakan untuk memetakan kunci unik ke indeks dalam sebuah array, memungkinkan akses data yang cepat dan efisien. Fungsi hash ini dirancang untuk meminimalkan bentrokan, di mana dua atau lebih kunci berbeda dipetakan ke indeks yang sama, sehingga memaksimalkan kecepatan akses data.

```
class HashTable:
    def __init__(self, size):
        self.size = size
        self.table = [None] * size

    def hash_function(self, key):
        return hash(key) % self.size

    def insert(self, key, value):
        index = self.hash_function(key)
        self.table[index] = value

    def search(self, key):
        index = self.hash_function(key)
        return self.table[index]
```

Dalam contoh kode di atas, metode hash_function menggunakan fungsi hash bawaan Python dan operasi modulus untuk memetakan kunci ke indeks dalam tabel. Ini adalah contoh sederhana, tetapi dalam aplikasi nyata, fungsi hash yang lebih kompleks sering digunakan untuk meminimalkan bentrokan dan memaksimalkan efisiensi. Dengan demikian, Matematika Diskrit memungkinkan kita

untuk merancang sistem penyimpanan data yang lebih efisien dan responsif.

Secara keseluruhan, penggunaan Matematika Diskrit dalam sistem penyimpanan data seperti tabel hash membantu dalam meningkatkan kecepatan dan efisiensi akses data. Ini sangat penting dalam aplikasi yang memerlukan akses data yang cepat dan efisien, seperti basis data, mesin pencarian, dan banyak lagi.

2. Pengembangan Algoritma

   Algoritma adalah jantung dari banyak aplikasi komputasi, dan Matematika Diskrit sering digunakan dalam desain dan analisis algoritma. Salah satu contoh adalah algoritma Dijkstra untuk menemukan jalur terpendek dalam graf berbobot. Algoritma ini menggunakan konsep dari teori graf dan struktur data seperti heap untuk menemukan jalur terpendek dari satu simpul ke semua simpul lainnya dalam graf.

```
import heapq

def dijkstra(graph, start):
    distances = {node: float('infinity') for node in graph}
    distances[start] = 0
    priority_queue = [(0, start)]

    while priority_queue:
        current_distance, current_node =
heapq.heappop(priority_queue)
```

```
        if current_distance > distances[current_node]:
            continue

        for neighbor, weight in
graph[current_node].items():
            distance = current_distance + weight

            if distance < distances[neighbor]:
                distances[neighbor] = distance
                heapq.heappush(priority_queue, (distance,
neighbor))

    return distances
```

Dalam contoh kode di atas, kita menggunakan struktur data heap untuk menyimpan simpul yang akan dieksplorasi, diurutkan berdasarkan jarak dari simpul asal. Ini adalah contoh bagaimana teori graf dan prioritas queue digabungkan untuk menciptakan algoritma yang efisien. Algoritma ini memiliki banyak aplikasi, mulai dari sistem navigasi hingga jaringan komputer.

Penggunaan Matematika Diskrit dalam pengembangan algoritma tidak hanya membantu dalam menciptakan solusi yang efisien tetapi juga dalam analisis formal dari algoritma tersebut. Misalnya, kita bisa menggunakan konsep dari Matematika Diskrit untuk menentukan kompleksitas waktu dan ruang dari algoritma, yang sangat penting dalam mengevaluasi efisiensi algoritma.

Secara keseluruhan, Matematika Diskrit memainkan peran kunci dalam pengembangan algoritma yang efisien dan efektif. Dari algoritma sorting hingga algoritma graf, prinsip-prinsip Matematika Diskrit digunakan untuk merancang, menganalisis, dan mengoptimalkan algoritma untuk berbagai jenis masalah komputasi.

3. Komputasi Paralel

   Komputasi paralel adalah salah satu area di mana Matematika Diskrit menunjukkan kekuatannya. Dalam komputasi paralel, masalah besar dipecah menjadi sub-problem yang lebih kecil yang bisa diselesaikan secara bersamaan. Teori graf digunakan untuk memodelkan dan menganalisis komunikasi antar prosesor. Misalnya, dalam algoritma paralel untuk pengurutan atau pencarian, model komunikasi antar prosesor bisa diwakili sebagai graf, di mana setiap simpul mewakili sebuah prosesor dan setiap sisi mewakili komunikasi antar prosesor.

```
from multiprocessing import Pool

def square(x):
    return x * x

if __name__ == '__main__':
    data = [1, 2, 3, 4, 5]
    with Pool(5) as p:
        print(p.map(square, data))
```

Dalam contoh kode di atas, kita menggunakan modul multiprocessing dari Python untuk melakukan komputasi paralel. Fungsi square diterapkan ke setiap elemen dalam array data secara paralel menggunakan 5 prosesor. Ini adalah contoh sederhana tetapi efektif dari bagaimana komputasi paralel bisa mempercepat proses komputasi.

Pemahaman tentang Matematika Diskrit memungkinkan kita untuk merancang algoritma paralel yang lebih efisien, memaksimalkan penggunaan sumber daya komputasi yang tersedia. Ini sangat penting dalam aplikasi yang memerlukan komputasi intensif, seperti analisis data besar, simulasi ilmiah, dan lain-lain.

Secara keseluruhan, Matematika Diskrit memberikan alat yang diperlukan untuk merancang dan menganalisis algoritma dalam komputasi paralel. Dari memodelkan komunikasi antar prosesor sebagai graf hingga menggunakan kombinatorika untuk merancang algoritma paralel, Matematika Diskrit membantu dalam memaksimalkan efisiensi dan efektivitas dari komputasi paralel.

5.4.2 Analisis Pengaruh Matematika Diskrit Dalam Efisiensi Dan Efektivitas Struktur Data

Matematika Diskrit memainkan peran kunci dalam meningkatkan efisiensi dan efektivitas dari berbagai jenis struktur

data. Misalnya, penggunaan fungsi hash yang baik dalam tabel hash dapat secara signifikan mengurangi waktu yang diperlukan untuk operasi pencarian, penyisipan, dan penghapusan, seringkali mencapai kompleksitas waktu $O(1)$. Dalam konteks teori graf, algoritma seperti Dijkstra atau A* memungkinkan kita untuk menemukan jalur terpendek dalam graf dengan efisiensi yang relatif tinggi, yang sangat berguna dalam aplikasi seperti penentuan rute dalam sistem GPS atau optimasi jaringan dalam keamanan siber.

Selain itu, pemahaman tentang Matematika Diskrit juga memungkinkan kita untuk melakukan analisis formal terhadap algoritma dan struktur data, seperti menentukan kompleksitas waktu dan ruang. Ini sangat penting dalam penelitian dan pengembangan untuk memastikan bahwa solusi yang dihasilkan tidak hanya benar, tetapi juga optimal dari segi sumber daya yang digunakan.

Pentingnya Matematika Diskrit dalam struktur data dan algoritma juga tercermin dalam berbagai metrik kinerja, seperti kecepatan, skalabilitas, dan kehandalan. Dalam banyak kasus, penerapan prinsip Matematika Diskrit bisa menghasilkan peningkatan signifikan dalam metrik-metrik ini, membuat solusi lebih layak untuk aplikasi dunia nyata yang memerlukan efisiensi dan keandalan tinggi.

Secara keseluruhan, Matematika Diskrit memberikan dasar teoritis yang memungkinkan kita untuk merancang, menganalisis,

dan mengoptimalkan struktur data dan algoritma. Dalam dunia yang semakin bergantung pada data dan komputasi, peran Matematika Diskrit menjadi semakin penting dalam membentuk dasar dari banyak solusi teknologi yang kita gunakan setiap hari.

5.5 Kritik, Tantangan, dan Prospek

5.5.1 Keterbatasan Matematika Diskrit dalam Struktur Data

Meskipun Matematika Diskrit telah membantu dalam pengembangan dan optimasi struktur data, ada beberapa keterbatasan yang perlu diakui. Pertama, fungsi hash yang digunakan dalam tabel hash, meskipun efisien, bisa mengalami bentrokan yang mempengaruhi kinerja. Selain itu, algoritma berbasis graf seperti Dijkstra atau A* bisa menjadi sangat kompleks dan memakan waktu jika graf yang dihadapi memiliki banyak simpul dan sisi, atau jika bobot pada sisi-sisi tersebut berubah dinamis.

Kedua, Matematika Diskrit seringkali memberikan solusi yang ideal dalam konteks teoritis tetapi mungkin tidak selalu praktis dalam implementasi dunia nyata. Misalnya, algoritma yang memiliki kompleksitas waktu polinomial mungkin tidak efisien untuk data set yang sangat besar, meskipun secara teoritis dianggap "cepat". Ini menunjukkan adanya kesenjangan antara teori dan praktek yang kadang-kadang sulit dijembatani.

Ketiga, Matematika Diskrit biasanya tidak mempertimbangkan faktor-faktor eksternal seperti kegagalan perangkat keras, latensi

jaringan, atau masalah keamanan yang bisa mempengaruhi kinerja struktur data dan algoritma. Ini berarti bahwa solusi yang dihasilkan mungkin memerlukan penyesuaian atau modifikasi untuk mengakomodasi variabel-variabel ini.

Keempat, Matematika Diskrit seringkali bersifat statis dan tidak selalu dapat menangani dinamika dari data yang berubah-ubah. Misalnya, dalam kasus struktur data adaptif yang memerlukan reorganisasi dinamis, pendekatan tradisional Matematika Diskrit mungkin tidak selalu efisien.

5.5.2 Tantangan dan Solusi dalam Mengintegrasikan Matematika Diskrit dan Struktur Data

Salah satu tantangan utama dalam mengintegrasikan Matematika Diskrit dan struktur data adalah bagaimana memilih model atau algoritma yang paling sesuai untuk suatu masalah spesifik. Tidak semua algoritma atau struktur data cocok untuk setiap jenis masalah, dan memilih pendekatan yang salah bisa berdampak negatif pada efisiensi dan efektivitas solusi. Oleh karena itu, pemilihan model yang tepat, yang seringkali memerlukan keahlian interdisipliner, menjadi sangat penting.

Selain itu, implementasi algoritma dan struktur data seringkali memerlukan kompromi antara berbagai faktor, seperti kecepatan, memori, dan keandalan. Misalnya, struktur data yang sangat cepat mungkin memerlukan lebih banyak memori, yang bisa menjadi masalah dalam sistem dengan sumber daya terbatas. Solusi untuk

tantangan ini bisa berupa desain algoritma yang lebih efisien atau penggunaan teknik optimasi lainnya.

Tantangan lainnya adalah kebutuhan untuk validasi empiris dari model atau algoritma. Meskipun analisis teoritis memberikan wawasan penting tentang kinerja suatu algoritma, pengujian empiris dalam kondisi dunia nyata adalah langkah penting untuk memastikan bahwa algoritma tersebut memang efektif dan efisien dalam prakteknya.

Untuk mengatasi tantangan-tantangan ini, salah satu solusinya adalah melalui pendekatan kolaboratif antara matematikawan, ilmuwan komputer, dan insinyur. Kolaborasi ini memungkinkan tim untuk memanfaatkan keahlian dari berbagai disiplin ilmu, dari teori hingga implementasi, sehingga menghasilkan solusi yang lebih holistik dan efektif.

