

RINALDI MUNIR

Buku Teks
Ilmu
Komputer



MATEMATIKA DISKRIT

pak
TEGUH

an | Matriks | Relasi | Pengolahan | Algoritma | Kombinatorial | Peluang Diskrit

Revisi Keempat



Penerbit INFORMATIKA

<http://www.pakteguh.com>

MATEMATIKA DISKRIT

Edisi 3

Rinaldi Munir

**DENGAN MEMBACA KITA MENDAPAT ILMU
UNTUK MENJADI PRAJURIT PROFESIONAL**



Penerbit INFORMATIKA Bandung

MATEMATIKA DISKRIT

Penyusun	: RINALDI MUNIR Dosen Program Studi Teknik Informatika Sekolah Teknik Elektro dan Informatika Institut Teknologi Bandung
Penerbit	: Informatika Bandung
Pemasaran	: BI-OBSES Pasar Buku Palasari No. 82 Bandung 40264 Telp. (022) 7317812 Fax. (022) 7317896
Cetakan Keempat	: September 2010 (Revisi Keempat)
ISBN	: 978-602-8758-07-9

Copyright © 2010 pada Penerbit INFORMATIKA Bandung

Kata Pengantar Edisi Ketiga

Rebut hari ini, dan jangan begitu saja mempercayai hari esok.
(Horace)

Buku *Matematika Diskrit (Edisi Ketiga)* disusun sebagai buku teks mahasiswa yang mengambil mata kuliah Matematika Diskrit. Matematika Diskrit merupakan mata kuliah yang fundamental dalam pendidikan ilmu komputer atau Teknik Informatika. Bahkan, saat ini Matematika Diskrit merupakan mata kuliah wajib pada program pendidikan yang termasuk ke dalam kelompok teknologi informasi (Ilmu Komputer, Teknik Informatika, Sistem Informasi, dan Teknik Elektro).

Bahan kuliah Matematika Diskrit yang disajikan di dalam buku ini meliputi: logika, himpunan, matriks, relasi dan fungsi, induksi matematik, algoritma, bilangan bulat (*integer*), kombinatorial dan peluang diskrit, aljabar Boolean, graf, pohon, dan kompleksitas algoritma.

Sebenarnya, masih ada beberapa materi Matematika Diskrit yang tidak dicakup di dalam buku ini, seperti Barisan dan Deret, Otomata, Teori Bahasa, Fungsi Pembangkit, serta Grup dan Ring. Namun, beberapa materi tersebut diberikan pada ~~mata~~ kuliah yang lain, sehingga pembahasannya dapat menjadi lebih menyeluruh ~~dari~~ mendalam.

Sebagai mata kuliah yang diberikan pada tahap Sarjana Muda, maka mata kuliah Matematika Diskrit termasuk ke dalam mata kuliah dasar yang diperlukan sebagai fondasi mata kuliah tingkat lanjut. Buku ini memberikan contoh-contoh ilustrasi penggunaan materi matematika diskrit pada bidang-bidang lain di Teknik Informatika.

Materi kuliah Matematika Diskrit yang dirangkum di dalam buku ini diambil dari berbagai pustaka yang judul-judulnya dicantumkan pada halaman Daftar Pustaka.

Beberapa buku penting yang penulis gunakan dalam menyusun diktat ini adalah [ROS99], [LIU85], dan [JOH97]. Buku-buku lainnya merupakan buku pelengkap.

Pada akhir setiap bab, selalu diberikan soal-soal latihan untuk dikerjakan di rumah. Makin banyak mengerjakan soal latihan, niscaya akan semakin baik pemahamannya pada materi matematika diskrit.

Buku ini pada mulanya adalah diktat kuliah di Departemen Teknik Informatika ITB. Berhubung mahasiswa di luar Teknik Informatika ITB maupun mahasiswa di luar ITB banyak juga yang menggunakan diktat tersebut, maka Penyusun menerbitkannya menjadi sebuah buku agar cakupan pengguna buku ini dapat lebih luas lagi.

Buku ini merupakan edisi yang ketiga. Di dalam edisi ketiga ini, terdapat beberapa materi tambahan sebagai berikut:

1. **Bab 1 Logika:** materi baru yang ditambahkan adalah metode inferensi, perbedaan antara aksioma, *lemma*, teorema, dan *corolarry*, serta argumen dan pembuktian validitasnya.
2. **Bab 2 Himpunan:** materi baru yang ditambahkan adalah himpunan dan logika *fuzzy*. Materi baru ini hanya bersifat opsional dan cukup diketahui sebagai pengetahuan saja.
3. **Bab 3 Matriks, Relasi, dan Fungsi:** materi baru yang ditambahkan adalah klosur relasi (*closure of relation*), relasi kesetaraan (*equivalence relation*), dan relasi pengurutan parsial (*partial ordering relation*).
4. **Bab 5 Algoritma dan Bilangan Bulat:** materi baru yang ditambahkan adalah pembangkit bilangan acak semu (*pseudo-random generator*).
5. **Bab 6 Kombinatorail dan Peluang Diskrit:** materi baru yang ditambahkan prinsip sarang burung merpati (*pigeonhole principle*).

Selain penambahan materi baru, pada setiap bab ditambahkan upabab baru yaitu **Ragam Soal dan Penyelesaian**. Upabab ini berisi contoh-contoh soal tambahan dan penyelesaian.

Penyusun menyadari bahwa buku ini jauh dari sempurna, bahkan masih mungkin terdapat kesalahan-kesalahan. Koreksi, saran perbaikan, kritik membangun, dsb, dapat dilayangkan ke alamat *e-mail* penyusun: rinaldi@informatika.org.

Bandung, Juli 2005

Ir. Rinaldi Munir, M.T.

Daftar Isi

Kata Pengantar	iii
Daftar Isi	v
Apakah Matematika Diskrit Itu ?	xi
1. Logika	1
1.1 Proposisi	2
1.2 Mengkombinasikan Proposisi	4
1.3 Tabel kebenaran	6
1.4 Disjungsi Eksklusif	9
1.5 Hukum-hukum Logika Proposisi	10
1.6 Operasi Logika di dalam Komputer	12
1.7 Proposisi Bersyarat (Implikasi)	15
1.8 Varian Proposisi Bersyarat	21
1.9 Bikondisional (Bi-implikasi).....	22
1.10 Inferensi	26
1.11 Argumen	30
1.12 Aksioma, Teorema, <i>Lemma</i> , dan <i>Colollary</i>	35
1.13 Ragam Contoh Soal dan Penyelesaian	35
Soal Latihan	42

2. Himpunan	47
2.1 Definisi Himpunan	48
2.2 Penyajian Himpunan	48
2.3 Kardinalitas	53
2.4 Himpunan Kosong	54
2.5 Himpunan Bagian (<i>Subset</i>)	54
2.6 Himpunan yang Sama	57
2.7 Himpunan yang Ekivalen	57
2.8 Himpunan Saling Lepas	58
2.9 Himpunan Kuasa	59
2.10 Operasi Terhadap Himpunan	60
2.11 Perampatan Operasi Himpunan.....	66
2.12 Hukum-hukum Himpunan	67
2.13 Prinsip Dualitas	68
2.14 Prinsip Inklusi-Eksklusi	70
2.15 Partisi	72
2.16 Pembuktian Proposisi Himpunan	73
2.17 Himpunan Ganda	78
2.18 Tipe <i>Set</i> dalam Bahasa Pascal	79
2.19 Pengantar Logika dan Himpunan <i>Fuzzy</i>	80
2.20 Ragam Contoh Soal dan Penyelesaian	87
Soal Latihan	92
3. Matriks, Relasi dan Fungsi	97
3.1 Matriks	98
3.2 Relasi	103
3.3 Representasi Relasi	105
3.3.1 Representasi Relasi dengan Tabel	105
3.3.2 Representasi Relasi dengan Matriks	106
3.3.3 Representasi Relasi dengan Graf Berarah	107
3.4 Relasi Inversi.....	108
3.5 Mengkombinasikan Relasi.....	109
3.6 Komposisi Relasi	110
3.7 Sifat-sifat Relasi	112
3.8 Relasi Kesetaraan	118
3.9 Relasi Pengurutan Parsial.....	119

3.10	Klosur Relasi	120
3.11	Relasi n -ary	124
3.12	Fungsi.....	128
3.13	Fungsi Inversi	134
3.14	Komposisi Fungsi	135
3.15	Beberapa Fungsi Khusus	136
3.16	Fungsi Rekursif	138
3.17	Ragam Soal dan Penyelesaian	141
	Soal Latihan.....	144
4.	Induksi Matematik	149
4.1	Pernyataan Perihal Bilangan Bulat	150
4.2	Prinsip Induksi Sederhana	151
4.3	Prinsip Induksi yang Dirampatkan	156
4.4	Prinsip Induksi Kuat	160
4.5	Bentuk Induksi Secara Umum	163
4.6	Ragam Soal dan Penyelesaian	165
	Soal Latihan	170
5.	Algoritma dan Bilangan Bulat	175
5.1	Algoritma	176
5.2	Notasi untuk Algoritma	177
5.3	Beberapa Contoh Algoritma	180
5.4	Bilangan Bulat.....	183
5.5	Sifat Pembagian pada Bilangan Bulat.....	183
5.6	Pembagi Bersama Terbesar	185
5.7	Algoritma Euclidean	187
5.8	Aritmetika Modulo	191
5.9	Bilangan Prima.....	200
5.10	Kriptografi	203
5.11	Fungsi <i>Hash</i>	214
5.12	<i>International Standard Book Number</i> (ISBN)	215
5.13	Pembangkit Bilangan Acak Semu	217
5.14	Ragam Soal dan Penyelesaian	218
	Soal Latihan	222

6. Kombinatorial dan Peluang Diksrit.....	225
6.1 Percobaan	226
6.2 Kaidah Dasar Menghitung	227
6.3 Perluasan Kaidah Menghitung	230
6.4 Prinsip Inklusi-Eksklusi	236
6.5 Permutasi	236
6.6 Kombinasi	244
6.7 Permutasi dan Kombinasi Bentuk Umum	251
6.8 Kombinasi dengan Pengulangan	254
6.9 Koefisien Binomial	256
6.10 Prinsip Sarang Merpati	258
6.11 Peluang Diskrit	260
6.12 Ragam Soal dan Penyelesaian.....	268
Soal Latihan	277
7. Aljabar Boolean	281
7.1 Definisi Aljabar Boolean	282
7.2 Aljabar Boolean Dua-Nilai	285
7.3 Ekspresi Boolean	286
7.4 Prinsip Dualitas	288
7.5 Hukum-hukum Aljabar Boolean	289
7.6 Fungsi Boolean	293
7.7 Penjumlahan dan Perkalian Dua Fungsi	295
7.8 Komplemen Fungsi Boolean.....	296
7.9 Bentuk Kanonik	298
7.10 Konversi Antar Bentuk Kanonik	303
7.11 Bentuk Baku	304
7.12 Aplikasi Aljabar Boolean	305
7.12.1 Jaringan Pensaklaran (<i>Switching Network</i>)	305
7.12.2 Sirkuit Elektronik	306
7.13 Penyederhanaan Fungsi Boolean	308
7.13.1 Penyederhanaan Fungsi Boolean Secara Aljabar	309
7.13.2 Metode Peta Karnaugh	310
7.13.3 Contoh-contoh Penyederhanaan Fungsi Boolean	324
7.13.4 Peta Karnaugh untuk Lima Peubah	329
7.13.5 Keadaan <i>Don't Care</i>	330
7.14 Penyederhanaan Rangkaian Logika	333

7.15 Metode Quine-McCluskey	334
7.16 Ragam Soal dan Penyelesaian	338
Soal Latihan	348
8. Graf	353
8.1 Sejarah Graf	354
8.2 Definisi Graf	356
8.3 Jenis-jenis Graf	357
8.4 Contoh Terapan Graf	359
8.5 Terminologi Dasar	364
8.6 Beberapa Graf Sederhana Khusus	377
8.7 Representasi Graf	381
8.8 Graf Isomorfik (<i>Isomorphic Graph</i>)	386
8.9 Graf Planar dan Graf Bidang.....	392
8.10 Graf Dual (<i>Dual Graph</i>)	402
8.11 Lintasan dan Sirkuit Euler	404
8.12 Lintasan dan Sirkuit Hamilton	408
8.13 Lintasan Terpendek (<i>Shortest Path</i>)	412
8.14 Persoalan Pedagang Keliling	421
8.15 Persoalan Tukang Pos Cina	424
8.16 Pewarnaan Graf	425
8.17 Ragam Soal dan Penyelesaian	430
Soal Latihan.....	437
9. Pohon	443
9.1 Definisi Pohon	444
9.2 Sifat-sifat Pohon	446
9.3 Pewarnaan Pohon	447
9.4 Pohon Merentang	447
9.5 Pohon Berakar	457
9.6 Terminologi Pada Pohon Berakar	458
9.7 Pohon Berakar Terurut	461
9.8 Pohon <i>m</i> -ary	463
9.9 Pohon Biner	467
9.10 Pohon Ekspresi	469
9.11 Pohon Keputusan	475

9.12 Kode Awalan	476
9.13 Kode Huffman	477
9.14 Pohon Pencarian	481
9.15 Traversal Pohon Biner	483
9.16 Ragam Soal dan Pembahasan	487
Soal Latihan.....	491
10. Kompleksitas Algoritma.....	495
10.1 Kemangkusan Algoritma	496
10.2 Mengapa Kita Memerlukan Algoritma yang Mangkus?	496
10.3 Kebutuhan Waktu dan Ruang.....	498
10.4 Kompleksitas Waktu dan Ruang	499
10.5 Kompleksitas Waktu Asimptotik	510
10.6 Ragam Soal dan Penyelesaian	538
Soal Latihan	541
Daftar Pustaka	457

Apakah Matematika Diskrit Itu ?

Matematika diskrit (*discrete mathematics* atau *finite mathematics*) adalah cabang matematika yang mengkaji objek-objek diskrit. Menurut kamus ensiklopedi wikipedia (http://en.wikipedia.org/wiki/Computer_science), ACM (*Association for Computing Machinery*) mendefinisikan matematika diskrit sebagai berikut:

Discrete mathematics, sometimes called finite mathematics, is the study of mathematical structures that are fundamentally discrete, in the sense of not supporting or requiring the notion of continuity. Most, if not all, of the objects studied in finite mathematics are countable sets, such as the integers.

Apa yang dimaksud dengan kata **diskrit** (*discrete*)? Benda disebut diskrit jika ia terdiri dari sejumlah berhingga elemen yang berbeda atau elemen-elemen yang tidak bersambungan. Himpunan bilangan bulat (*integer*) dipandang sebagai objek diskrit. Kita dapat memahami diskrit dengan membandingkan lawan katanya yaitu **kontinyu** atau **menerus** (*continuous*). Himpunan bilangan riil (*real*) dipandang sebagai objek kontinyu. Di dalam matematika kita mengenal fungsi diskrit dan fungsi kontinyu. Fungsi diskrit digambarkan sebagai sekumpulan titik-titik, sedangkan fungsi kontinyu digambarkan sebagai kurva.

Matematika diskrit berkembang sangat pesat dalam dekade terakhir ini. Salah satu alasan yang menyebabkan perkembangan pesat itu adalah karena komputer digital bekerja secara diskrit. Informasi yang disimpan dan dimanipulasi oleh komputer adalah dalam bentuk diskrit.

Matematika diskrit merupakan ilmu paling dasar di dalam pendidikan informatika atau ilmu komputer. Pada dasarnya informatika adalah kumpulan disiplin ilmu dan teknik yang mengolah dan memanipulasi objek diskrit. Matematika diskrit memberikan landasan matematis untuk kuliah-kuliah lain di informatika. Mahasiswa yang mengambil kuliah seperti algoritma, struktur data, basis data, otomata dan teori bahasa formal, jaringan komputer, keamanan komputer, sistem operasi, teknik kompilasi, dan sebagainya akan menemui kesulitan jika tidak mempunyai landasan matematis dari matematika diskrit. Hal ini tidak mengherankan karena kebanyakan kuliah tersebut sering mengacu kepada konsep-konsep di dalam matematika diskrit. Karena itulah kuliah matematika diskrit diberikan pada tahun pertama perkuliahan informatika atau ilmu komputer.

Di dalam kuliah Matematika Diskrit, materi matematika yang diberikan adalah matematika yang khas informatika, sehingga kadang-kadang kuliah ini dinamakan juga **Matematika Informatika**. Adapun materi-materi yang termasuk dalam Matematika Diskrit adalah:

1. Logika
2. Teori Himpunan
3. Matriks
4. Relasi dan Fungsi
5. Induksi Matematik
6. Algoritma
7. Teori Bilangan Bulat

8. Barisan dan Deret
9. Teori Grup dan *Ring*
10. Aljabar Boolean
11. Kombinatorial
12. Teori Peluang Diskrit
13. Fungsi Pembangkit dan Analisis Rekurens
14. Teori Graf (termasuk pohon)
15. Kompleksitas Algoritma
16. Pemodelan Komputasi (Otomata dan Teori Bahasa Formal)

Namun, tidak semua materi di atas akan dicakup di dalam buku ini. Beberapa materi sudah menjadi kuliah tersendiri, seperti materi algoritma dipelajari secara lebih mendalam pada kuliah Algoritma dan Pemrograman, teori peluang diskrit pada kuliah Probabilitas dan Statistik, pemodelan komputasi pada kuliah Otomata dan Teori Bahasa Formal. Materi Fungsi Pembangkit dimasukkan ke dalam kuliah Model dan Simulasi, sedangkan barisan dan deret biasanya dimasukkan ke dalam kuliah kalkulus.

Contoh-contoh persoalan dalam kehidupan sehari-hari yang diselesaikan dengan matematika diskrit antara lain:

1. Berapa banyak kemungkinan jumlah *password* yang dapat dibuat dari 8 karakter?
2. Jumlah permintaan sambungan telepon di sebuah area diperkirakan sebanyak 500.000 buah. Jika nomor telepon di area tersebut adalah 8-angka, yaitu dengan format xxx-xxxx, yang dalam hal ini 3 angka pertama menyatakan kode area (angka pertama di dalam kode area tidak boleh 0, dua angka lainnya adalah angka 0 sampai 9), apakah nomor telepon 8-angka bisa mengakomodasi semua permintaan sambungan telepon? Setiap nomor telepon harus unik.
3. Bagaimana nomor *ISBN* sebuah buku divalidasi?
4. Berapa banyak *string* biner yang panjangnya 8 bit yang mempunyai bit 1 sejumlah ganjil?
5. Bagaimana menentukan lintasan terpendek dari satu kota *a* ke kota *b*?
6. Buktikan bahwa perangko senilai n ($n \geq 8$) rupiah dapat menggunakan hanya perangko 3 rupiah dan 5 rupiah saja
7. Diberikan dua buah algoritma yang berbeda untuk menyelesaian sebuah persoalan, bagaimana menentukan algoritma mana yang terbaik?
8. Bagaimana rangkaian logika untuk membuat peraga digital yang disusun oleh 7 buah batang (*bar*)?
9. Dapatkah kita melalui semua jalan di sebuah kompleks perubahan tepat hanya sekali dan kembali lagi ke tempat semula?
10. "Makanan murah tidak enak", "makanan enak tidak murah". Apakah kedua pernyataan tersebut menyatakan hal yang sama?

Contoh-contoh persoalan di atas memperlihatkan bahwa matematika diskrit digunakan bilamana kita menghitung objek-objek, mengkaji relasi antara objek-objek dikaji, dan langkah-langkah di dalam proses dianalisis [ROS03].

Moral dari semua cerita di atas adalah bahwa mahasiswa informatika atau ilmu komputer harus memiliki pemahaman yang kuat dalam matematika diskrit, agar mereka tidak mendapat kesulitan dalam memahami kuliah-kuliah lainnya di informatika.

Perjalanan satu mil dimulai dari satu langkah.

BAB 1

Logika

Benteng kehidupan yang terkuat adalah kebenaran.
(Anonim)

Materi Matematika Diskrit di dalam buku ini dimulai dari pokok bahasan logika. Logika merupakan studi penalaran (*reasoning*). Dalam Kamus Besar Bahasa Indonesia disebutkan definisi penalaran, yaitu cara berpikir dengan mengembangkan sesuatu berdasarkan akal budi dan bukan dengan perasaan atau pengalaman. Pelajaran logika difokuskan pada hubungan antara pernyataan-pernyataan (*statements*). Tinjau argumen berikut:

- Semua pengendara sepeda motor memakai helm.
- Setiap orang yang memakai helm adalah mahasiswa.
- Jadi, semua pengendara sepeda motor adalah mahasiswa.

Meskipun logika tidak membantu menentukan apakah pernyataan-pernyataan tersebut benar atau salah, tetapi jika kedua pernyataan tersebut benar, maka penalaran dengan menggunakan logika membawa kita pada kesimpulan bahwa pernyataan

Semua pengendara sepeda motor adalah mahasiswa

juga benar.