5.5.3 Potensi dan Arah Inovasi untuk Masa Depan

Matematika Diskrit memiliki potensi besar untuk terus mempengaruhi dan membentuk masa depan struktur data dan algoritma. Salah satu area yang menjanjikan adalah pengembangan algoritma adaptif yang bisa "belajar" dari data dan dinamika sistem untuk meningkatkan kinerja. Ini bisa sangat berguna dalam aplikasi seperti mesin pencarian, analisis data besar, dan kecerdasan buatan.

Selain itu, dengan kemajuan dalam komputasi kuantum, ada potensi untuk mengembangkan algoritma dan struktur data yang bisa memanfaatkan keuntungan dari komputasi kuantum. Ini bisa

membawa peningkatan signifikan dalam kecepatan dan efisiensi, membuka peluang untuk menyelesaikan masalah yang saat ini dianggap terlalu kompleks untuk dipecahkan.

Penggunaan Matematika Diskrit dalam analisis dan verifikasi keamanan juga menjadi semakin penting, terutama dalam konteks keamanan siber. Dengan meningkatnya ancaman keamanan, ada kebutuhan untuk struktur data dan algoritma yang tidak hanya cepat dan efisien tetapi juga aman dari berbagai jenis serangan.

Secara keseluruhan, Matematika Diskrit akan terus menjadi komponen penting dalam desain dan optimasi struktur data dan algoritma. Dengan tantangan dan kebutuhan yang terus berkembang, ada banyak peluang untuk inovasi dan penelitian lebih lanjut dalam bidang ini. Dari algoritma adaptif hingga komputasi kuantum, potensi untuk inovasi adalah tak terbatas, dan Matematika Diskrit akan terus memainkan peran kunci dalam membentuk teknologi masa depan.

5.6 Kesimpulan

5.6.1 Ringkasan Materi dan Penerapan Matematika Diskrit dalam Struktur Data

Matematika Diskrit, sebagai cabang matematika yang mempelajari struktur diskrit seperti himpunan, graf, dan fungsi, memiliki peran yang sangat penting dalam pengembangan dan optimasi struktur data. Dari tabel hash yang memanfaatkan fungsi

hash untuk penyimpanan data yang efisien, hingga algoritma graf seperti Dijkstra yang digunakan untuk menemukan jalur terpendek, Matematika Diskrit memberikan dasar teoritis yang kuat. Selain itu, konsep-konsep ini juga digunakan dalam komputasi paralel, di mana teori graf dan kombinatorika memungkinkan desain algoritma yang lebih efisien dan efektif.

5.6.2 Implikasi dan Kontribusi Matematika Diskrit terhadap Struktur Data

Implikasi dari penerapan Matematika Diskrit dalam struktur data sangat luas. Pertama, ia memungkinkan peningkatan signifikan dalam efisiensi dan efektivitas operasi data, seringkali mengurangi kompleksitas waktu dari operasi tersebut. Ini sangat penting dalam aplikasi yang memerlukan kecepatan dan efisiensi, seperti mesin pencarian, basis data, dan analisis data besar. Kedua, Matematika Diskrit juga memungkinkan analisis formal dan verifikasi dari struktur data dan algoritma, yang penting dalam penelitian dan pengembangan teknologi baru. Ini tidak hanya mempengaruhi kecepatan dan efisiensi, tetapi juga keamanan dan keandalan dari sistem yang diimplementasikan.

5.6.3 Saran dan Prospek untuk Studi Selanjutnya

Meskipun Matematika Diskrit telah memberikan banyak kontribusi pada struktur data, masih ada banyak ruang untuk penelitian dan pengembangan lebih lanjut. Salah satu area yang menjanjikan adalah pengembangan algoritma dan struktur data

yang adaptif, yang bisa "belajar" dari data dan dinamika sistem untuk meningkatkan kinerja. Selain itu, dengan kemajuan dalam komputasi kuantum, ada potensi untuk mengembangkan algoritma dan struktur data yang memanfaatkan keuntungan dari komputasi kuantum. Akhirnya, integrasi antara Matematika Diskrit dan bidang lain seperti keamanan siber, analisis data besar, dan kecerdasan buatan akan menjadi langkah penting dalam mencapai solusi yang lebih holistik dan efektif.

Dengan demikian, Matematika Diskrit akan terus menjadi area penelitian yang penting, dengan banyak peluang untuk inovasi dan peningkatan dalam desain dan implementasi struktur data. Untuk para peneliti dan praktisi, pemahaman mendalam tentang bagaimana Matematika Diskrit dapat diaplikasikan dalam konteks ini akan menjadi aset yang sangat berharga.

DAFTAR PUSTAKA

Biggs, N.L. (2002) *Discrete Mathematics*. 2nd Ed. Oxford University Press.

Gallian, J. (2012) *Contemporary Abstract Algebra*. 8th Ed. Cengage Learning.

Goodrich, M.T., Tamassia, R. and Goldwasser, M.H. (2014) *Data Structures and Algorithms in Java*. Wiley.

Rosen, K. (2011) *Discrete Mathematics and Its Applications*. McGraw Hill.

BAB 6
PENERAPAN MATEMATIKA DISKRIT DALAM BAHASA FORMAL

Oleh Indah Resti Ayuni Suri

6.1 Pendahuluan

Matematika diskrit adalah disiplin ilmu yang kita pelajari di sekolah dan dapat diterapkan di banyak cabang ilmu pengetahuan lainnya. Dalam matematika diskrit, berbagai topik dibahas, termasuk bilangan bulat, ekspresi logika, teori grafik, dan banyak lagi. Sejumlah penelitian telah menggunakan teori graf sebagai subjek hingga saat ini. Secara umum, suatu graf dapat dianggap sebagai suatu himpunan tak kosong dengan simpul sebagai elemennya dan sisi sebagai elemennya.

Topik matematika dijelaskan dengan menggunakan bahasa formal dengan cara yang sangat tepat, mudah dipahami, dan lugas. Kita dapat mengungkapkan dan menjelaskan topik matematika diskrit secara tepat dan tanpa ambiguitas dengan menggunakan bahasa formal. Hal ini penting dalam banyak disiplin ilmu, termasuk teori grafik, teori himpunan, enkripsi, dan algoritma, di mana matematika diskrit memainkan peran penting dalam pemodelan dan pemecahan masalah.

6.2 Induksi Matematika

Teknik umum untuk menunjukkan bahwa klaim tertentu akurat untuk setiap bilangan asli adalah induksi matematika. Pendekatan ini juga didukung secara teknis dan biasanya

digunakan untuk memverifikasi kebenaran semua bilangan asli (bilangan bulat non-negatif).

Metode pembuktian ini terdiri dari dua langkah, yaitu:

1. Pernyataan yang menunjukkan bahwa itu berlaku untuk bilangan 1.
2. Pernyataan yang menunjukkan bahwa itu berlaku untuk bilangan n. Maka pernyataan itu juga berlaku untuk bilangan $n + 1$.

Untuk membuktikan pernyataan bahwa jumlah n bilangan asli pertama, yaitu $1 + 2 + \cdots . + n$, sama dengan $\frac{n(n+1)}{2}$, kita dapat mengikuti langkah-langkah berikut untuk membuktikan bahwa pernyataan ini benar untuk setiap bilangan asli:

1. Pernyataan tersebut memulai dengan menguji kebenaran pada n = 1. Ini jelas benar karena jumlah 1 bilangan asli pertama adalah $\frac{n(n+1)}{2} = 1$. Dengan kata lain, pernyataan ini terbukti benar untuk $n = 1$.

2. Selanjutnya, pernyataan tersebut menyatakan bahwa jika benar untuk $n = k$, maka pernyataan tersebut juga benar untuk $n = k + 1$. Ini dapat dijelaskan sebagai berikut:

   - Jika kita berasumsi bahwa pernyataan tersebut benar ketika $n = k$, yang artinya, kita menganggap bahwa pernyataan tersebut valid untuk kasus di mana $n = k$, ini berarti...

   $$1 + 2 + \ldots + k = \frac{k(k+1)}{2}$$

   - Menambahkan k+1 pada kedua sisi:

$$1 + 2 + \ldots + k + (k+1) = \frac{k(k+1)}{2} + (k+1)$$

- Dengan melakukan manipulasi aljabar, kita bisa mendapatkan:

$$\frac{k(k+1)}{2} + (k+1) = \frac{k(k+1)}{2} + \frac{k(k+1)}{2}$$

$$= \frac{(k+1)(k+2)}{2}$$

$$= \frac{(k+1)(k+1)+1}{2}$$

- Jadi: $1 + 2 + \ldots + k + (k+1) = \frac{(k+1)(k+2)}{2}$

- Dengan demikian, dapat disimpulkan bahwa pernyataan tersebut benar untuk $n = k + 1$.

3. Induksi matematika mengarah pada kesimpulan bahwa setiap bilangan asli n dipengaruhi oleh pernyataan ini. Secara formal, induksi matematika dapat didefinisikan sebagai berikut:

   Definisi 1

   Misalkan kita memiliki pernyataan P(n) yang dapat berupa benar atau salah untuk setiap bilangan asli n. Misalkan:

   1. P(1) adalah pernyataan yang benar.
   2. Jika P(n) adalah benar, maka P(n+1) juga benar.

   Dengan begitu, kita bisa menyatakan bahwa $P(n)$ adalah benar untuk setiap bilangan bulat positif n. Tahap awal dari argumen ini disebut sebagai Langkah Dasar, sedangkan tahap berikutnya disebut sebagai Langkah

Induktif. Dalam Langkah Induktif, kita mengasumsikan bahwa pernyataan $P(i)$ adalah benar untuk semua bilangan $i \leq n$ (i yang kurang dari atau sama dengan n), maka formulasi induksi matematika ini disebut Bentuk Kuat Induksi Matematika.

6.3 Aljabar Boolean

Aljabar Boolean (Komputasi Aljabar) adalah sebuah sistem aljabar yang menggunakan himpunan dengan dua operator biner yang didefinisikan di dalamnya, yaitu:

+ Operator Penambahan, yang setara dengan operator OR (∨).
• Operator Perkalian, yang setara dengan operator AND (∧

Dalam aljabar Boolean, juga terdapat hukum logika yang menghubungkan operator-operator tersebut:

1) Operator AND (∧) biasanya diganti dengan tanda asterisk (*).
2) Operator OR (∨) biasanya diganti dengan tanda plus (+).
3) Operator Negasi (~) biasanya diganti dengan komplemen.

Sebagai contoh, jika kita memiliki himpunan B yang didefinisikan dengan operator + dan *, serta operator uner (') dan elemen 0 dan 1 yang berbeda di dalamnya, maka keseluruhan struktur $\{B, +, *, ', 0, 1\}$ dapat disebut sebagai aljabar Boolean.

1. Eksprensi Boolen

Dalam aljabar Boolean $\{B, +, ., ', 0, 1\}$ suatu ekspresi Boolean dapat dibentuk dengan menggunakan komponen-komponen berikut:

I. Setiap elemen dalam ekspresi Boolean harus berasal dari himpunan B. Dalam konteks aljabar Boolean, himpunan B biasanya terdiri dari dua elemen, yaitu 0 dan 1, yang mewakili nilai Boolean "False" dan "True" secara berturut-turut.