Di dalam matematika, hukum-hukum logika menspesifikasikan makna dari pernyataan matematis. Hukum-hukum logika tersebut membantu kita untuk membedakan antara argumen yang valid dan tidak valid. Logika juga digunakan untuk membuktikan teorema-teorema di dalam matematika.

Logika pertama kali dikembangkan oleh filosof Yunani, Aristoteles, sekitar 2300 tahun yang lalu. Saat ini, logika mempunyai aplikasi yang luas di dalam ilmu komputer, misalnya dalam bidang pemrograman, analisis kebenaran algoritma, kecerdasan buatan (*artificial intelligence*), perancangan komputer, dan sebagainya.

Bab 1 ini dimulai dengan definisi proposisi dan notasi yang digunakan untuk melambangkan proposisi. Selanjutnya dijelaskan pula cara mengkombinasikan proposisi majemuk dan membentuk tabel kebenarannya. Proposisi majemuk yang lain seperti implikasi dan bi-implikasi dibahas pada bagian akhir buku.

1.1 Proposisi

Di dalam matematika, tidak semua kalimat berhubungan dengan logika. Hanya kalimat yang bernilai benar atau salah saja yang digunakan dalam penalaran. Kalimat tersebut dinamakan **proposisi** (*preposition*).

DEFINISI 1.1. Proposisi adalah kalimat deklaratif yang bernilai benar (*true*) atau salah (*false*), tetapi tidak dapat sekaligus keduanya. Kebenaran atau kesalahan dari sebuah kalimat disebut nilai kebenarannya (*truth value*).

Tiga buah contoh berikut ini dapat mengilustrasikan kalimat mana yang merupakan proposisi dan mana yang bukan.

Contoh 1.1

Kalimat-kalimat berikut ini,

- (a) 6 adalah bilangan genap.
- (b) Soekarno adalah Presiden Indonesia yang pertama.
- (c) $2 + 2 = 4$.
- (d) Ibukota Provinsi Jawa Barat adalah Semarang.
- (e) $12 \geq 19$.
- (f) Kemarin hari hujan.
- (g) Suhu di permukaan laut adalah 21 derajat Celcius.
- (h) Pemuda itu tinggi.
- (i) Kehidupan hanya ada di planet Bumi.

Semuanya merupakan proposisi. Proposisi a, b, dan c bernilai benar, tetapi proposisi d salah karena ibukota Jawa Barat seharusnya adalah Bandung dan proposisi e bernilai

salah karena seharusnya $12 \leq 19$. Proposisi f sampai i memang tidak dapat langsung ditetapkan kebenarannya, namun satu hal yang pasti, proposisi-proposisi tersebut tidak mungkin benar dan salah sekaligus. Kita bisa menetapkan nilai proposisi tersebut benar atau salah. Misalnya, proposisi f bisa kita andaikan benar (hari kemarin memang hujan) atau salah (hari kemarin tidak hujan). Demikian pula halnya untuk proposisi g dan h. Proposisi i bisa benar atau salah, karena sampai saat ini belum ada ilmuwan yang dapat memastikan kebenarannya. ■

Contoh 1.2.

Kalimat-kalimat berikut ini,

- (a) Jam berapa kereta api Argo Bromo tiba di Gambir?
- (b) Serahkan uangmu sekarang!
- (c) $x + 3 = 8$.
- (d) $x > 3$.

bukan proposisi. Kalimat a adalah kalimat tanya, sedangkan kalimat b adalah kalimat perintah, keduanya tidak mempunyai nilai kebenaran. Dari Contoh 1.1, dan 1.2 di atas, kita dapat menyimpulkan bahwa proposisi selalu dinyatakan sebagai kalimat berita, bukan sebagai kalimat tanya maupun kalimat perintah. Kalimat c dan d bukan proposisi karena kedua kalimat tersebut tidak dapat ditentukan benar maupun salah sebab keduanya mengandung peubah (variabel) yang tidak dispesifikasikan nilainya. Tetapi kalimat

“Untuk sembarang bilangan bulat $n \geq 0$, maka $2n$ adalah bilangan genap”

adalah proposisi yang bernilai benar karena kalimat tersebut merupakan cara lain untuk menyatakan bilangan genap. Begitu juga kalimat

$$x + y = y + x \text{ untuk setiap } x \text{ dan } y \text{ bilangan riil}$$

adalah proposisi karena kalimat tersebut merupakan cara lain untuk menyatakan hukum komutatif penjumlahan pada sistem bilangan riil. Dalam hal ini x dan y tidak perlu diberi suatu nilai sebab proposisi tersebut pasti benar untuk x dan y berapa saja. ■

Bidang logika yang membahas proposisi dinamakan **kalkulus proposisi** (*propositional calculus*) atau **logika proposisi** (*propositional logic*), sedangkan bidang logika yang membentuk proposisi pada pernyataan yang mengandung peubah seperti pada Contoh 1.2 c dan d di atas dibahas pada logika **kalkulus predikat** yang mana di luar cakupan buku ini.

Secara simbolik, proposisi biasanya dilambangkan dengan huruf kecil seperti p , q , r , Misalnya,

p : 6 adalah bilangan genap.

untuk mendefinisikan p sebagai proposisi “6 adalah bilangan genap”. Begitu juga untuk

q : Soekarno adalah Presiden Indonesia yang pertama.

r : $2 + 2 = 4$.

dan sebagainya.

1.2 Mengkombinasikan Proposisi

Kita dapat membentuk proposisi baru dengan cara mengkombinasikan satu atau lebih proposisi. Operator yang digunakan untuk mengkombinasikan proposisi disebut **operator logika**. Operator logika dasar yang digunakan adalah **dan (and)**, **atau (or)**, dan **tidak (not)**. Dua operator pertama dinamakan operator **biner** karena operator tersebut mengoperasikan dua buah proposisi, sedangkan operator ketiga dinamakan operator **uner** karena ia hanya membutuhkan satu buah proposisi.

Proposisi baru yang diperoleh dari pengkombinasian tersebut dinamakan **proposisi majemuk (compound proposition)**. Proposisi yang bukan merupakan kombinasi proposisi lain disebut **proposisi atomik**. Dengan kata lain, proposisi majemuk disusun dari proposisi-proposisi atomik. Metode pengkombinasian proposisi dibahas oleh matematikawan Inggris yang bernama George Boole pada tahun 1854 di dalam bukunya yang terkenal, *The Laws of Thought*. Proposisi majemuk ada tiga macam, yaitu konjungsi, disjungsi, dan ingkaran. Ketiganya didefinisikan sebagai berikut:

DEFINISI 1.2. Misalkan p dan q adalah proposisi. **Konjungsi (conjunction)** p dan q , dinyatakan dengan notasi $p \wedge q$, adalah proposisi

p dan q

Disjungsi (disjunction) p dan q , dinyatakan dengan notasi $p \vee q$, adalah proposisi

p atau q

Ingkaran atau (negation) dari p , dinyatakan dengan notasi $\sim p$, adalah proposisi

tidak p

Catatan:

1. Beberapa literatur menggunakan notasi “ $\neg p$ ”, “ \overline{p} ”, atau “**not p** ” untuk menyatakan ingkaran.
2. Kata “tidak” dapat dituliskan di tengah pernyataan. Jika kata “tidak” diberikan di awal pernyataan maka ia biasanya disambungkan dengan kata “benar” menjadi “tidak benar”. Kata “tidak” dapat juga diganti dengan “bukan” bergantung pada rasa bahasa yang tepat untuk pernyataan tersebut.

Berikut contoh-contoh proposisi majemuk dan notasi simboliknya. Ekspresi proposisi majemuk dalam notasi simbolik disebut juga **ekspresi logika**.

Contoh 1.3

Diketahui proposisi-proposisi berikut:

- p : Hari ini hujan
 q : Murid-murid diliburkan dari sekolah

maka

- $p \wedge q$: Hari ini hujan dan murid-murid diliburkan dari sekolah
 $p \vee q$: Hari ini hujan atau murid-murid diliburkan dari sekolah
 $\sim p$: Tidak benar hari ini hujan
(atau dalam kalimat lain yang lebih lazim: Hari ini *tidak* hujan)
-

Contoh 1.4

Diketahui proposisi-proposisi berikut:

- p : Hari ini hujan
 q : Hari ini dingin

maka

- $q \vee \sim p$: Hari ini dingin atau hari ini tidak hujan
atau, dengan kata lain, "Hari ini dingin atau tidak hujan"
 $\sim p \wedge \sim q$: Hari ini tidak hujan dan hari ini tidak dingin
atau, dengan kata lain, "Hari ini tidak hujan maupun dingin"
 $\sim(\sim p)$: Tidak benar hari ini tidak hujan
atau dengan kata lain, "Salah bahwa hari ini tidak hujan"
-

Contoh 1.5

Diketahui proposisi-proposisi berikut:

- p : Pemuda itu tinggi
 q : Pemuda itu tampan

Nyatakan proposisi berikut (asumsikan "Pemuda itu pendek" berarti "Pemuda itu tidak tinggi") ke dalam ekspresi logika (notasi simbolik):

- Pemuda itu tinggi dan tampan
 - Pemuda itu tinggi tapi tidak tampan
 - Pemuda itu tidak tinggi maupun tampan
 - Tidak benar bahwa pemuda itu pendek atau tidak tampan
 - Pemuda itu tinggi, atau pendek dan tampan
 - Tidak benar bahwa pemuda itu pendek maupun tampan
-

Penyelesaian:

- (a) $p \wedge q$
 - (b) $p \wedge \neg q$ (catatan: kata “tetapi” bermakna sama dengan “dan”)
 - (c) $\neg p \wedge \neg q$
 - (d) $\neg(\neg p \vee \neg q)$
 - (e) $p \vee (\neg p \wedge q)$
 - (f) $\neg(\neg p \wedge \neg q)$
-

1.3 Tabel Kebenaran

Nilai kebenaran dari proposisi majemuk ditentukan oleh nilai kebenaran dari proposisi atomiknya dan cara mereka dihubungkan oleh operator logika.

DEFINISI 1.3 Misalkan p dan q adalah proposisi.

- (a) Konjungsi $p \wedge q$ bernilai benar jika p dan q keduanya benar, selain itu nilainya salah
 - (b) Disjungsi $p \vee q$ bernilai salah jika p dan q keduanya salah, selain itu nilainya benar
 - (c) Negasi p , yaitu $\neg p$, bernilai benar jika p salah, sebaliknya bernilai salah jika p benar.
-

Contoh 1.6

Misalkan

$$\begin{aligned} p &: 17 \text{ adalah bilangan prima} \\ q &: \text{bilangan prima selalu ganjil} \end{aligned}$$

jelas bahwa p bernilai benar dan q bernilai salah sehingga konjungsi

$$p \wedge q : 17 \text{ adalah bilangan prima dan bilangan prima selalu ganjil}$$

adalah salah. ■

Satu cara yang praktis untuk menentukan nilai kebenaran proposisi majemuk adalah menggunakan tabel kebenaran (*truth table*). Tabel kebenaran menampilkan hubungan antara nilai kebenaran dari proposisi atomik. Tabel 1.1 menunjukkan tabel kebenaran untuk konjungsi, disjungsi, dan ingkaran. Pada tabel tersebut, $T = True$ (benar), dan $F = False$ (salah).