II. Setiap peubah dalam ekspresi Boolean dapat digunakan. Peubah ini adalah simbol-simbol yang digunakan untuk mewakili nilai Boolean, seperti A, B, C, dst.

III. Jika e_1 dan e_2 adalah ekspresi Boolean, maka Anda dapat melakukan operasi-operasi berikut pada mereka:

- $e_1 + e_2$ menggambarkan operasi logika OR (atau disebut juga penjumlahan Boolean).
- $e_1 . e_2$ menggambarkan operasi logika AND (atau disebut juga perkalian Boolean).
- e_1' menggambarkan operasi logika NOT (atau disebut juga negasi Boolean).

Contoh:

Mari kita tentukan hasil evaluasi ekspresi Boolean berikut:

$a' + (b.c)$

Penyelesaian

Misalkan $a = 0, b = 1$ dan $c = 0$, maka hasil evaluasi ekspresi

$0' + (1.0) = 1.0 = 0$

Dua ekspresi Boolean dianggap ekivalen (dilambangkan dengan '=') jika keduanya memberikan hasil yang sama untuk setiap kombinasi nilai pada n peubah. Hal ini dapat dinyatakan atau dibuktikan dengan menggunakan tabel kebenaran yang menunjukkan semua kemungkinan nilai-nilai peubah dan perbandingan hasil ekspresi-ekspresi tersebut.

2. Hukum – Hukum Aljabar Boolean

Hukum-hukum aljabar Boolean didasarkan pada prinsip-prinsip aljabar yang berlaku dalam himpunan.

Tabel Hukum – Hukum Aljabar Boolean

1.	Hukum identitas	1. $A + 0 = A$ 2. $A * 1 = A$
2.	Hukum idempoten	1. $A + A = A$ 2. $A * A = A$

3.	Hukum komplemen (Hukum Inversi)	1. $A + A' = 1$ 2. $A * A' = 0$
4.	Hukum dominasi	1. $A.0 = 0$ 2. $A + 1 = 1$
5.	Hukum involusi	1. $(A')' = A$
6.	Hukum penyerapan	1. $A + (A * B) = A$ 2. $A * (A + B) = A$
7.	Hukum komutatif	1. $A + B = B + A$ 2. $A * B = B * A$
8.	Hukum asosiatif	1. $(A + B) + C = A + (B + C)$ 2. $(A * B) * C = A * (B * C)$
9.	Hukum distributif	1. $A * (B + C) = (A * B) + (A * C)$ 2. $A + (B * C) = (A + B) * (A + C)$
10.	Hukum de morgan	1. $(A + B)' = A' * B'$ 2. $(A * B)' = A' + B'$
11.	Hukum 0/1	1. $0' = 1$ 2. $1' = 0$

3. Fungsi Boolean

Setiap ekspresi Boolean dapat dianggap sebagai fungsi

Boolean. Fungsi Boolean, juga dikenal sebagai fungsi biner, adalah pemetaan dari sekumpulan n-tuple terurut (B^n) ke sekumpulan elemen biner (B), yang dinyatakan melalui ekspresi Boolean. Dengan kata lain, fungsi ini mengambil n-tuple sebagai masukan dan menghasilkan satu elemen biner sebagai keluaran. Ini bisa direpresentasikan sebagai:

$$f : B^n \rightarrow B$$

Dalam konteks ini, B^n adalah himpunan pasangan terurut n-tuple yang berada dalam domain (daerah asal) B. Fungsi ini kemudian mengonversi masukan n-tuple ini menjadi elemen biner di dalam himpunan gambaran (daerah tujuan) B.

6.4 Ragam Bahasa

Masyarakat umum, anak sekolah, dan mahasiswa yang mempelajari variasi bahasa dapat mempelajari bahasa Indonesia. Dalam berbagai penggunaan bahasa yang sama, misalnya lisan dan tulisan. Berbicara lebih banyak ditemui dalam aktivitas sehari-hari seperti ngobrol, puisi, pidato, ceramah, dan sebagainya. Penyesuaian bahasa adalah nama lain untuk penggunaan ini.

Dengan kemajuan ilmu pengetahuan dan teknologi serta meningkatnya hubungan sosial di kalangan remaja, telah terjadi perubahan besar dalam cara orang dewasa, remaja, dan anak-anak mempelajari bahasa Indonesia. Bahasa gaul, atau bahasa sosial, seperti "elo", "saya", "kamu tahu", "cewek", dan istilah-istilah lain seperti ini kemudian berkembang.

Dialek daerah ditemukan variasi bahasa yang tidak menganut kaidah konvensional. Berdasarkan apa yang telah dikemukakan di atas, yang dimaksud dengan bahasa tidak baku adalah bahasa yang lazim digunakan dalam suasana informal dengan keluarga, tulisan pribadi, dan pergaulan sehari-hari, namun kurang tepat digunakan dalam suasana formal seperti penulisan ilmiah, diskusi. , pembicaraan dalam lingkungan formal, dan situasi lainnya.

1. Pengertian Ragam Bahasa Resmi (Formal)

Bahasa yang digunakan dalam konteks resmi atau formal, seperti dalam surat resmi, pidato, dokumen, atau karya sastra, disebut bahasa formal. Menurut EYD, bahasa formal seringkali menggunakan tata bahasa yang padat, sederhana, sopan, dan menggunakan bahasa standar baik dalam bentuk lisan maupun tulisan. Bahasa yang digunakan dalam konteks resmi, seperti komunikasi dan berbicara dengan seseorang yang tidak kita kenal baik atau yang mempunyai kedudukan dan pangkat lebih tinggi, disebut dengan bahasa resmi atau formal.

2. Ciri-ciri Ragam Bahasa Resmi (Formal)

1. Digunakan dalam situasi resmi
2. Nada bicara yang cenderung dan
3. Kalimat yang digunakan kalimat lengkap

3. Ragam Bahasa Berdasarkan Situasi Pemakaiannya

Variasi bahasa dibedakan menjadi tiga kategori berdasarkan konteks penggunaannya: ragam bahasa formal, ragam bahasa semiformal, dan varian bahasa nonformal. Standar-standar berikut ini menjadi pertimbangan variasi bahasa formal untuk menjadikan suatu bahasa resmi (Kementerian Pendidikan dan Kebudayaan, 2013: 6).

1. Stabilitas dinamis dalam penerapan norma, memungkinkan adanya perubahan istilah dan bahasa yang tepat namun tetap tidak terlalu kaku dan kaku.
2. Penggunaan kaidah tata bahasa yang konsisten dan jelas.
3. Jangan mempersingkat kata-kata; sebagai gantinya, gunakan bentuk kata lengkap.
4. Penggunaan imbuhan yang jelas dan teratur.
5. Gunakan pengucapan dan ejaan standar di semua varian bahasa lisan dan tulisan.

Perbedaan antara ragam formal, ragam semiformal, dan ragam nonformal dapat dilihat dari beberapa syarat berikut untuk ragam bahasa formal:

1. Pokok bahasan yang dibicarakan,
2. Hubungan antara pembicara dan pendengar,
3. Medium/bahasa media yang digunakan baik lisan maupun tulisan,
4. Setting atau lingkungan tempat terjadinya percakapan, dan
5. Keadaan di mana hal itu terjadi.

6.5 Penerapan Matematika Diskrit Dalam Bahasa Formal

Berbagai struktur dan konsep matematika digunakan untuk mewakili dan memecahkan masalah yang melibatkan objek diskrit atau independen dalam penerapan matematika diskrit. Matematika yang dapat dihitung atau dihitung secara mandiri, seperti bilangan bulat, himpunan, grafik, logika, dan lain-lain, merupakan penekanan dari matematika diskrit.

Topik matematika dijelaskan dengan menggunakan bahasa formal dengan cara yang sangat tepat, mudah dipahami, dan lugas. Matematikawan dapat berkomunikasi secara tepat menggunakan bahasa formal, yang juga membantu menjelaskan konsep, teorema, dan penalaran matematika dengan jelas.

Berikut beberapa contoh penerapan matematika diskrit dalam bahasa formal:

Berikut beberapa contoh penerapan matematika diskrit dalam bahasa formal:

1. Teori Bilangan: Demonstrasi teorema dasar bilangan prima merupakan ilustrasi penerapan dalam bahasa formal. Misalnya, menurut teorema dasar aritmatika, setiap bilangan bulat positif dapat dibagi secara unik menjadi faktor primanya.

2. Teori Himpunan: Kita dapat mendeskripsikan operasi pada himpunan seperti penyatuan, perpotongan, dan perbedaan dalam bahasa formal dan mendemonstrasikan fitur-fitur yang terkait. Misalnya, kita dapat mendefinisikan gabungan dari dua himpunan A dan B sebagai {x | x adalah anggota dari A atau B}.

3. Teori Graf: Definisi graf, simpul, sisi, dan atribut graf terlibat dalam penggunaan bahasa formal dalam teori graf. Misalnya, kumpulan simpul dan sekumpulan sisi dapat digunakan untuk mendefinisikan suatu graf.

4. Logika Matematika: Kemampuan mendeskripsikan pernyataan, konjungsi, disjungsi, implikasi, dan istilah lain dengan benar sangat penting dalam logika matematika. Selain itu, bahasa formal membantu pembuktian proposisi dan argumen dalam logika matematika.

5. Algoritma dan Struktur Data: Dengan menggunakan notasi matematika dan kodesemu, algoritma dapat didefinisikan secara lengkap dalam bahasa formal. Untuk memahami proses dan kompleksitas struktur data seperti tumpukan, antrian, dan pohon, deskripsi formal konstruksinya juga dimungkinkan.

Penerapan matematika diskrit dalam bahasa formal sangat banyak; contoh-contoh yang disebutkan hanyalah sedikit. Pemodelan,

analisis, dan penyelesaian masalah kompleks dalam berbagai disiplin ilmu, termasuk ilmu komputer, ilmu data, teori informasi, dan lain-lain, dimungkinkan oleh matematika diskrit dan bahasa formal.

Penggunaan simbol formal dan batasan yang ketat untuk mendeskripsikan gagasan matematika diskrit merupakan kebutuhan untuk menerapkan matematika diskrit dalam bahasa formal. Untuk menghilangkan ambiguitas dan salah tafsir dan untuk memungkinkan komunikasi yang jelas dan tepat mengenai kesulitan matematika, bahasa formal digunakan.

Elemen-elemen berikut ini penting dalam bahasa formal untuk matematika diskrit:

1. Simbol matematika: Untuk menjelaskan konsep dan prosedur matematika, bahasa formal menggunakan simbol matematika yang telah ditentukan sebelumnya. Misalnya saja simbol penjumlahan, pengurangan, perkalian, dan pembagian dalam matematika, seperti + dan -.

2. Variabel dan konstanta: Dalam persamaan matematika, variabel digunakan untuk menyatakan besaran yang dapat berubah-ubah. Sebaliknya, konstanta adalah nilai tetap yang tidak pernah berubah. Variabel dan konstanta sering digunakan dalam matematika diskrit untuk menyatakan bilangan bulat, himpunan, dan objek diskrit lainnya.

3. Fungsi dan relasi: Terdapat teknik untuk mendefinisikan fungsi dan relasi matematika dalam bahasa formal. Suatu fungsi memindahkan setiap elemen pada himpunan pertama ke satu elemen pada himpunan kedua, sehingga membentuk korespondensi antara elemen-elemen pada kedua himpunan tersebut. Sebaliknya, relasi adalah hubungan antara dua atau lebih objek matematika.

4. Aksioma dan definisi: Dalam bahasa formal, gagasan penting dalam matematika diskrit ditentukan oleh aksioma dan definisi. Aksioma adalah pernyataan yang dianggap benar tanpa pembenaran tambahan, sedangkan definisi memberikan makna pada konsep matematika.

5. Pembuktian dan teorema: Pembuktian matematis, yaitu argumen logis yang digunakan untuk menunjukkan kebenaran suatu klaim atau teorema, juga dibuat menggunakan bahasa formal.

Notasi formal sifat bilangan bulat merupakan ilustrasi dasar penggunaan bahasa formal dalam matematika diskrit.Misalnya, kita dapat menggunakan bahasa formal untuk menyatakan bahwa untuk setiap bilangan bulat positif n, n + 1 juga merupakan bilangan bulat positif:

$\forall n \in Z^+, n + 1 \in Z^+$

Keterangan:

∀ adalah simbol kuantor universal yang berarti "untuk setiap" atau "untuk semua."

n adalah variabel yang mewakili bilangan bulat positif.

Z^+ adalah himpunan bilangan bulat positif.

∈ adalah simbol yang menunjukkan "elemen dari."

Kita dapat mengungkapkan dan menjelaskan topik matematika diskrit secara tepat dan tanpa ambiguitas dengan menggunakan bahasa formal. Hal ini penting dalam banyak disiplin ilmu, termasuk teori grafik, teori himpunan, enkripsi, dan algoritma, di mana matematika diskrit memainkan peran penting dalam pemodelan dan pemecahan masalah.

DAFTAR PUSTAKA

Dr. Rina Filia Sari, M.Si, Rima Aprilia, M.Si. *Matematika Diskrit Dan Ayat Alquran*. Medan: CV. Pusdikra Mitra Jaya, 2021.

Novia Arianti, S.Si., M.Pd, Nuril Lutvi Azizah, S.Si., M.Si. *MATEMATIKA DISKRIT*. Sidoarjo: UMSIDA PRESS, 2020.

Prof.Dr. Benny Pinontoan, M.Sc, Jullia Titaley. *MATEMATIKA DISKRIT I*. Manado: CV. PATRA MEDIA GRAFINDO BANDUNG, 2019.

Sujinah, idhoofiyatul fatin, Dian Karina Rachmawati. *Buku Ajar Bahasa Indonesia*. Surabaya: UM Surabaya Publishing, 2018.

BAB 7
PENERAPAN MATEMATIKA DISKRIT DALAM KRIPTOGRAFI

Oleh Wirawan Istiono

7.1 Matematika Diskrit dalam Kriptografi

Disiplin kriptografi, meliputi desain struktur aman dan kata sandi untuk sistem komputer dan perangkat elektronik lainnya, pada dasarnya didasarkan pada matematika diskrit. Fenomena ini dapat dikaitkan, setidaknya sebagian, dengan fakta bahwa komputer mengirimkan data dalam satuan diskrit, terkadang disebut sebagai bit, yang terpisah dan unik satu sama lain. Teori bilangan adalah komponen mendasar dari matematika diskrit yang memainkan peran penting dalam bidang kriptografi, memungkinkan kriptografer untuk membuat dan menguraikan kata sandi numerik. Karena nilai moneter yang besar dan sifat sensitif dari informasi yang dipertaruhkan, sangat penting bagi kriptografer untuk memiliki dasar yang kuat dalam teori bilangan sebagai prasyarat untuk menunjukkan kemampuan mereka dalam merancang mekanisme sandi dan enkripsi yang kuat.

Basis data relasional mengacu pada jenis sistem manajemen basis data yang mengatur dan menyimpan data secara terstruktur, memanfaatkan tabel dengan baris

Database relasional merupakan bagian integral dari operasi hampir semua organisasi yang memerlukan pengelolaan dan pengorganisasian data karyawan, klien, atau sumber daya. Database relasional menetapkan koneksi antara atribut dari bagian tertentu dari informasi. Dalam konteks database yang menampung informasi pelanggan, sifat relasional dari database ini memfasilitasi kemampuan sistem komputer untuk membuat koneksi antara nama klien, alamat, nomor telepon, dan data relevan lainnya. Proses tersebut di atas dijalankan dengan menggunakan ide matematis himpunan dalam bidang matematika diskrit. Set memfasilitasi organisasi dan pengaturan informasi ke dalam pengelompokan logis. Mengingat bahwa setiap potongan informasi dan ciri-ciri yang terkait berbeda, pengaturan informasi ini dalam database memerlukan pemanfaatan teknik matematika diskrit.

Penerapan matematika diskrit dalam bidang logistik. Logistik adalah disiplin akademis yang berfokus pada analisis sistematis dan pengelolaan arus informasi, barang, dan jasa dalam suatu organisasi. Keberadaan logistik bergantung pada prinsip dan konsep matematika diskrit. Alasannya adalah karena logistik secara ekstensif menggunakan grafik dan teori grafik, yang merupakan sub-disiplin matematika diskrit. Bidang teori graf memungkinkan transformasi masalah logistik yang rumit menjadi representasi yang disederhanakan yang terdiri dari node dan edge. Matematikawan

memiliki kemampuan untuk mengevaluasi grafik ini dengan menggunakan prinsip dan teknik teori grafik untuk memastikan rute optimal untuk pengiriman atau menyelesaikan berbagai masalah logistik.

Algoritme komputer adalah aspek mendasar dari ilmu komputer dan memainkan peran penting dalam memecahkan masalah komputasi. Algoritma ini adalah prosedur langkah demi langkah

Algoritma dapat didefinisikan sebagai kumpulan instruksi atau aturan yang mengatur operasi sistem komputer. Pembentukan peraturan ini berasal dari prinsip matematika diskrit. Matematika diskrit digunakan oleh pemrogram komputer untuk mengembangkan algoritme yang dioptimalkan untuk efisiensi. Desain ini menggabungkan pemanfaatan matematika diskrit untuk memastikan kompleksitas komputasi suatu algoritma, sehingga menunjukkan efisiensinya dalam hal waktu eksekusi. Peningkatan kecepatan komputasi komputer modern dapat dikaitkan dengan pemanfaatan prinsip matematika diskrit dalam aplikasi algoritmik.

7.2 Apa itu Kriptografi

Kriptografi adalah teknik mapan yang digunakan untuk melindungi informasi dan menyediakan komunikasi yang aman dengan menggunakan kode, sehingga memastikan bahwa hanya penerima yang berwenang yang memiliki kemampuan untuk memahami dan memanipulasi informasi.

Dalam bidang ilmu komputer, kriptografi mencakup pemanfaatan prinsip matematika dan perhitungan algoritmik untuk melindungi informasi dan menyediakan komunikasi yang aman. Disiplin ini melibatkan transformasi pesan dengan cara yang membuatnya sulit dibaca. Algoritme deterministik digunakan di berbagai domain, termasuk pembuatan kunci kriptografi, penandatanganan digital, verifikasi, perlindungan privasi data, penjelajahan web internet, dan saluran komunikasi yang aman seperti transaksi kartu kredit dan pertukaran email (Richards, 2021).

7.2.1 Teknik Kriptografi

Kriptografi menunjukkan hubungan yang erat dengan bidang akademik kriptologi dan kriptanalisis. Metode yang disebutkan di atas mencakup beberapa taktik, seperti penggunaan mikrodot, penggabungan kata dengan visual, dan strategi lain yang digunakan untuk menyembunyikan informasi selama penyimpanan atau pengirimannya. Di era kontemporer yang didominasi oleh teknologi komputer, kriptografi terutama terkait dengan proses mengubah teks biasa (sering disebut teks-jelas) menjadi teks sandi melalui enkripsi, dan selanjutnya membalikkan proses ini melalui dekripsi. Individu yang terlibat dalam domain khusus ini biasanya disebut sebagai kriptografer (Richards, 2021).

Kriptografi modern difokuskan untuk mencapai empat tujuan utama:

1. Kerahasiaan. Informasi yang disampaikan tidak dapat dipahami oleh individu yang bukan penerima yang dimaksud.
2. Integritas. Integritas informasi tetap utuh selama penyimpanan dan transmisi, karena setiap modifikasi yang dibuat pada data dapat dengan mudah diidentifikasi.
3. Non-penolakan. Individu atau entitas yang bertanggung jawab untuk menghasilkan atau menyebarkan informasi tidak dapat kemudian mengingkari niat awal mereka dalam produksi atau komunikasi materi tersebut.
4. Autentikasi. Verifikasi identifikasi dan validasi asal/tujuan informasi dapat dikonfirmasikan bersama oleh pengirim dan penerima.

Cryptosystems ditetapkan sebagai prosedur dan protokol yang memenuhi satu atau lebih kriteria yang disebutkan di atas. Sistem kripto tidak hanya mencakup algoritme matematika dan perangkat lunak komputer, tetapi juga mencakup tata kelola perilaku manusia, seperti pemilihan kata sandi yang rumit dan sulit diprediksi, praktik keluar dari sistem yang tidak aktif, dan menghindari membocorkan operasi sensitif ke siapa pun di luar sistem (Richards, 2021).

7.2.2 Algoritma Kriptografi

Kriptosistem menggunakan metode kriptografi, terkadang disebut sebagai cipher, untuk menyandikan dan mendekode pesan, sehingga memastikan kerahasiaan dan integritas komunikasi antara sistem komputer, perangkat, dan aplikasi.

Cipher suite terdiri dari algoritme berbeda yang melayani tujuan tertentu di dalam sistem kriptografi. Algoritme ini termasuk satu untuk enkripsi, satu lagi untuk otentikasi pesan, dan satu lagi untuk pertukaran kunci. Prosedur ini, yang diintegrasikan ke dalam protokol dan diimplementasikan oleh perangkat lunak yang berfungsi pada sistem operasi (OS) dan sistem komputer jaringan, meliputi (Richards, 2021):

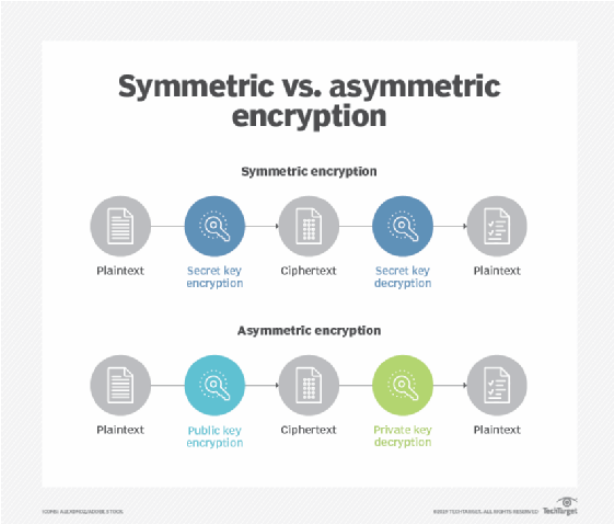
- Proses menghasilkan kunci publik dan pribadi untuk tujuan enkripsi dan dekripsi data.
- Pemanfaatan teknik penandatanganan dan verifikasi digital memainkan peran penting dalam memastikan otentikasi pesan.
- Proses pertukaran kunci adalah aspek mendasar dari sistem kriptografi, di mana dua atau lebih pihak membuat kunci rahasia bersama untuk komunikasi yang aman.