Tabel 1.1 Tabel kebenaran konjungsi, disjungsi, dan ingkaran

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

Contoh 1.7

Jika p , q , dan r adalah proposisi. Bentuklah tabel kebenaran dari ekspresi logika

$$(p \wedge q) \vee (\sim q \wedge r).$$

Penyelesaian:

Ada 3 buah proposisi atomik di dalam ekspresi logika dan setiap proposisi hanya mempunyai 2 kemungkinan nilai, sehingga jumlah kombinasi dari semua proposisi tersebut adalah $2 \times 2 \times 2 = 8$ buah. Tabel kebenaran dari proposisi $(p \wedge q) \vee (\sim q \wedge r)$ ditunjukkan pada Tabel 1.2. ■

Tabel 1.2 Tabel kebenaran proposisi $(p \wedge q) \vee (\sim q \wedge r)$

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

Proposisi majemuk dapat selalu bernilai benar untuk berbagai kemungkinan nilai kebenaran masing-masing proposisi atomiknya, atau selalu bernilai salah untuk berbagai kemungkinan nilai kebenaran masing-masing proposisi atomiknya. Kondisi ini didefinisikan di dalam Definisi 1.4 berikut:

DEFINISI 1.4 Sebuah proposisi majemuk disebut **tautologi** jika ia benar untuk semua kasus, sebaliknya disebut **kontradiksi** jika ia salah untuk semua kasus.

Yang dimaksud dengan “semua kasus” di dalam Definisi 1.4 di atas adalah semua kemungkinan nilai kebenaran dari proposisi atomiknya. Proposisi tautologi dicirikan pada kolom terakhir pada tabel kebenarannya hanya memuat T. Proposisi kontradiksi dicirikan pada kolom terakhir pada tabel kebenaran hanya memuat F.

Contoh 1.8

Misalkan p dan q adalah proposisi. Proposisi majemuk $p \vee \sim(p \wedge q)$ adalah sebuah tautologi (Tabel 1.3) karena kolom terakhir pada tabel kebenarannya hanya memuat T, sedangkan $(p \wedge q) \wedge \sim(p \vee q)$ adalah sebuah kontradiksi (Tabel 1.4) karena kolom terakhir pada tabel kebenarannya hanya memuat F. ■

Tabel 1.3 $p \vee \sim(p \wedge q)$ adalah tautologi

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

Tabel 1.4 $(p \wedge q) \wedge \sim(p \vee q)$ adalah kontradiksi

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

Adakalanya dua buah proposisi majemuk dapat dikombinasikan dalam berbagai cara namun semua kombinasi tersebut selalu menghasilkan tabel kebenaran yang sama. Kita mengatakan bahwa kedua proposisi majemuk tersebut ekivalen secara logika. Hal ini kita definisikan sebagai berikut:

DEFINISI 1.5 Dua buah proposisi majemuk, $P(p, q, ..)$ dan $Q(p, q, ..)$ disebut **ekivalen** secara logika, dilambangkan dengan $P(p, q, ...) \Leftrightarrow Q(p, q, ...)$ jika keduanya mempunyai tabel kebenaran yang identik.

Catatan:

Beberapa literatur menggunakan notasi “ \equiv ” untuk melambangkan ekivalen secara logika.

Menurut Definisi 1.5 terdapat banyak cara untuk menuliskan ekspresi logika, yang pada hakikatnya semua ekspresi logika tersebut mempunyai nilai kebenaran yang sama.

Contoh 1.9

Tabel 1.5 memperlihatkan tabel kebenaran untuk proposisi $\sim(p \wedge q)$ dan proposisi $\sim p \vee \sim q$. Kolom terakhir pada kedua tabel tersebut sama nilainya (yaitu F, T, T, T), sehingga kita katakan bahwa kedua proposisi tersebut ekivalen secara logika, atau ditulis sebagai $\sim(p \wedge q) \Leftrightarrow \sim p \vee \sim q$. Bentuk keekivalenan ini dikenal dengan nama Hukum De Morgan. ■

Tabel 1.5 $\sim(p \wedge q)$ ekivalen secara logika dengan $p \vee \sim q$

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

1.4 Disjungsi Eksklusif

Kata “atau” (*or*) dalam operasi logika digunakan dalam dua cara. Cara pertama, “atau” digunakan secara inklusif (*inclusive or*) yaitu dalam bentuk “ p atau q atau keduanya”. Artinya, disjungsi dengan operator “atau” bernilai benar jika salah satu dari proposisi atomiknya benar atau keduanya benar. Operator “atau” yang sudah kita bahas pada contoh-contoh di atas adalah yang dari jenis inklusif ini. Sebagai contoh, pernyataan

“Tenaga IT yang dibutuhkan harus menguasai Bahasa C++ atau Java”.

diartikan bahwa tenaga IT (*Information Technology*) yang diterima harus mempunyai kemampuan penguasaan salah satu dari Bahasa Java atau Bahasa C++ atau kedua-duanya. Tabel kebenaran untuk “atau” secara inklusif adalah seperti pada tabel 1.1 yang sudah dijelaskan di atas.

Cara kedua, “atau” digunakan secara eksklusif (*exclusive or*) yaitu dalam bentuk “ p atau q tetapi bukan keduanya”. Artinya, disjungsi p dengan q bernilai benar hanya jika salah satu proposisi atomiknya benar (tapi bukan keduanya). Sebagai contoh, pada sebuah ajang perlombaan pemenang dijanjikan mendapat hadiah. Hadiahnya adalah sebuah pesawat televisi 20 inchi. Jika pemenang tidak menginginkan membawa TV, panitia menggantinya dengan senilai uang.. Proposisi untuk masalah ini dituliskan sebagai berikut:

“Pemenang lomba mendapat hadiah berupa TV atau uang”

Kata “atau” pada disjungsi di atas digunakan secara eksklusif. Artinya, hadiah yang dapat dibawa pulang oleh pemenang hanya salah satu dari uang atau TV tetapi tidak bisa keduanya

Khusus untuk disjungsi eksklusif kita menggunakan operator logika *xor*, untuk membedakannya dengan *inclusive or*, yang definisinya adalah sebagai berikut:

DEFINISI 1.5. Misalkan p dan q adalah proposisi. Operasi *xor*, ditulis $p \oplus q$, dinyatakan dengan notasi: $p \oplus q$ adalah proposisi yang bernilai benar bila hanya salah satunya p atau q benar, selain itu nilainya salah.

Tabel kebenaran untuk operasi *exclusive or* ditunjukkan pada Tabel 1.6. Dari tabel tersebut dapat dibaca proposisi $p \oplus q$ hanya benar jika salah satu, tapi tidak keduanya, dari proposisi atomiknya benar.

Tabel 1.6 Tabel kebenaran *exclusive or*

p	q	$p \oplus q$
T	T	F
T	F	T
F	T	T
F	F	F

1.5 Hukum-hukum Logika Proposisi

Proposisi, dalam kerangka hubungan ekivalensi logika, memenuhi sifat-sifat yang dinyatakan dalam sejumlah hukum pada Tabel 1.7. Beberapa hukum tersebut mirip dengan hukum aljabar pada sistem bilangan riil, misalnya $a(b + c) = ab + bc$, yaitu hukum distributif, sehingga kadang-kadang hukum logika proposisi dinamakan juga **hukum-hukum aljabar proposisi**.

Tabel 1.7 Hukum-hukum logika (atau hukum-hukum aljabar proposisi)

1. Hukum identitas: (i) $p \vee \mathbf{F} \Leftrightarrow p$ (ii) $p \wedge \mathbf{T} \Leftrightarrow p$	2. Hukum <i>null/dominasi</i> : (i) $p \wedge \mathbf{F} \Leftrightarrow \mathbf{F}$ (ii) $p \vee \mathbf{T} \Leftrightarrow \mathbf{T}$
3. Hukum negasi: (i) $p \vee \neg p \Leftrightarrow \mathbf{T}$ (ii) $p \wedge \neg p \Leftrightarrow \mathbf{F}$	4. Hukum idempoten: (i) $p \vee p \Leftrightarrow p$ (ii) $p \wedge p \Leftrightarrow p$

5. Hukum involusi (negasi ganda): $\sim(\sim p) \Leftrightarrow p$	6. Hukum penyerapan (absorpsi): (i) $p \vee (p \wedge q) \Leftrightarrow p$ (ii) $p \wedge (p \vee q) \Leftrightarrow p$
7. Hukum komutatif: (i) $p \vee q \Leftrightarrow q \vee p$ (ii) $p \wedge q \Leftrightarrow q \wedge p$	8. Hukum asosiatif: (i) $p \vee (q \vee r) \Leftrightarrow (p \vee q) \vee r$ (ii) $p \wedge (q \wedge r) \Leftrightarrow (p \wedge q) \wedge r$
9. Hukum distributif: (ii) $p \vee (q \wedge r) \Leftrightarrow (p \vee q) \wedge (p \vee r)$ (ii) $p \wedge (q \vee r) \Leftrightarrow (p \wedge q) \vee (p \wedge r)$	10. Hukum De Morgan: (i) $\sim(p \wedge q) \Leftrightarrow \sim p \vee \sim q$ (ii) $\sim(p \vee q) \Leftrightarrow \sim p \wedge \sim q$

Hukum-hukum logika di atas bermanfaat untuk membuktikan keekivalenensi dua buah proposisi. Selain menggunakan tabel kebenaran, keekivalenensi dapat dibuktikan dengan hukum-hukum logika, khususnya pada proposisi majemuk yang mempunyai banyak proposisi atomik. Bila suatu proposisi majemuk mempunyai n buah porposisi atomik, maka tabel kebenarannya terdiri dari 2^n baris. Untuk n yang besar jelas tidak praktis menggunakan tabel kebenaran, misalnya untuk $n = 10$ terdapat 2^{10} baris di dalam tabel kebenarannya.

Contoh 1.10

Tunjukkan bahwa $p \vee \sim(p \vee q)$ dan $p \vee \sim q$ keduanya ekivalen secara logika.