7.2.3 Jenis Kriptografi

Metode enkripsi kunci tunggal atau kunci simetris menghasilkan cipher blok dengan panjang yang telah ditentukan dalam bentuk bit. Block cipher ini disertai dengan kunci rahasia, yang digunakan oleh pembuat atau pengirim untuk mengenkripsi data, dan kemudian digunakan oleh penerima untuk mendekripsi informasi yang dienkripsi. Contoh kriptografi kunci simetris yang dapat dikutip adalah Advanced Encryption Standard (AES). Standar Enkripsi Lanjutan (AES) secara resmi diperkenalkan pada November 2001 oleh Institut Nasional Standar dan Teknologi (NIST)

sebagai Standar Pemrosesan Informasi Federal (FIPS 197). Tujuan utamanya adalah untuk melindungi informasi sensitif. Standar ini diwajibkan secara hukum oleh pemerintah Amerika Serikat dan digunakan secara luas di sektor swasta (Jashkothari1, 2023).

Pada bulan Juni 2003, pemerintah Amerika Serikat memberikan persetujuan untuk penggunaan AES dalam menangani material rahasia. Spesifikasi diimplementasikan secara global baik dalam perangkat lunak maupun perangkat keras, dan bebas dari kewajiban royalti apa pun. Advanced Encryption Standard (AES) dianggap sebagai penerus Data Encryption Standard (DES) dan Triple Data Encryption Algorithm (DES3). Untuk meningkatkan langkah-langkah keamanan terhadap kekerasan dan serangan jahat lainnya, panjang kunci yang lebih besar dari 128-bit, 192-bit, dan 256-bit digunakan (Jashkothari1, 2023).



Gambar 7.1 Kriptografi simetris menggunakan kunci tunggal sedangkan kriptografi asimetris menggunakan pasangan kunci untuk mengenkripsi dan mendekripsi data

(Sumber : Jashkothari1, 2023)

Teknik enkripsi kunci publik atau kunci asimetris menggunakan dua set kunci, yaitu kunci publik yang dihubungkan dengan pembuat atau pengirim untuk tujuan mengenkripsi pesan, dan kunci privat yang hanya diketahui oleh pembuatnya (sampai kunci tersebut diketahui). dikompromikan atau diungkapkan secara sukarela) untuk tujuan mendekripsi informasi ini (Jashkothari1, 2023).

Kriptografi kunci publik mencakup berbagai contoh, seperti (Jashkothari1, 2023):

- Algoritme enkripsi RSA, yang digunakan secara luas di internet, adalah sistem kriptografi yang diakui secara luas.
- Elliptic Curve Digital Signature Algorithm (ECDSA) digunakan dalam konteks Bitcoin. Selain itu, Algoritma Tanda Tangan Digital (DSA) telah diakui secara resmi oleh National Institute of Standards and Technology (NIST) sebagai Standar Pemrosesan Informasi Federal untuk keperluan tanda tangan digital, sebagaimana diuraikan dalam FIPS 186-4.
- Pertukaran kunci Diffie-Hellman adalah protokol kriptografi yang memungkinkan dua pihak membuat kunci rahasia bersama

melalui saluran komunikasi yang tidak aman. Protokol ini dikembangkan oleh Whit

Untuk menjaga integritas data di bidang kriptografi, fungsi hash digunakan. Fungsi-fungsi ini memiliki kemampuan untuk memberikan keluaran yang konsisten berdasarkan nilai masukan yang diberikan, sehingga memudahkan pemetaan data ke ukuran yang telah ditentukan. Ada beberapa fungsi hash kriptografi, yaitu SHA-1 (Secure Hash Algorithm 1), SHA-2, dan SHA-3 (Jashkothari1, 2023).

7.2.4 Masalah Kriptografi

Musuh memiliki kemampuan untuk menghindari tindakan kriptografi, menyusup ke sistem komputer yang dipercayakan dengan tugas enkripsi dan dekripsi data, dan memanfaatkan kerentanan yang berasal dari implementasi yang tidak memadai, seperti penggunaan kunci default. Kriptografi berfungsi untuk meningkatkan keamanan informasi sensitif dengan menghalangi akses tidak sah ke pesan dan data terenkripsi melalui pemanfaatan metode enkripsi (Mehta, 2018).

Kekhawatiran yang meningkat atas kemampuan komputasi komputasi kuantum dalam hal mengkompromikan standar enkripsi yang ada telah mendorong Institut Nasional Standar dan Teknologi (NIST) untuk mengeluarkan permintaan pengiriman ilmiah dari komunitas matematika dan ilmiah pada tahun 2016, dengan tujuan membangun standar baru untuk kriptografi kunci publik.

Berbeda dengan sistem komputer kontemporer, komputasi kuantum menggunakan bit kuantum (qubit) yang mampu mewakili 0 dan 1 secara bersamaan, memungkinkan eksekusi dua perhitungan secara bersamaan. Menurut National Institute of Standards and Technology (NIST), pembentukan kerangka kerja standar untuk algoritma yang dikenal publik dan dapat dipahami diperlukan untuk memastikan pendekatan yang aman dalam infrastruktur saat ini, meskipun pembangunan komputer kuantum skala besar mungkin tidak terjadi dalam sepuluh tahun mendatang. Batas waktu penyerahan proposal terjadi pada November 2017, dan diperkirakan bahwa analisis rencana ini akan berlangsung selama tiga hingga lima tahun (Richards, 2021).

7.2.5 Sejarah kriptografi

Istilah "kriptografi" berasal dari kata Yunani "kryptos," yang diterjemahkan sebagai "tersembunyi.". Awalan "crypt-" menunjukkan konsep penyembunyian atau ruang penyimpanan yang aman, sedangkan akhiran "-graphy" menandakan tindakan atau proses penulisan.

Dimulainya kriptografi umumnya dikaitkan dengan sekitar 2000 SM, ketika orang Mesir mulai menggunakan hieroglif sebagai semacam enkripsi. Entitas yang disebutkan di atas terdiri dari simbol piktografik yang rumit, yang makna semantik lengkapnya dipahami secara eksklusif oleh sekelompok individu terpilih yang memiliki pengetahuan khusus (Mehta, 2018).

Contoh awal yang terdokumentasi tentang penggunaan sandi kontemporer mungkin dikaitkan dengan Julius Caesar (100 SM hingga 44 SM), yang meragukan keandalan utusannya selama interaksi dengan gubernur dan pejabatnya. Akibatnya, individu menyusun mekanisme di mana setiap karakter dalam korespondensinya diganti dengan karakter yang terletak tiga posisi di sebelahnya dalam alfabet Romawi (Richards, 2021).

Di zaman sekarang, bidang kriptografi telah menjadi domain tempat matematikawan dan ilmuwan komputer terkemuka terlibat dalam kompetisi intelektual yang intens. Penyimpanan dan distribusi informasi sensitif yang aman telah muncul sebagai penentu penting keberhasilan dalam operasi militer dan usaha komersial.

Pemerintah memberlakukan pembatasan pada kriptografi untuk mencegah entitas tertentu, baik domestik maupun asing, memperoleh sarana transmisi dan penerimaan informasi rahasia yang berpotensi membahayakan kepentingan nasional. Pembatasan ini mencakup berbagai tindakan, termasuk pembatasan penggunaan dan ekspor perangkat lunak kriptografi, serta pembatasan penyebaran konsep matematika kepada publik yang dapat digunakan dalam pengembangan sistem kriptografi (Kumar*et al.*, 2019).

Namun demikian, munculnya internet telah memfasilitasi penyebaran program perangkat lunak yang kuat dan, secara

signifikan, prinsip dasar kriptografi. Akibatnya, sejumlah besar kriptosistem dan konsep mutakhir saat ini dapat diakses oleh masyarakat umum (Richards, 2021).

7.3 Bagaimana Menggabungkan Matematika Diskrit Dengan Kriptografi

Integrasi matematika diskrit dan kriptografi memerlukan penerapan prinsip dan struktur matematika yang beragam dalam pengembangan algoritme dan sistem kriptografi yang kuat. Kriptografi mencakup penerapan strategi komunikasi yang aman dalam menghadapi musuh potensial, sedangkan matematika diskrit berfungsi sebagai kerangka dasar untuk membangun dan mengevaluasi protokol kriptografi. Salah satu pendekatan yang mungkin untuk mengintegrasikan kedua disiplin ini adalah sebagai berikut (Mehta, 2018):

1. Teori bilangan berfungsi sebagai kerangka dasar untuk berbagai teknik kriptografi. Konsep dasar meliputi bilangan prima, aritmatika modular, dan pembagi persekutuan terbesar. Bilangan prima digunakan dalam teknik kriptografi kunci publik seperti RSA, sedangkan aritmatika modular memainkan peran penting dalam proses enkripsi dan pembuatan kunci.
2. Aritmatika modular memainkan peran penting dalam bidang kriptografi. Teknologi ini digunakan dalam pengembangan sandi, pembuatan kunci kriptografi, dan implementasi berbagai metode

enkripsi. Pemahaman invers modular memainkan peran penting dalam bidang dekripsi dan kriptografi asimetris.

3. Konsep logaritma diskrit sangat penting dalam protokol kriptografi seperti pertukaran kunci Diffie-Hellman dan tanda tangan digital. Komputasi entitas ini menimbulkan tantangan yang signifikan, karena berfungsi sebagai dasar untuk memastikan keamanan berbagai sistem kriptografi.

4. Kriptografi sering kali melibatkan pemanfaatan medan berhingga, terkadang dikenal sebagai medan Galois. Bidang yang disebutkan di atas menemukan aplikasi dalam kode koreksi kesalahan, kriptografi kurva eliptik, dan metode enkripsi tertentu.

5. Aljabar Boolean memainkan peran penting dalam pengembangan dan evaluasi algoritma kriptografi, yaitu dalam ranah kriptografi kunci simetris dan pembuatan fungsi kriptografi. Penerapan konsep teori graf terbukti dalam pembuatan protokol jaringan yang aman dan analisis konektivitas antar node dalam komunikasi yang aman.

6. Combinatorics memainkan peran penting dalam pemeriksaan algoritma kriptografi, yaitu dalam evaluasi kekuatan kunci dan parameter keamanan. Prinsip-prinsip probabilitas dan statistik digunakan dalam analisis sifat stokastik dari algoritma dan protokol kriptografi. Generator angka acak memainkan peran penting dalam desain sistem yang aman dan evaluasi kemungkinan serangan potensial.

7. Teori pengkodean memainkan peran penting dalam pengembangan kode deteksi kesalahan dan koreksi kesalahan, yang penting untuk menjaga integritas dan ketergantungan data yang ditransmisikan di dalam sistem kriptografi. Teori grup berfungsi sebagai kerangka dasar untuk memahami struktur aljabar yang digunakan dalam banyak algoritma kriptografi, khususnya di bidang kriptografi kurva eliptik dan beberapa aspek kriptografi kunci publik.