Penyelesaian:

$$\begin{aligned}
 p \vee \sim(p \vee q) &\Leftrightarrow p \vee (\sim p \wedge \sim q) && \text{(Hukum De Morgan)} \\
 &\Leftrightarrow (p \vee \sim p) \wedge (p \vee \sim q) && \text{(Hukum distributif)} \\
 &\Leftrightarrow T \wedge (p \vee \sim q) && \text{(Hukum negasi)} \\
 &\Leftrightarrow p \vee \sim q && \text{(Hukum identitas)}
 \end{aligned}$$

■

Contoh 1.11

Buktikan hukum penyerapan: $p \wedge (p \vee q) \Leftrightarrow p$

Penyelesaian:

$$\begin{aligned}
 p \wedge (p \vee q) &\Leftrightarrow (p \vee F) \wedge (p \vee q) && \text{(Hukum Identitas)} \\
 &\Leftrightarrow p \vee (F \wedge q) && \text{(Hukum distributif)} \\
 &\Leftrightarrow p \vee F && \text{(Hukum Null)} \\
 &\Leftrightarrow p && \text{(Hukum Identitas)}
 \end{aligned}$$

■

1.6 Operasi Logika di dalam Komputer

Bahasa pemrograman umumnya menyediakan tipe data **boolean** untuk data yang bertipe logika, misalnya tipe boolean dalam Bahasa Pascal, logical dalam Bahasa Fortran, dan sebagainya. Tipe data **boolean** hanya mempunyai dua buah konstanta nilai saja, yaitu *true* dan *false*. Peubah yang bertipe boolean disebut **peubah boolean** (*boolean variable*). Nilai peubah tersebut hanya *true* atau *false*.

Operasi *boolean* sering dibutuhkan dalam pemrograman. Operasi *boolean* dinyatakan dalam ekspresi logika (atau dinamakan juga ekspresi boolean). Operator *boolean* yang digunakan adalah *AND*, *OR*, *XOR*, dan *NOT*. Ekspresi *boolean* tersebut hanya menghasilkan salah satu dari dua nilai, *true* atau *false*. Misalkan x_1 , x_2 , x_3 , dan x_4 adalah peubah *boolean* dalam Bahasa Pascal, maka ekspresi *boolean* di bawah ini adalah valid:

```
x1 and x2  
x1 or (not(x2 and x3))
```

yang bersesuaian dengan ekspresi logika

$$\begin{aligned} &x_1 \wedge x_2 \\ &x_1 \vee \sim(x_2 \wedge x_3) \end{aligned}$$

Operasi lain dalam pemrograman yang bersesuaian dengan operasi logika adalah **operasi bit**. Komputer merepresentasikan informasi dengan menggunakan bit. Sebuah bit hanya mempunyai dua nilai, yaitu 1 atau 0. Sebuah bit dapat digunakan untuk merepresentasikan nilai kebenaran, yaitu kita menyatakan 1 untuk merepresentasikan *true* (T) dan 0 untuk merepresentasikan *false* (F). Kita menggunakan notasi \sim , \wedge , \vee , dan \oplus masing-masing untuk melambangkan operator *NOT*, *AND*, *OR*, dan *XOR*. Dengan demikian, operasi bit

$$\begin{aligned} &\sim 0 \\ &1 \wedge 0 \\ &0 \vee 0 \\ &1 \oplus 0 \end{aligned}$$

bersesuaian dengan operasi logika

$$\begin{aligned} &\sim F \\ &T \wedge F \\ &F \vee F \\ &T \oplus F \end{aligned}$$

Operasi bit dapat diperluas untuk rangkaian bit yang panjangnya tetap, misalnya 10011011 dioperasikan dengan 01010101. Operasi ini dinamakan *bitwise*, dan

operasi semacam ini digunakan untuk memanipulasi informasi. Dua buah rangkaian bit yang panjangnya sama dapat dioperasikan dengan 3-operasi *bitwise*, yaitu *bitwise AND*, *bitwise OR*, dan *bitwise XOR*. Jika dua buah rangkaian bit dioperasikan dengan salah satu dari operator *bitwise* di atas, maka setiap bit yang bersesuaian pada masing-masing *operand* dikenai operasi yang sama. Misalnya,

$$\begin{array}{r} 10011011 \\ 01010101 \\ \hline 00010001 \quad \textit{bitwise AND} \\ 11011111 \quad \textit{bitwise OR} \\ 11001110 \quad \textit{bitwise XOR} \end{array}$$

Aplikasi operasi logika lainnya ditemukan pada mesin pencarian (*search engine*) di internet. Salah satu mesin pencarian yang terkenal dan banyak digunakan orang adalah *Google* (www.google.com). Tersedia juga *Google* versi Bahasa Indonesia (www.google.co.id). Antarmuka *Google* diperlihatkan pada Gambar 1.1. Mesin pencarian adalah aplikasi yang sangat penting di internet, karena mesin pencarian mampu menampilkan semua informasi yang kita butuhkan dalam waktu yang cepat. Hasil pencarian adalah halaman *web* yang berkaitan dengan *term* yang kita ketikkan.



Gambar 1.1 Antarmuka mesin pencarian *Google*.

Operator logika AND, OR, dan NOT dapat digunakan sebagai kata hubung logika di antara *term-term* yang dicari. Misalkan kita ingin mencari semua halaman *web* yang berkaitan dengan “aljabar” atau “boolean”, maka *term* yang kita cari ditulis sebagai

aljabar OR boolean

Hasilnya adalah semua halaman yang mengandung salah satu kata “aljabar”, “boolean”, atau kedua-duanya. Bila kita ingin mencari semua halaman web yang tepat mengandung kata “aljabar” dan “boolean” sekaligus, maka *term* yang kita ketikkan di dalam mesin pencarian ditulis sebagai

aljabar AND boolean

Hasilnya adalah semua halaman yang mengandung dua kata “aljabar” dan “boolean” sekaligus. (catatan: beberapa mesin pencarian tidak memerlukan penulisan AND secara eksplisit).

Bila kita ingin mencari semua halaman *web* yang berkaitan dengan dengan topik “aljabar” atau “boolean” dan untuk setiap topik tersebut harus berkaitan dengan “matematika”, maka *term* dituliskan sebagai

(aljabar OR boolean) AND matematika

Hasilnya adalah semua halaman yang mengandung tepat kata “aljabar” dan “matematika”, atau yang mengandung tepat kata “boolean” dan “matematika”, atau yang sekaligus mengandung “aljabar”, “boolean”, dan “matematika”. Gambar 1.2 memperlihatkan hasil pencarian untuk *term* di atas (pencarian dilakukan pada Tanggal 22 Juni 2005 pukul 11.00 WIB. Perhatikan bahwa informasi di internet dapat berubah setiap saat (*up to date*), jadi mungkin saja pada hari ini hasil pencarian sangat berbeda dengan hasil pencarian pada tanggal 22 Juni 2005).

The screenshot shows a Microsoft Internet Explorer window displaying Google search results. The search query is '(aljabar OR boolean) AND matematika'. The results page includes a navigation bar with links for 'Web', 'Gambar', 'Grup', 'Email', and 'Direktori'. A search bar with the query is present, along with a 'Cari' button and a 'Pencarian Canggih' link. Below the search bar, there's a note about searching Indonesian websites. The main results section starts with a link to a document titled 'UNDANGAN' from 'Format File Microsoft Word 2000'. Other results include a seminar announcement from UPI, a document from FMIPA ITB, and a link to the 'S1 Matematika :: FMIPA UGM' website. The search results also mention 'Ilmu komputer - Wikipedia Indonesia'.

Gambar 1.2 Hasil pencarian untuk term "(aljabar OR boolean) AND matematika"

1.7 Proposisi Bersyarat (Implikasi)

Selain dalam bentuk konjungsi, disjungsi, dan negasi, proposisi majemuk juga dapat muncul berbentuk "jika p , maka q ", seperti pada contoh-contoh berikut:

- Jika adik lulus ujian, maka ia mendapat hadiah dari ayah.
- Jika suhu mencapai 80°C , maka alarm berbunyi.
- Jika anda tidak mendaftar ulang, maka anda dianggap mengundurkan diri

Pernyataan berbentuk "jika p , maka q " semacam itu disebut **proposisi bersyarat** atau **kondisional** atau **implikasi**.

DEFINISI 1.6. Misalkan p dan q adalah proposisi. Proposisi majemuk "jika p , maka q " disebut proposisi bersyarat (implikasi) dan dilambangkan dengan

$$p \rightarrow q$$

Proposisi p disebut **hipotesis** (atau **anteseden** atau **premis** atau **kondisi**) dan proposisi q disebut **konklusi** (atau **konsekuensi**).

Tabel kebenaran implikasi ditunjukkan pada Tabel 1.8. Catatlah bahwa implikasi $p \rightarrow q$ hanya salah jika p benar tetapi q salah, selain itu implikasi bernilai benar. Tidak sukar memahami mengapa tabel kebenaran implikasi demikian. Hal ini dijelaskan dengan contoh analogi berikut: Misalkan dosen anda berkata kepada mahasiswanya di dalam kelas “Jika nilai ujian akhir anda 80 atau lebih, maka anda akan mendapat nilai A untuk kuliah ini”. Apakah dosen anda mengatakan kebenaran atau dia berbohong? Tinjau empat kasus berikut ini:

Kasus 1: Nilai ujian akhir anda di atas 80 (hipotesis benar) dan anda mendapat nilai A untuk kuliah tersebut (konklusi benar). Pada kasus ini, pernyataan dosen anda benar.

Kasus 2: Nilai ujian akhir anda di atas 80 (hipotesis benar) tetapi anda tidak mendapat nilai A (konklusi salah). Pada kasus ini, dosen anda berbohong (pernyataannya salah).

Kasus 3: Nilai ujian akhir anda di bawah 80 (hipotesis salah) dan anda mendapat nilai A (konklusi benar). Pada kasus ini, dosen anda tidak dapat dikatakan salah (Mungkin ia melihat kemampuan anda secara rata-rata bagus sehingga ia tidak ragu memberi nilai A).

Kasus 4: Nilai ujian akhir anda di bawah 80 (hipotesis salah) dan anda tidak mendapat nilai A (konklusi salah). Pada kasus ini dosen anda benar.

Tabel 1.8 Tabel kebenaran implikasi

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

Di dalam bahasa alami (bahasa percakapan manusia), seperti Bahasa Indonesia dan Bahasa Inggris, terdapat hubungan sebab-akibat antara hipotesis dengan konklusi, misalnya pada implikasi

“Jika suhu mencapai 80°C , maka alarm berbunyi.”