Dalam konteks mengintegrasikan matematika diskrit dengan kriptografi, sangat penting untuk memiliki basis pengetahuan yang kuat di kedua domain tersebut. Mendapatkan pemahaman yang komprehensif tentang prinsip-prinsip matematika yang mendasari kriptografi adalah penting dalam memfasilitasi desain yang mahir dan analisis yang cermat dari sistem yang aman. Selain itu, sangat penting untuk tetap mengikuti perkembangan terbaru di kedua disiplin ilmu, karena bidang kriptografi terus mengalami evolusi untuk mengatasi masalah dan kerentanan yang muncul.

7.4 Contoh Cara Menggabungkan Matematika Diskrit dan Kriptografi

Tentu saja, inilah contoh sederhana yang menunjukkan bagaimana matematika diskrit dapat digabungkan dengan kriptografi:

7.4.1. Contoh: Enkripsi Modular menggunakan Matematika Diskrit

Dalam contoh ini, kita akan menggunakan aritmetika modular, sebuah konsep dari matematika diskrit, untuk membuat algoritme enkripsi dasar. Contoh ini untuk tujuan pendidikan dan tidak mewakili sistem kriptografi yang aman.

Langkah 1: Pembuatan Kunci

Pilih dua bilangan prima, p dan q: Misalkan p = 17 dan q = 19.

Hitung n = p * q: n = 323.

Hitung φ(n) = (p - 1) * (q - 1): φ(n) = 288. Ini adalah fungsi totient Euler.

Langkah 2: Pilih Kunci Enkripsi

Pilih bilangan bulat e sehingga 1 < e < φ(n) dan gcd(e, φ(n)) = 1: Katakanlah e = 5.

Langkah 3: Kunci Publik

Kunci publik terdiri dari nilai n dan e: Kunci publik = (323, 5).

Langkah 4: Kunci Pribadi

Hitung d, invers perkalian modular dari e modulo φ(n): d = 173. Dengan kata lain, d adalah nilai sehingga (e * d) % φ(n) = 1.

Enkripsi:

Konversikan pesan teks biasa "HELLO" menjadi angka menggunakan pemetaan sederhana (A=1, B=2, ..., Z=26): HELLO -> 8 5 12 12 15.

Enkripsi setiap nomor menggunakan kunci publik:

Terenkripsi 8: (8^5) % 323 = 32768 % 323 = 193

Terenkripsi 5: (5^5) % 323 = 3125 % 323 = 159

Terenkripsi 12: (12^5) % 323 = 248832 % 323 = 101

Terenkripsi 15: (15^5) % 323 = 759375 % 323 = 239

Teks sandi:

Pesan terenkripsi adalah 193 159 101 101 239.

Dekripsi:

Dekripsi setiap nomor terenkripsi menggunakan kunci privat:

Didekripsi 193: (193^173) % 323 = ...

Didekripsi 159: (159^173) % 323 = ...

Didekripsi 101: (101^173) % 323 = ...

Didekripsi 239: (239^173) % 323 = ...

Hasil Akhir:

Angka yang didekripsi dapat dipetakan kembali ke huruf (1=A, 2=B, ..., 26=Z), dan pesan asli "HELLO" dapat diperoleh.

Ingatlah bahwa contoh ini bersifat dasar dan tidak mempertimbangkan aspek keamanan atau kerentanan. Dalam sistem kriptografi dunia nyata, lebih banyak kompleksitas, langkah-langkah keamanan, dan analisis diperlukan untuk memastikan kerahasiaan dan integritas data.

Tentu saja, mari kita selidiki ilustrasi tambahan yang mencakup penggunaan aritmatika modular dan prinsip-prinsip dari teori bilangan di bidang kriptografi:

7.4.2. Contoh: Pertukaran Kunci Diffie-Hellman

Pertukaran kunci Diffie-Hellman adalah protokol kriptografi dasar yang menggunakan aritmatika modular dan logaritma diskrit untuk memungkinkan dua pihak bertukar rahasia bersama secara aman melalui saluran yang tidak aman.

Langkah 1: Membuat dan Berbagi Kunci

Alice dan Bob secara terbuka menyepakati bilangan prima besar, p = 23, dan modulo akar primitif p, g = 5.

Langkah 2: Pemilihan Kunci Pribadi

Alice memilih kunci pribadi, a = 6.

Bob memilih kunci pribadi, b = 15.

Langkah 3: Perhitungan Kunci Publik

Alice menghitung kunci publiknya, A = (g^a) % p = (5^6) % 23 = 8.

Bob menghitung kunci publiknya, B = (g^b) % p = (5^15) % 23 = 19.

Langkah 4: Pertukaran Kunci

Alice menerima kunci publik Bob, B = 19.

Bob menerima kunci publik Alice, A = 8.

Langkah 5: Perhitungan Rahasia Bersama

Alice menghitung rahasia bersama sebagai S = (B^a) % p = (19^6) % 23 = 2.

Bob menghitung rahasia bersama sebagai S = (A^b) % p = (8^15) % 23 = 2.

Hasil:

Baik Alice dan Bob sekarang memiliki nilai rahasia bersama yang sama, yang dapat mereka gunakan untuk enkripsi atau tujuan kriptografi lainnya.

Penjelasan:

Keamanan pertukaran kunci Diffie-Hellman bergantung pada kesulitan menghitung logaritma diskrit. Meskipun relatif mudah untuk menghitung (g^a) % p atau (g^b) % p, menemukan 'a' atau 'b' dari nilai-nilai tersebut secara komputasi tidak mungkin dilakukan tanpa mengetahui informasi tambahan. Hal ini membuat sangat sulit bagi penyadap untuk menentukan rahasia bersama bahkan jika mereka tahu p, g, A, dan B.

Dalam contoh ini, konsep matematika diskrit yang terkait dengan aritmatika modular dan sifat-sifat bilangan prima

memainkan peran penting dalam memastikan keamanan proses pertukaran kunci.

Ingatlah bahwa implementasi dunia nyata mencakup tindakan keamanan tambahan untuk mencegah potensi serangan dan kerentanan.

7.4.3. Contoh Enkripsi dan Dekripsi RSA

RSA (Rivest–Shamir–Adleman) adalah algoritma enkripsi asimetris yang banyak digunakan yang bergantung pada properti matematika bilangan prima besar.

Pembuatan Kunci:

Pilih dua bilangan prima berbeda, p = 17 dan q = 19.

Hitung n = p * q = 17 * 19 = 323.

Hitung φ(n) = (p - 1) * (q - 1) = 16 * 18 = 288.

Pilih eksponen enkripsi, e, sehingga 1 < e < φ(n) dan gcd(e, φ(n)) = 1. Katakanlah e = 5.

Hitung invers perkalian modular, d, dari e modulo φ(n), sehingga (e * d) % φ(n) = 1. Dalam kasus ini, d = 173.

Kunci Publik:

Kunci publik terdiri dari n dan e: Kunci publik = (323, 5).

Kunci Pribadi:

Kunci pribadi terdiri dari d: Kunci pribadi = 173.

Enkripsi:

Ubah pesan plaintext "SECRET" menjadi nilai numerik menggunakan pemetaan sederhana (A=1, B=2, ..., Z=26): SECRET -> 19 5 3 18 5 20.

Enkripsi setiap nomor menggunakan kunci publik:

Terenkripsi 19: (19^5) % 323 = 2476099 % 323 = 175.

Terenkripsi 5: (5^5) % 323 = 3125 % 323 = 159.

Terenkripsi 3: (3^5) % 323 = 243 % 323 = 243.

Terenkripsi 18: (18^5) % 323 = 1889568 % 323 = 198.

Terenkripsi 5: (5^5) % 323 = 3125 % 323 = 159.

Terenkripsi 20: (20^5) % 323 = 3200000 % 323 = 92.

Teks sandi:

Pesan terenkripsi adalah 175 159 243 198 159 92.

Dekripsi:

Dekripsi setiap nomor terenkripsi menggunakan kunci privat:

Didekripsi 175: (175^173) % 323 = ...

Didekripsi 159: (159^173) % 323 = ...

Didekripsi 243: (243^173) % 323 = ...

Didekripsi 198: (198^173) % 323 = ...

Didekripsi 159: (159^173) % 323 = ...

Didekripsi 92: (92^173) % 323 = ...

Hasil Akhir:

Angka yang didekripsi dapat dipetakan kembali ke huruf (1=A, 2=B, ..., 26=Z), dan pesan asli "SECRET" dapat diperoleh.

Ingat, contoh ini untuk tujuan pendidikan dan tidak mencakup semua tindakan keamanan dan pengoptimalan yang digunakan dalam implementasi RSA dunia nyata.

7.4.4. Contoh Enkripsi ElGamal

Tentu saja, mari kita periksa ilustrasi tambahan yang mencakup aritmatika modular dan pemanfaatannya di bidang kriptografi.

ElGamal adalah algoritma enkripsi kunci asimetris yang menggunakan sifat-sifat logaritma diskrit dalam bidang terbatas.

Pembuatan Kunci:

Pilih bilangan prima besar, p = 23.

Pilih modulo akar primitif p, g = 5.

Pilih kunci pribadi, x = 6.

Hitung kunci publik yang sesuai, y = (g^x) % p = (5^6) % 23 = 8.

Enkripsi:

Pilih nilai rahasia acak, k = 3.

Hitung kunci publik sesaat, K = (y^k) % p = (8^3) % 23 = 7.

Ubah pesan teks biasa "HELLO" menjadi nilai numerik: HELLO -> 8 5 12 12 15.

Teks sandi:

Untuk setiap nilai plaintext, hitung pasangan ciphertext (C1, C2):

C1 = (g^k) % p = (5^3) % 23 = 10.

C2 = (M * K) % p = (8 * 7) % 23 = 21.

Cipherteks Akhir:

Pesan terenkripsi menjadi rangkaian pasangan ciphertext: (10, 21) (10, 16) (10, 11) (10, 11) (10, 16).

Dekripsi:

Untuk mendekripsi setiap pasangan ciphertext (C1, C2):

Hitung nilai rahasia bersama, s = (C1^x) % p = (10^6) % 23 = 8.

Hitung invers perkalian modular dari s, s_inv = s^-1 % p = 8^-1 % 23 = 19.

Pulihkan nilai teks asli, M = (C2 * s_inv) % p = (21 * 19) % 23 = 15.

Pesan yang Didekripsi:

Petakan nilai numerik kembali ke huruf (1=A, 2=B, ..., 26=Z), dan pesan asli "HELLO" dipulihkan.

Penjelasan:

Enkripsi ElGamal bergantung pada kesulitan menghitung logaritma diskrit. Keamanan terletak pada fakta bahwa menghitung x dari y

atau K secara komputasi tidak mungkin dilakukan tanpa mengetahui k, yang berfungsi sebagai kunci sesaat.

Ingatlah bahwa penerapan kriptografi dunia nyata jauh lebih kompleks dan melibatkan langkah-langkah keamanan tambahan.