Implikasi seperti ini adalah normal dalam Bahasa Indonesia. Tetapi, dalam penalaran matematik, kita memandang implikasi lebih umum daripada implikasi dalam bahasa alami. Konsep matematik mengenai implikasi independen dari hubungan sebab-akibat antara hipotesis dan konklusi. Definisi kita mengenai

implikasi adalah pada nilai kebenarannya, bukan didasarkan pada penggunaan bahasa [ROS03]. Misalnya pada implikasi

“Jika Paris adalah ibukota Perancis, maka $1 + 1 = 2$ ”

Implikasi di atas tetap valid secara matematis meskipun tidak ada kaitan antara Paris sebagai ibukota Perancis dengan $1 + 1 = 2$. Implikasi tersebut bernilai benar karena hipotesis benar (Paris ibukota Perancis adalah benar) dan konklusi juga benar ($1 + 1 = 2$ adalah benar). Implikasi

“Jika Paris adalah ibukota Perancis, maka $1 + 1 = 3$ ”

bernilai salah karena hipotesis benar tetapi $1 + 1 = 3$ salah.

Implikasi $p \rightarrow q$ memainkan peranan penting dalam penalaran. Implikasi ini tidak hanya diekspresikan dalam pernyataan standard “jika p , maka q ” tetapi juga dapat diekspresikan dalam berbagai cara, antara lain:

- | | |
|-------------------------------|--|
| (a) Jika p , maka q | <i>(if p, then q)</i> |
| (b) Jika p , | <i>(if p, q)</i> |
| (c) p mengakibatkan q | <i>(p implies q)</i> |
| (d) q jika p | <i>(q if p)</i> |
| (e) p hanya jika q | <i>(p only if q)</i> |
| (f) p syarat cukup agar q | <i>(p is sufficient for q)</i> |
| (g) q syarat perlu bagi p | <i>(q is necessary for p)</i> |
| (i) q bilamana p | <i>(q whenever p)</i> |

Contoh-contoh berikut memperlihatkan implikasi dalam berbagai ekspresi serta bagaimana mengubah berbagai bentuk implikasi menjadi bentuk standard “jika p , maka q ”.

Contoh 1.12

Proposisi-proposisi berikut adalah implikasi dalam berbagai bentuk:

- (a) Jika hari hujan, maka tanaman akan tumbuh subur.
 - (b) Jika tekanan gas diperbesar, mobil melaju kencang.
 - (c) Es yang mencair di kutub mengakibatkan permukaan air laut naik.
 - (d) Orang itu mau berangkat jika ia diberi ongkos jalan.
 - (e) Ahmad bisa mengambil matakuliah Teori Bahasa Formal hanya jika ia sudah lulus matakuliah Matematika Diskrit.
 - (f) Syarat cukup agar pom bensin meledak adalah percikan api dari rokok.
 - (g) Syarat perlu bagi Indonesia agar ikut Piala Dunia adalah dengan mengontrak pemain asing kenamaan.
 - (h) Banjir bandang terjadi bilamana hutan ditebangi.
-

Contoh 1.13

Ubahlah proposisi c sampai h di dalam Contoh 1.12 ke dalam bentuk proposisi “jika p , maka q ”.

Penyelesaian:

- (c) Jika es mencair di kutub, maka permukaan air laut naik.
- (d) Jika orang itu diberi ongkos jalan, maka ia mau berangkat.
- (e) Jika Ahmad mengambil matakuliah Teori Bahasa Formal, maka ia sudah lulus matakuliah Matematika Diskrit.
- (f) Pernyataan yang diberikan ekivalen dengan “Percikan api dari rokok adalah syarat cukup untuk membuat pom bensin meledak” atau “Jika api memercik dari rokok maka pom bensin meledak”
- (g) Pernyataan yang diberikan ekivalen dengan “Mengontrak pemain asing kenamaan adalah syarat perlu untuk Indonesia agar ikut Piala Dunia” atau “Jika Indonesia ikut Piala Dunia maka Indonesia mengontrak pemain asing kenamaan”.
- (h) Jika hutan-hutan ditebangi, maka banjir bandang terjadi. ■

Contoh 1.14

Misalkan

- x : Anda berusia 17 tahun
- y : Anda dapat memperoleh SIM

Nyatakan preposisi berikut ke dalam notasi implikasi:

- (a) Hanya jika anda berusia 17 tahun maka anda dapat memperoleh SIM.
- (b) Syarat cukup agar anda dapat memperoleh SIM adalah anda berusia 17 tahun.
- (c) Syarat perlu agar anda dapat memperoleh SIM adalah anda berusia 17 tahun.
- (d) Jika anda tidak dapat memperoleh SIM maka anda tidak berusia 17 tahun.
- (e) Anda tidak dapat memperoleh SIM bilamana anda belum berusia 17 tahun.

Penyelesaian:

- (a) Pernyataan yang diberikan ekivalen dengan “Anda dapat memperoleh SIM hanya jika anda berusia 17 tahun”. Ingat kembali bahwa $p \rightarrow q$ bisa dibaca “ p hanya jika q ”. Jadi, pernyataan yang diberikan dilambangkan dengan $y \rightarrow x$.
- (b) Pernyataan yang diberikan ekivalen dengan “Anda berusia 17 tahun adalah syarat cukup untuk dapat memperoleh SIM”. Ingat kembali bahwa $p \rightarrow q$ bisa dibaca “ p syarat cukup untuk q ”. Jadi, pernyataan yang diberikan dilambangkan dengan $x \rightarrow y$.
- (c) Pernyataan yang diberikan ekivalen dengan “Anda berusia 17 tahun adalah syarat perlu untuk dapat memperoleh SIM”. Ingat kembali bahwa $p \rightarrow q$ bisa dibaca “ q syarat perlu untuk q ”. Jadi, pernyataan yang diberikan dilambangkan dengan $y \rightarrow x$.
- (d) $\sim y \rightarrow \sim x$
- (e) Ingat kembali bahwa $p \rightarrow q$ bisa dibaca “ q bilamana p ”. Jadi, pernyataan yang diberikan dilambangkan dengan $\sim x \rightarrow \sim y$. ■

Contoh 1.15

Tunjukkan bahwa $p \rightarrow q$ ekivalen secara logika dengan $\sim p \vee q$.

Penyelesaian:

Tabel 1.9 memperlihatkan bahwa memang benar $p \rightarrow q \Leftrightarrow \sim p \vee q$. Dengan kata lain, pernyataan “Jika p maka q ” ekivalen secara logika dengan “Tidak p atau q ”. ■

Tabel 1.9 Tabel kebenaran $p \rightarrow q$ dan $\sim p \vee q$.

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

Contoh 1.16

Tentukan ingkaran (negasi) dari $p \rightarrow q$.

Penyelesaian:

Dari Contoh 1.15 sudah ditunjukkan bahwa $p \rightarrow q$ ekivalen secara logika dengan $\sim p \vee q$. Gunakan hukum DeMorgan untuk menentukan ingkaran dari $p \rightarrow q$:

$$\sim(p \rightarrow q) \Leftrightarrow \sim(\sim p \vee q) \Leftrightarrow \sim(\sim p) \wedge \sim q \Leftrightarrow p \wedge \sim q$$

Contoh 1.17

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?

Penyelesaian:

Untuk memeriksa apakah kedua moto tersebut sama, kita perlu membandingkan tabel kebenaran keduanya. Misalkan p menyatakan proposisi “Barang itu bagus” sedangkan q menyatakan “Barang itu murah”. Maka, moto pedagang pertama dapat ditulis sebagai “Jika barang itu bagus maka barang itu tidak murah” atau $p \rightarrow \sim q$, sedangkan moto kedua dapat ditulis sebagai “Jika barang itu murah maka barang itu tidak bagus” atau $q \rightarrow \sim p$. Tabel kebenaran untuk proposisi $p \rightarrow \sim q$ dan proposisi $q \rightarrow \sim p$ ditunjukkan pada Tabel 1.10. Dari tabel tersebut dapat dilihat ternyata nilai kebenaran proposisi $p \rightarrow \sim q$ dan proposisi $q \rightarrow \sim p$ sama, dengan kata lain $p \rightarrow \sim q \Leftrightarrow q \rightarrow \sim p$. Jadi kita dapat menyimpulkan bahwa kedua moto tersebut menyatakan hal yang sama. ■

Tabel 1.10 Tabel kebenaran $p \rightarrow \sim q$ dan $q \rightarrow \sim p$

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

Banyak orang yang bingung mengapa bentuk “ p hanya jika q ” sama dengan “jika p , maka q ”. Untuk menjelaskan hal ini kita harus mengingat bahwa “ p hanya jika q ” menyatakan bahwa p tidak dapat benar bila q salah. Dengan kata lain, pernyataan “ p hanya jika q ” salah jika p benar, tetapi q salah. Bila p salah, q dapat salah satu dari benar atau salah, karena pernyataan tersebut tidak menyatakan apa-apa tentang nilai kebenaran q [ROS03].

Implikasi dalam Bahasa Pemrograman

Struktur *if-then* yang digunakan pada kebanyakan bahasa pemrograman berbeda dengan implikasi *if-then* yang digunakan dalam logika. Struktur *if-then* dalam bahasa pemrograman berbentuk

if c then S

yang dalam hal ini c adalah sebuah ekspresi logika yang menyatakan syarat atau kondisi, sedangkan S berupa satu atau lebih pernyataan. Ketika program dieksekusi dan menjumpai pernyataan *if-then*, S dieksekusi jika c benar, tetapi S tidak dieksekusi jika c salah.

Pernyataan *if-then* dalam bahasa pemrograman bukan proposisi karena tidak ada korespondensi antara pernyataan tersebut dengan operator implikasi (\rightarrow). Penginterpretasi bahasa pemrograman (disebut *interpreter* atau *compiler*) tidak melakukan penilaian kebenaran pernyataan *if-then* secara logika. *Interpreter* hanya memeriksa kebenaran kondisi c , jika c benar maka S dieksekusi, sebaliknya jika c salah maka S tidak dieksekusi. Sebagai contoh, perhatikan Contoh 1.18 berikut ini.

Contoh 1.18

Misalkan di dalam sebuah program yang ditulis dalam Bahasa Pascal terdapat pernyataan berikut:

if $x > y$ then $y := x + 10;$

(simbol := menyatakan operator pengisian nilai, yaitu nilai ekspresi di ruas kanan simbol := diisikan ke dalam peubah di ruas kiri simbol :=). “ $x > y$ ” adalah ekspresi logika yang nilainya benar atau salah bergantung pada nilai x dan y , sedangkan $y := x + 10$ adalah sebuah pernyataan aritmetika yang akan dieksekusi jika ekspresi logika $x > y$ benar.

Berapa nilai y setelah pelaksanaan pernyataan *if-then* di atas jika nilai x dan y sebelum pernyataan tersebut adalah (i) $x = 2$, $y = 1$, dan (ii) $x = 3$, $y = 5$?

Penyelesaian:

- (i) sebelum pernyataan *if-then* nilai $x = 2$ dan $y = 1$, maka ekspresi $x > y$ bernilai benar sehingga pernyataan $y := x + 10$ dilaksanakan, yang mengakibatkan nilai y sekarang menjadi $y = 2 + 10 = 12$.
 - (ii) sebelum pernyataan *if-then* nilai $x = 3$ dan $y = 5$, maka ekspresi $x > y$ bernilai salah sehingga pernyataan $y := x + 10$ tidak dilakukan. Dalam hal ini, nilai y tetap seperti sebelumnya, yaitu 5. ■
-

1.8 Varian Proposisi Bersyarat

Terdapat bentuk implikasi lain yang berkaitan dengan $p \rightarrow q$, yaitu proposisi sederhana yang merupakan varian dari implikasi. Ketiga variasi proposisi bersyarat tersebut adalah konvers, invers, dan kontraposisi dari proposisi asal $p \rightarrow q$.

Konvers (kebalikan)	:	$q \rightarrow p$
Invers	:	$\sim p \rightarrow \sim q$
Kontraposisi	:	$\sim q \rightarrow \sim p$

Tabel 1.11 memperlihatkan tabel kebenaran dari ketiga varian proposisi bersyarat tersebut. Dari tabel tersebut terlihat bahwa proposisi bersyarat $p \rightarrow q$ ekivalen secara logika dengan dengan kontraposisinya, $\sim q \rightarrow \sim p$.

Tabel 1.11 Tabel kebenaran implikasi, konvers, invers, dan kontraposisi

p	q	$\sim p$	$\sim q$	Implikasi $p \rightarrow q$	Konvers $q \rightarrow p$	Invers $\sim p \rightarrow \sim q$	Kontraposisi $\sim q \rightarrow \sim p$
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 1.19

Tentukan konvers, invers, dan kontraposisi dari pernyataan berikut

“Jika Amir mempunyai mobil, maka ia orang kaya”

Penyelesaian:

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 ia tidak mempunyai mobil ■

Contoh 1.20

Tentukan kontraposisi dari pernyataan:

- (a) Jika dia bersalah maka ia dimasukkan ke dalam penjara.
- (b) Jika 6 lebih besar dari 0 maka 6 bukan bilangan negatif.
- (c) Iwan lulus ujian hanya jika ia belajar.
- (d) Hanya jika ia tidak terlambat maka ia akan mendapat pekerjaan itu.
- (e) Perlu ada angin agar layang-layang bisa terbang.
- (f) Cukup hari hujan agar hari ini dingin.

Penyelesaian:

- (a) Jika ia tidak dimasukkan ke dalam penjara, maka ia tidak bersalah.
 - (b) Jika 6 bilangan negatif, maka 6 tidak lebih besar dari 0.
 - (c) Pernyataan yang diberikan ekivalen dengan “Jika Iwan lulus ujian maka ia sudah belajar”, sehingga kontraposisinya adalah “Jika Iwan tidak belajar maka ia tidak lulus ujian”
 - (d) Pernyataan yang diberikan ekivalen dengan “Jika ia mendapat pekerjaan itu maka ia tidak terlambat”, sehingga kontraposisinya adalah “Jika ia terlambat maka ia tidak akan mendapat pekerjaan itu”
 - (e) Pernyataan yang diberikan dapat ditulis kembali menjadi “Ada angin adalah syarat perlu agar layang-layang bisa terbang” yang dalam hal ini ekivalen dengan “Jika layang-layang bisa terbang maka hari ada angin”. Kontraposisinya adalah “Jika hari tidak ada angin, maka layang-layang tidak bisa terbang”.
 - (f) Pernyataan yang diberikan dapat ditulis kembali menjadi “Hari hujan adalah syarat cukup agar hari ini dingin”, yang dalam hal ini ekivalen dengan “Jika hari hujan maka hari ini dingin”. Kontraposisinya adalah “Jika hari ini tidak dingin maka hari tidak hujan”. ■
-

1.9 Bikondisional (Bi-implikasi)

Proposisi bersyarat penting lainnya adalah berbentuk “ p jika dan hanya jika q ” yang dinamakan **bikondisional** atau **bi-implikasi**. Definisi bikondisional dikemukakan sebagai berikut.

DEFINISI 1.7. Misalkan p dan q adalah proposisi. Proposisi majemuk “ p jika dan hanya jika q ” disebut bikondisional (bi-implikasi) dan dilambangkan dengan $p \leftrightarrow q$.

Pernyataan $p \leftrightarrow q$ adalah benar bila p dan q mempunyai nilai kebenaran yang sama, yakni $p \leftrightarrow q$ benar jika p dan q keduanya benar atau p dan q keduanya salah. Tabel kebenaran selengkapnya diperlihatkan pada Tabel 1.12.

Tabel 1.12 Tabel kebenaran bikondisional

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

Perhatikan bahwa bikondisional $p \leftrightarrow q$ ekivalen secara logika dengan $(p \rightarrow q) \wedge (q \rightarrow p)$. Keekivalenannya tersebut ditunjukkan pada Tabel 1.13. Dengan kata lain, pernyataan “ p jika dan hanya jika q ” dapat dibaca “Jika p maka q dan jika q maka p ”.

Tabel 1.13 $p \leftrightarrow q \Leftrightarrow (p \rightarrow q) \wedge (q \rightarrow p)$.

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

Terdapat sejumlah cara untuk menyatakan bikondisional $p \leftrightarrow q$ dalam kata-kata, yaitu:

- (a) p jika dan hanya jika q . *(p if and only if q)*
- (b) p adalah syarat perlu dan cukup untuk q . *(p is necessary and sufficient for q)*
- (c) Jika p maka q , dan sebaliknya. *(if p then q, and conversely)*
- (d) p iff q

Contoh 1.21

Proposisi majemuk berikut adalah bi-implikasi:

- (a) $1 + 1 = 2$ jika dan hanya jika $2 + 2 = 4$.
- (b) Syarat cukup dan syarat perlu agar hari hujan adalah kelembaban udara tinggi.
- (c) Jika anda orang kaya maka anda mempunyai banyak uang, dan sebaliknya.
- (d) Bandung terletak di Jawa Barat iff Jawa Barat adalah sebuah propinsi di Indonesia.

Contoh 1.22

Tuliskan setiap proposisi berikut ke dalam bentuk “ p jika dan hanya jika q ”:

- (a) Jika udara di luar panas maka anda membeli es krim, dan jika anda membeli es krim -maka udara di luar panas.
- (b) Syarat cukup dan perlu agar anda memenangkan pertandingan adalah anda melakukan banyak latihan.
- (c) Anda naik jabatan jika anda punya koneksi, dan anda punya koneksi jika anda naik jabatan.
- (d) Jika anda lama menonton televisi maka mata anda lelah, begitu sebaliknya.
- (e) Kereta api datang terlambat tepat pada hari-hari ketika saya membutuhkannya.

Penyelesaian:

- (a) Anda membeli es krim jika dan hanya jika udara di luar panas.
 - (b) Anda melakukan banyak latihan adalah syarat perlu dan cukup untuk anda memenangkan pertandingan.
 - (c) Anda naik jabatan jika dan hanya jika anda punya koneksi.
 - (d) Mata anda lelah jika dan hanya jika anda lama menonton televisi.
 - (e) Kereta api datang terlambat jika dan hanya jika saya membutuhkan kereta hari itu.
-

Contoh 1.23

[LIU85] Sebuah pulau didiami oleh dua suku asli. Penduduk suku pertama selalu mengatakan hal yang benar, sedangkan penduduk dari suku lain selalu mengatakan kebohongan. Anda tiba di pulau ini dan bertanya kepada seorang penduduk setempat apakah di pulau tersebut ada emas atau tidak. Ia menjawab, “Ada emas di pulau ini jika dan hanya jika saya selalu mengatakan kebenaran”. Apakah ada emas di pulau tersebut?

Penyelesaian:

Misalkan

$$\begin{aligned} p &: \text{saya selalu menyatakan kebenaran} \\ q &: \text{ada emas di pulau ini} \end{aligned}$$

Pernyataan orang tersebut dapat dinyatakan sebagai

$$p \leftrightarrow q$$

Tinjau dua kemungkinan kasus mengenai orang yang kita tanya tadi. Kasus 1, orang yang memberi jawaban adalah orang dari suku yang selalu menyatakan hal yang benar. Kasus 2, orang yang memberi jawaban adalah orang dari suku yang selalu menyatakan hal yang bohong. Kita analisis setiap kasus satu persatu sebagai berikut:

Kasus 1: orang tersebut selalu menyatakan hal yang benar. Ini berarti p benar, dan jawabannya terhadap pertanyaan kita pasti juga benar, sehingga pernyataan bi-implikasi tersebut bernilai benar. Dari Tabel 1.12 kita melihat bahwa bila p benar dan $p \leftrightarrow q$ benar, maka q harus benar. Jadi, ada emas di pulau tersebut adalah benar.

Kasus 2: orang tersebut selalu menyatakan hal yang bohong. Ini berarti p salah, dan jawabannya terhadap pertanyaan kita pasti juga salah, sehingga pernyataan bi-implikasi tersebut salah. Dari Tabel 1.12 kita melihat bahwa bila p salah dan $p \leftrightarrow q$ salah, maka q harus benar. Jadi, ada emas di pulau tersebut adalah benar.

Dari kedua kasus, kita selalu berhasil menyimpulkan bahwa ada emas di pulau tersebut, meskipun kita tidak dapat memastikan dari suku mana orang tersebut. ■

Tinjau kembali bahasan dua buah proposisi majemuk yang ekivalen secara logika. Kita juga dapat menggunakan definisi bikondisional untuk menyatakan keekivalenan. Ingatlah bahwa bikondisional bernilai benar jika kedua proposisi atomiknya mempunyai nilai kebenaran sama. Oleh karena itu, bila dua proposisi majemuk yang ekivalen di-bikondisionalkan, maka hasilnya adalah tautologi. Hal ini kita nyatakan pada definisi 1.8 berikut ini.

DEFINISI 1.8. Dua buah proposisi majemuk, $P(p, q, \dots)$ dan $Q(p, q, \dots)$ disebut ekivalen secara logika, dilambangkan dengan $P(p, q, \dots) \Leftrightarrow Q(p, q, \dots)$, jika $P \leftrightarrow Q$ adalah tautologi.

Definisi 1.8 di atas mudah dimengerti. Dari tabel kebenaran bikondisional pada Tabel 1.12 kita melihat bahwa bikondisional hanya benar jika kedua proposisi mempunyai nilai kebenaran yang sama. Jika dua proposisi majemuk mempunyai tabel kebenaran yang sama, maka bikondisional terhadap kedua proposisi majemuk tersebut menghasilkan nilai yang semuanya benar, dengan kata lain tautologi.