7.5 Keuntungan Dan Kerugian Matematika Diskrit Dengan Kriptografi

Bidang matematika diskrit memegang posisi penting dalam bidang kriptografi, menyajikan keuntungan dan kerugian yang mempengaruhi pemanfaatannya dalam pengembangan sistem komunikasi yang kuat. Berikut ini adalah beberapa keuntungan dan kerugian penting (Mehta, 2018):

Keuntungan:

Keamanan sistem kriptografi didasarkan pada pemanfaatan prinsip-prinsip matematika yang memiliki kompleksitas yang melekat, membuatnya tahan terhadap rekayasa balik dengan cara komputasi. Sifat rumit dari kerumitan ini berfungsi sebagai dasar keamanan dalam berbagai teknik enkripsi.

Kriptografi kunci publik adalah bidang yang mengandalkan matematika diskrit untuk mengembangkan sistem yang penting untuk memastikan pertukaran kunci yang aman dan tanda tangan digital. Teknologi ini memberikan komunikasi yang aman tanpa adanya kunci rahasia bersama (Mehta, 2018).

Tantangan matematika yang melekat, seperti faktorisasi bilangan besar atau perhitungan logaritma diskrit, saat ini dianggap menimbulkan kesulitan yang signifikan untuk komputer klasik dan kuantum dalam hal solusi yang efisien. Sifat kekerasan memainkan peran penting dalam meningkatkan daya tahan sistem kriptografi.

Konsep integritas dan otentikasi data mencakup pemanfaatan mekanisme berbasis matematika diskrit seperti fungsi hashing dan tanda tangan digital. Mekanisme ini memainkan peran penting dalam verifikasi integritas data dan otentikasi pengirim pesan.

Protokol kriptografi memanfaatkan prinsip matematika diskrit untuk membangun fondasi yang kuat untuk pengembangan sistem aman di berbagai aplikasi. Protokol-protokol ini sangat penting dalam memastikan transmisi data yang aman, proses autentikasi, dan memfasilitasi komputasi multi-pihak yang aman (Rotella dan Canteaut, 2018).

Kekurangan:

Adanya kerentanan algoritme dapat menimbulkan risiko yang signifikan terhadap keamanan sistem kriptografi, karena dapat muncul dari desain algoritme yang kurang optimal atau implementasi yang salah. Pemanfaatan fitur matematika mungkin menghadirkan keuntungan dan kerugian jika metode ini tidak dibuat dengan cermat.

Manajemen dan distribusi kunci publik menimbulkan tantangan yang signifikan meskipun kemudahan yang ditawarkan oleh kriptografi kunci publik. Jika kunci publik tidak diautentikasi secara memadai, integritas keamanan sistem dapat terancam.

Munculnya ancaman kuantum menimbulkan kekhawatiran yang signifikan terhadap keamanan sistem kriptografi saat ini yang mengandalkan matematika diskrit. Meskipun sistem ini sekarang dianggap aman terhadap komputer klasik, kemunculan komputer kuantum memperkenalkan kemungkinan pemecahan masalah matematika spesifik yang dapat membahayakan integritas sistem kriptografi ini secara efisien. Keadaan yang disebutkan di atas mengharuskan penerapan kriptografi pasca-kuantum.

Overhead Performa: Algoritma kriptografi tertentu, terutama yang bergantung pada proses matematika yang rumit, mungkin menimbulkan beban pemrosesan yang besar. Faktor tersebut di atas berpotensi mempengaruhi kecepatan dan efektivitas prosedur enkripsi dan dekripsi (Rotella dan Canteaut, 2018).

Pemanfaatan gagasan matematika yang rumit memerlukan pemahaman yang mendalam tentang matematika diskrit dan kriptografi. Pemahaman materi pelajaran yang tidak tepat dapat menimbulkan masalah implementasi, yang pada gilirannya dapat menyebabkan munculnya kerentanan.

Keterbatasan dalam Aplikasi Khusus: Aplikasi tertentu mungkin memerlukan pemanfaatan metode enkripsi yang

memprioritaskan pendekatan yang ringan dan efisien, daripada hanya mengandalkan matematika diskrit. Dalam kasus seperti ini, mungkin perlu menggunakan sintesis metodologi (Rotella dan Canteaut, 2018).

Singkatnya, matematika diskrit berfungsi sebagai dasar fundamental untuk kriptografi kontemporer dan menawarkan kerangka kerja yang tangguh untuk menjaga komunikasi digital. Namun, sangat penting untuk mengakui kelebihan dan kekurangan yang terkait dengan bidang ini. Pembangunan sistem kriptografi yang aman memerlukan desain algoritme yang cermat, peningkatan yang konsisten untuk mengatasi ancaman yang berkembang, dan pemahaman komprehensif tentang prinsip matematika yang mendasarinya (Rotella dan Canteaut, 2018).

7.6 Bagaimana Matematika Diskrit Diterapkan pada Kriptografi Simetris

Mari kita jelajahi aplikasi praktis matematika diskrit dalam konteks kriptografi simetris, disertai dengan ilustrasi grafik yang ringkas.

Ikhtisar Kriptografi Simetris: Kriptografi simetris melibatkan pemanfaatan kunci rahasia tunggal untuk proses enkripsi dan dekripsi. Pemanfaatan prinsip matematika diskrit merupakan bagian integral dari pengembangan jadwal kunci dan jaringan substitusi-permutasi (SPN) dalam metode enkripsi simetris (Rotella dan Canteaut, 2018).

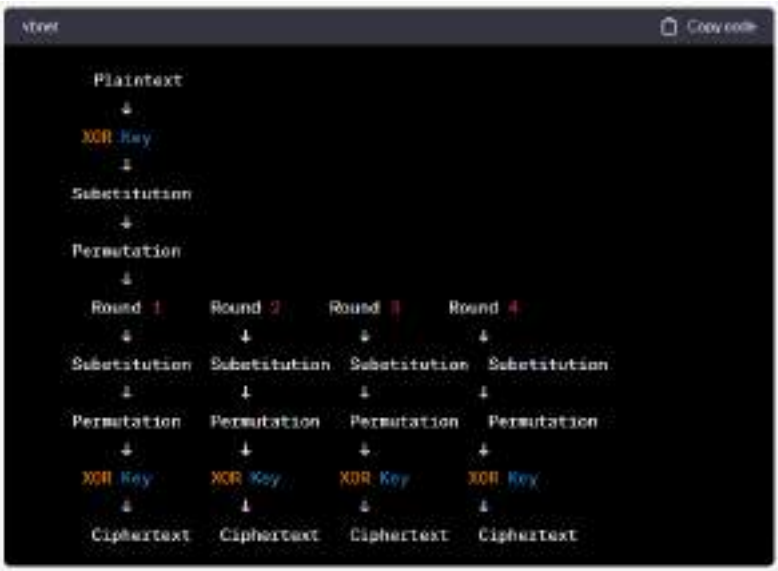
1. Pembuatan Kunci:

Pembangkitan Kunci: Bidang matematika diskrit digunakan dalam proses pembuatan kunci rahasia yang akan digunakan bersama oleh pengirim dan penerima. Kuncinya adalah urutan biner yang mengatur prosedur enkripsi dan dekripsi.

2. Jaringan Substitusi-Permutasi (SPN)

Jaringan Substitusi-Permutasi (SPN) adalah kerangka dasar yang digunakan dalam berbagai teknik enkripsi simetris. Prosesnya melibatkan serangkaian iterasi termasuk operasi substitusi dan permutasi, dengan matematika diskrit menyediakan kerangka kerja untuk mendefinisikan aturan dan fungsi yang mengatur operasi ini.

Disajikan di bawah ini adalah penggambaran grafis yang disederhanakan dari Jaringan Pergantian-Permutasi (SPN).



Gambar 7.2. representasi visual yang disederhanakan dari SPN

3. Efek Longsor:

Matematika diskrit memastikan bahwa perubahan kecil pada input (plaintext) atau kunci menghasilkan perubahan signifikan pada output (ciphertext). Properti ini, yang dikenal sebagai efek longsor, penting untuk keamanan. Sedikit perubahan baik pada plainteks atau kunci akan menghasilkan cipherteks yang berbeda secara drastis.

4. Kebingungan dan Difusi Data:

Konsep matematika diskrit seperti aritmatika modular dan operasi bitwise digunakan untuk memperkenalkan kebingungan dan difusi. Kebingungan memastikan bahwa hubungan antara plaintext dan ciphertext rumit, sehingga sulit untuk menyimpulkan kuncinya. Difusi menyebarkan pengaruh perubahan bit tunggal dalam teks biasa ke beberapa bit dalam teks sandi.

5. Jadwal Utama:

Jadwal kunci adalah proses yang menghasilkan kunci bulat dari kunci rahasia utama. Matematika diskrit digunakan untuk membuat mekanisme perluasan kunci aman yang meningkatkan keamanan algoritme.

6. Kriptanalisis:

Matematika diskrit juga memainkan peran penting dalam kriptoanalisis, yang melibatkan pemecahan sistem kriptografi.

Menganalisis sifat matematika dari algoritma enkripsi membantu mengidentifikasi kerentanan dan kelemahan (Rotella dan Canteaut, 2018).

Memvisualisasikan Kriptografi Simetris:

Meskipun sulit untuk menggambarkan seluruh kompleksitas kriptografi simetris dalam satu diagram, ilustrasi SPN memberikan gambaran tentang bagaimana konsep matematika diskrit terjalin ke dalam proses enkripsi. Langkah-langkah kunci melibatkan substitusi, permutasi, kebingungan, difusi, dan manajemen kunci yang kuat, semuanya berdasarkan prinsip matematika.

Ingat, ini adalah representasi yang disederhanakan, dan kriptografi simetris dunia nyata melibatkan detail rumit dan lapisan keamanan tambahan untuk melawan berbagai serangan.

7.7 Pemahaman Tentang Prinsip Dan Konsep Kunci.

Saat menggabungkan matematika diskrit dengan kriptografi, ada beberapa hal penting yang harus Anda ketahui untuk memastikan keamanan dan efektivitas sistem kriptografi Anda. Berikut beberapa pertimbangan utama (Rotella dan Canteaut, 2018; Jashkothari1, 2023):

1. Dasar-dasar Matematika: Memiliki pemahaman yang kuat tentang konsep matematika diskrit seperti aritmatika modular,

bilangan prima, bidang hingga, teori grup, dan teori bilangan. Konsep-konsep ini membentuk dasar dari algoritma kriptografi.

2. Desain Algoritma: Pahami cara merancang algoritme kriptografi yang memanfaatkan properti matematika diskrit sambil memastikan keamanan. Pertimbangkan faktor-faktor seperti panjang kunci, entropi, dan ketahanan algoritme terhadap serangan.
3. Asumsi Keamanan: Waspadai asumsi yang menjadi dasar keamanan kriptografi, seperti kekerasan masalah matematika tertentu (misalnya, memfaktorkan bilangan besar, logaritma diskrit). Ketahui batasan dan potensi kerentanan yang terkait dengan asumsi ini.
4. Manajemen Kunci: Manajemen kunci yang tepat sangat penting. Pahami cara membuat, menukar, dan menyimpan kunci dengan aman. Manajemen kunci yang tidak memadai dapat menyebabkan pelanggaran keamanan.
5. Bukti Keamanan: Pelajari tentang bukti kriptografi dan perannya dalam mendemonstrasikan properti keamanan algoritme. Ini melibatkan menunjukkan bahwa masalah matematika tertentu secara komputasi sulit untuk dipecahkan.
6. Teknik Cryptanalysis: Pelajari teknik cryptanalysis umum dan serangan yang digunakan untuk memecahkan sistem kriptografi. Memahami bagaimana musuh dapat

mengeksploitasi kelemahan matematika sangat penting untuk membangun sistem yang kuat.

7. Keacakan dan Entropi: Matematika diskrit sering kali melibatkan pembuatan bilangan acak. Pahami cara menghasilkan keacakan yang sebenarnya dan mengelola entropi untuk memastikan bahwa kunci kriptografi tidak dapat diprediksi.
8. Algoritma Standar: Biasakan diri Anda dengan algoritma dan protokol kriptografi yang diterima secara luas, seperti AES, RSA, Diffie-Hellman, dan ECC. Algoritme ini didasarkan pada prinsip matematika diskrit yang baik dan telah mengalami analisis ekstensif.
9. Tingkat Keamanan: Waspadai berbagai tingkat keamanan dan hubungannya dengan panjang kunci. Memahami konsep kompleksitas komputasi dan bagaimana dampaknya terhadap waktu yang diperlukan untuk memecahkan suatu algoritme.
10. Kriptografi Pasca-Kuantum: Dengan munculnya komputer kuantum, teliti algoritme kriptografi pasca-kuantum yang menahan serangan dari komputer kuantum. Bersiaplah untuk mengadaptasi sistem kriptografi Anda agar aman untuk kuantum.
11. Pertimbangan Implementasi: Implementasi dapat menimbulkan kerentanan. Pahami tantangan serangan saluran samping,

serangan waktu, dan kelemahan implementasi yang dapat melemahkan keamanan kriptografi.

12. Pembelajaran Berkelanjutan: Kriptografi dan matematika diskrit adalah bidang yang berkembang. Tetap perbarui dengan penelitian terbaru, kerentanan, dan kemajuan. Berpartisipasi dalam komunitas, konferensi, dan diskusi yang relevan.
13. Aspek Hukum dan Etika: Waspadai peraturan hukum seputar kriptografi di wilayah Anda dan tanggung jawab etis terkait dengan membangun sistem yang aman.

Ingat, meskipun matematika adalah alat yang ampuh untuk kriptografi, membangun sistem yang aman memerlukan pendekatan komprehensif yang mencakup pemahaman dasar matematika, menerapkan protokol yang aman, dan mempertimbangkan potensi serangan dan kerentanan.

DAFTAR PUSTAKA

Jashkothari1 (2023)*Kriptografi dan Jenisnya*,*geeksforgeeks.org*. Tersedia di: https://www.geeksforgeeks.org/cryptography-and-its-types/ (Diakses: 8 Agustus 2023).

Kumar, A.*et al.* (2019) 'Protokol distribusi kunci kuantum yang ditingkatkan untuk verifikasi',*Jurnal Ilmu Matematika Diskrit dan Kriptografi*, 22(4), hlm. 491–498. dua: 10.1080/09720529.2019.1637153.

Mehta, P.K. (2018)*Matematika Diskrit dalam Kriptografi*,*techiesgazette.marwadiuniversity.ac.in*. Tersedia di: http://techiesgazette.marwadiuniversity.ac.in/?p=584#:~:text=Discrete Math in Cryptography,atau bit terpisah dan berbeda. (Diakses: 7 Agustus 2023).

Richards, K. (2021)*Apa itu kriptografi?*,*techtarget.com*. Tersedia di: https://www.techtarget.com/searchsecurity/definition/cryptography (Diakses: 8 Agustus 2023).

Rotella, Y. dan Canteaut, A. (2018) 'Matematika Diskrit yang Diterapkan pada Kriptografi Simetris', hlm. 1–10.

BIODATA PENULIS

Lulut Alfaris, S.T., M.T.
Dosen Program Studi Teknologi Kelautan
Politeknik Kelautan dan Perikanan Pangandaran

Penulis menempuh pendidikan di SMAN 1 Genteng Banyuwangi, selanjutnya pendidikan S1 dan S2 pada Jurusan Teknik Kelautan, Institut Teknologi Sepuluh Nopember Surabaya . Saat ini menjadi dosen teap di Program Studi Teknologi Kelautan, Politeknik Kelautan dan Perikanan Pangandaran. Mata Kuliah yang Penulis ajar ialah Oseanografi, Matematika Teknik, Statistika dan Termodinamika. Buku-buku penulis yang telah terbit diantaranya: Artificial Intelligence; Riset Operasi; Matriks dan Ruang Vektor; Filsafat Pendidikan Matematika; Konsep Dasar Matematika; Matematika untuk Perguruan Tinggi; Termodinamika: Tinjauan Teoritis dan Praktis; Aljabar Elementer; Representasi Pengetahuan; Pengelolaan Pesisir Terpadu"

BIODATA PENULIS

Khairunnisa Fadhilla Ramdhania, S.Si., M.Si.
Dosen Program Studi Informatika, Fakultas Ilmu Komputer
Universitas Bhayangkara Jakarta Raya

Penulis lahir di Jakarta, 28 Maret 1992. Sejak tahun 2007, penulis mulai tertarik pada bidang eksakta khususnya matematika. Kemudian di tahun 2010 penulis diterima sebagai mahasiswa Jurusan Matematika Universitas Padjadjaran dan menekuni bidang matematika analisis khususnya tentang pecahan atau fraksional. Penulis berhasil lulus pada tahun 2014, lalu mengajar beberapa anak sekolah secara berkelompok untuk belajar matematika. Dari sana penulis memiliki ketertarikan dalam pengajaran dan berkeinginan menjadi dosen sehingga memutuskan untuk melanjutkan studi S2 di Prodi Matematika Institut Teknologi Bandung dan menyelesaikannya dalam waktu 2 tahun. Pada tahun 2018 penulis resmi menjadi bagian dari Universitas Bhayangkara Jakarta Raya di Program Studi Informatika, yakni sebagai dosen pengampu bidang matematika yang menjadi ilmu dasar yang harus dikuasai oleh mahasiswa. Menjadi seorang penulis buku merupakan cita-cita yang sejak lama didambakan, dan ini adalah buku ketiga penulis. Harapannya, penulis dapat konsisten berkontribusi menghasilkan buku-buku lain yang menarik dan mudah dipahami oleh pembaca.

BIODATA PENULIS

Mindo H.Sinambela, S.Pd., M.Pd

Dosen Program Studi Pendidikan Matematika

STKIP Abdi Wacana Wamena

Penulis lahir di Pematang Siantar tanggal 26 Oktober 1985. Penulis adalah dosen tetap pada Program Studi Pendidikan Matematika di STKIP Abdi Wacana Wamena. Menyelesaikan pendidikan S1 Pendidikan Matematika di STKIP Abdi Wacana Wamena dan melanjutkan S2 pada jurusan Pendidikan Matematika di Universitas Cenderawasih. Sebelum menjadi dosen penulis pernah bekerja sebagai tenaga adminstrasi disalah satu perusahaan di Kabupaten Jayawijaya.

BIODATA PENULIS

Staf Dosen Program Studi Pendidikan Matematika

Ida Fitriana Ambarsari, S.Pd., M.Si Lahir di Situbondo Kec. Panji Jawa Timur, pada tanggal 18 Maret 1994, putri pertama dari pasangan ayah **Dede Yusmana (Alm)** dan ibu **Sri Hartatik**. Tahun 2000 – 2006 menempuh pendidikan Sekolah Dasar di SDN 6 Mimbaan, Situbondo. Tahun 2006 – 2009 menempuh pendidikan Sekolah Menengah Pertama di SMPN 1 Situbondo, Situbondo. Tahun 2009 – 2012 menempuh pendidikan Sekolah Menengah Atas menempuh pendidikan di SMAN 2 Situbondo, Situbondo. Dan tahun 2012 – 2016 menempuh Pendidikan Tinggi di Universitas Negeri Malang, Fakultas Matematika dan Ilmu Pengetahuan Alam, Jurusan Matematika, Program Studi Pendidikan Matematika. Lulus pada bulan April 2016 dengan gelar sarjana pendidikan (S.Pd). Kemudian pada awal tahun 2017 bekerja di PT BTPN Syariah selama 6 bulan. Setelah itu, melanjutkan Pendidikan Tinggi di Universitas Negeri Malang, Fakultas Matematika dan Ilmu Pengetahuan Alam, Jurusan Matematika, Program Studi S2 Matematika. Lulus pada bulan Oktober 2020 dengan gelar magister sains (M.Si).

Karir atau profesi staf pengajar (dosen) dimulai sejak tahun 2021 di STKIP PGRI Situbondo dan sekarang tinggal di Situbondo.

Saat berstatus mahasiswa pascasarjana di Universitas Negeri Malang, aktif mengikuti seminar dan konferensi prosiding internasional

dan mampu memublikasikan tiga artikel terindeks Scopus. Selain itu, memiliki pengalaman mengikuti kegiatan *mobility programme* di Universiti Malaya, Malaysia, serta aktif berpartisipasi mengikuti kegiatan pengabdian masyarakat yang diselenggarakan oleh Universitas Negeri Malang.

BIODATA PENULIS

Suwito Pomalingo, S.Kom., M.Kom
Dosen Program Studi Informatika
Universitas Multimedia Nusantara

Penulis adalah dosen tetap pada Program Studi Informatika Fakultas Teknik dan Informatika Universitas Multimedia Nusantara. Menyelesaikan pendidikan S1 pada Jurusan Teknik Informatika dan melanjutkan S2 pada Program Magister Informatika Universitas Islam Indonesia.

Saat ini sementara menyelesaikan studi Doktoral dengan topik penelitian di bidang Security Science dengan pendekatan Deep Learning. Penulis menekuni bidang keilmuan meliputi Cyber Security, Malware Analytics, Data Science, Machine Learning, dan Deep Learning, selain itu, penulis juga memiliki beberapa serifikasi internasional di bidang Cyber Security, Digital Forensics, Data Science dan Machine Learning.

BIODATA PENULIS

Indah Resti Ayuni Suri, S.Si., M.Si.
Dosen Program Studi Pendidikan Matematika
Fakultas Tarbiyah dan Keguruan UIN Raden Intan Lampung

Penulis lahir di Kotagajah Lampung Tengah tanggal 30 Maret 1988. Penulis adalah dosen program studi pendidikan matematika fakultas tarbiyah dan keguruan UIN Raden Intan Lampung. Menyelesaikan pendidikan S1 di UIN Malang pada Jurusan Matematika dan melanjutkan S2 di UB Malang pada Jurusan Matematika. Penulis menekuni bidang Menulis.

BIODATA PENULIS

Wirawan Istiono, S.Kom., M.Kom.
Dosen Informatika
Fakultas Teknik dan Informatika, Universitas Multimedia Nusantara

Penulis lahir di Jakarta tanggal 13 April 1983. Penulis adalah dosen pada Program Studi Informatika Fakultas Teknik dan Informatika, Universitas Multimedia Nusantara. Menyelesaikan pendidikan S1 pada Jurusan Teknik Informatika dan melanjutkan S2 pada bidang jurusan yang Teknik Informatika. Pada saat ini, penulis menekuni bidang Penelitian terkait komputer sains, sistem komputer dan teknologi informasi