Contoh 1.23

Tinjau kembali Contoh 1.9. Kita sudah membuktikan bahwa $\sim(p \wedge q) \Leftrightarrow \sim p \vee \sim q$. Keekivalenannya dapat juga kita tunjukkan dengan membikondisionalkan masing-masing proposisi majemuk tersebut. Dari Tabel 1.14 terlihat bahwa $\sim(p \wedge q) \Leftrightarrow \sim p \vee \sim q$ tautologi, dengan kata lain $\sim(p \wedge q) \Leftrightarrow \sim p \vee \sim q$. ■

Tabel 1.14 $\sim(p \wedge q) \Leftrightarrow \sim p \vee \sim q$ adalah tautologi

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

1.10 Inferensi

Misalkan kepada kita diberikan beberapa proposisi. Kita dapat menarik kesimpulan baru dari deret proposisi tersebut. Proses penarikan kesimpulan dituliskan dengan cara:

Di dalam kalkulus proposisi, terdapat sejumlah kaidah inferensi, beberapa di antaranya adalah sebagai berikut:

1. Modus Ponen atau *law of detachment*

Kaidah ini didasarkan pada tautologi $(p \wedge (p \rightarrow q)) \rightarrow q$, yang dalam hal ini, p dan $p \rightarrow q$ adalah hipotesis, sedangkan q adalah konklusi. Kaidah modus ponen dapat dituliskan dengan cara:

$$\begin{array}{c} p \rightarrow q \\ p \\ \hline \therefore q \end{array}$$

Simbol \therefore dibaca sebagai “jadi” atau “karena itu”. Modus ponen menyatakan bahwa jika hipotesis p dan implikasi $p \rightarrow q$ benar, maka konklusi q benar.

Contoh 1.24

Misalkan implikasi “Jika 20 habis dibagi 2, maka 20 adalah bilangan genap” dan hipotesis “20 habis dibagi 2” keduanya benar. Maka menurut modus ponen, inferensi berikut:

“Jika 20 habis dibagi 2, maka 20 adalah bilangan genap. 20 habis dibagi 2. Karena itu, 20 adalah bilangan genap”

adalah benar. Kita juga dapat menuliskan inferensi di atas sebagai:

Jika 20 habis dibagi 2, maka 20 adalah bilangan genap

20 habis dibagi 2

\therefore 20 adalah bilangan genap

2. Modus Tollen

Kaidah ini didasarkan pada tautologi $[\sim q \wedge (p \rightarrow q)] \rightarrow \sim p$, Kaidah ini modus tollens dituliskan dengan cara:

$$\begin{array}{c} p \rightarrow q \\ \sim q \\ \hline \therefore \sim p \end{array}$$

Contoh 1.25

Misalkan implikasi “Jika n bilangan ganjil, maka n^2 bernilai ganjil” dan hipotesis “ n^2 bernilai genap” keduanya benar. Maka menurut modus tollen, inferensi berikut

Jika n bilangan ganjil, maka n^2 bernilai ganjil
 n^2 bernilai genap

∴ n bukan bilangan ganjil

adalah benar. ■

3. Silogisme Hipotetis

Kaidah ini didasarkan pada tautologi $[(p \rightarrow q) \wedge (q \rightarrow r)] \rightarrow (p \rightarrow r)$. Kaidah silogisme ditulis dengan cara:

$$\begin{array}{c} p \rightarrow q \\ q \rightarrow r \\ \hline \therefore p \rightarrow r \end{array}$$

Contoh 1.26

Misalkan implikasi “Jika saya belajar dengan giat, maka saya lulus ujian” dan implikasi “Jika saya lulus ujian, maka saya cepat menikah” adalah benar. Maka menurut kaidah silogisme, inferensi berikut

Jika saya belajar dengan giat, maka saya lulus ujian

Jika saya lulus ujian, maka saya cepat menikah

∴ Jika saya belajar dengan giat, maka saya cepat menikah

adalah benar. ■

4. Silogisme Disjungtif

Kaidah ini didasarkan pada tautologi $[(p \vee q) \wedge \sim p] \rightarrow q$. Kaidah silogisme disjungtif ditulis dengan cara:

$$\begin{array}{c} p \vee q \\ \sim p \\ \hline \therefore q \end{array}$$

Contoh 1.27

Inferensi berikut:

“Saya belajar dengan giat atau saya menikah tahun depan.
Saya tidak belajar dengan giat. Karena itu, saya menikah tahun depan.”

menggunakan kaidah silogisme disjungtif, atau dapat ditulis dengan cara:

Saya belajar dengan giat atau saya menikah tahun depan.

Saya tidak belajar dengan giat.

∴ Saya menikah tahun depan. ■

5. Simplifikasi

Kaidah ini didasarkan pada tautologi $(p \wedge q) \rightarrow p$, yang dalam hal ini, p dan q adalah hipotesis, sedangkan p adalah konklusi. Kaidah simplifikasi ditulis dengan cara:

$$\frac{p \wedge q}{\therefore p}$$

Contoh 1.28

Penarikan kesimpulan seperti berikut ini:

“Hamid adalah mahasiswa ITB dan mahasiswa Unpar. Karena itu, Hamid adalah mahasiswa ITB.”

menggunakan kaidah simplifikasi, atau dapat juga ditulis dengan cara:

Hamid adalah mahasiswa ITB dan mahasiswa Unpar.

∴ Hamid adalah mahasiswa ITB.

Simplifikasi berikut juga benar:

“Hamid adalah mahasiswa ITB dan mahasiswa Unpar. Karena itu, Hamid adalah mahasiswa Unpar”

karena urutan proposisi di dalam konjungsi $p \wedge q$ tidak mempunyai pengaruh apa-apanya. ■

6. Penjumlahan

Kaidah ini didasarkan pada tautologi $p \rightarrow (p \vee q)$. Kaidah penjumlahan ditulis dengan cara:

$$\therefore \frac{p}{p \vee q}$$

Contoh 1.29

Penarikan kesimpulan seperti berikut ini:

“Taslim mengambil kuliah Matematika Diskrit. Karena itu, Taslim mengambil kuliah Matematika Diskrit atau mengulang kuliah Algoritma.”

menggunakan kaidah penjumlahan, atau dapat juga ditulis dengan cara:

Taslim mengambil kuliah Matematika Diskrit.

∴ Taslim mengambil kuliah Matematika Diskrit atau
mengulang kuliah Algoritma

7. Konjungsi

Kaidah ini didasarkan pada tautologi $((p) \wedge (q)) \rightarrow (p \wedge q)$. Kaidah konjungsi ditulis dengan cara:

$$\begin{array}{c} p \\ q \\ \hline \therefore p \wedge q \end{array}$$

Contoh 1.30

Penarikan kesimpulan seperti berikut ini:

“Taslim mengambil kuliah Matematika Diskrit. Taslim mengulang kuliah Algoritma. Karena itu, Taslim mengambil kuliah Matematika Diskrit dan mengulang kuliah Algoritma”

menggunakan kaidah konjungsi, atau dapat juga ditulis dengan cara:

Taslim mengambil kuliah Matematika Diskrit.

Taslim mengulang kuliah Algoritma.

∴ Taslim mengambil kuliah Matematika Diskrit dan
mengulang kuliah Algoritma.

1.11 Argumen

Argumen adalah suatu deret proposisi yang dituliskan sebagai

$$\begin{array}{c} p_1 \\ p_2 \\ \vdots \\ p_n \\ \hline \therefore q \end{array}$$

yang dalam hal ini, p_1, p_2, \dots, p_n disebut hipotesis (atau premis), dan q disebut konklusi.

Argumen ada yang sahih (*valid*) dan palsu (*invalid*). Catatlah bahwa kata “valid” tidak sama maknanya dengan “benar” (*true*).

Definisi 1.9. Sebuah argumen dikatakan sahif jika konklusi benar bilamana semua hipotesisnya benar; sebaliknya argumen dikatakan palsu (*fallacy* atau *invalid*).

Jika argumen sahih, maka kadang-kadang kita mengatakan bahwa secara logika konklusi mengikuti hipotesis atau sama dengan memperlihatkan bahwa implikasi

$$(p_1 \wedge p_2 \wedge \dots \wedge p_n) \rightarrow q$$

adalah benar (yaitu, sebuah tautologi). Argumen yang palsu menunjukkan proses penalaran yang tidak benar.

Contoh 1.31

Perlihatkan bahwa argumen berikut:

“Jika air laut surut setelah gempa di laut, maka tsunami datang.
Air laut surut setelah gempa di laut. Karena itu tsunami datang.”

adalah sahih.

Penyelesaian:

Misalkan p adalah proposisi “Air laut surut setelah gempa di laut” dan q adalah proposisi “tsunami datang”. Maka, argumen di dalam soal dapat ditulis sebagai:

$$\begin{array}{c} p \rightarrow q \\ p \\ \hline \therefore q \end{array}$$

Ada dua cara yang dapat digunakan untuk membuktikan kesahihan argumen ini. Keduanya menggunakan tabel kebenaran.

Cara 1: Bentuklah tabel kebenaran untuk p , q , dan $p \rightarrow q$

Tabel 1.15 Tabel kebenaran untuk p , q , dan $p \rightarrow q$

p	q	$p \rightarrow q$	
T	T	T	(baris 1)
T	F	F	(baris 2)
F	T	T	(baris 3)
F	F	T	(baris 4)

sahih jika semua hipotesisnya benar, maka konklusinya benar. Kita periksa apabila hipotesis p dan $p \rightarrow q$ benar, maka konklusi q juga benar sehingga argumen dikatakan benar. Periksa di Tabel 1.15, p dan $p \rightarrow q$ benar secara bersama-sama pada baris 1. Pada baris 1 ini q juga benar. Jadi, argumen yang berbentuk modus ponen di atas sahih.

Cara 2: Perlihatkan dengan tabel kebenaran apakah

$$[p \wedge (p \rightarrow q)] \rightarrow q$$

merupakan tautologi. Tabel 1.16 memperlihatkan bahwa $[p \wedge (p \rightarrow q)] \rightarrow q$ suatu tautologi, sehingga argumen dikatakan sahih.

Tabel 1.16 $[p \wedge (p \rightarrow q)] \rightarrow q$ adalah tautologi

p	q	$p \rightarrow q$	$p \wedge (p \rightarrow q)$	$[p \wedge (p \rightarrow q)] \rightarrow q$
T	T	T	T	T
T	F	F	F	T
F	T	T	F	T
F	F	T	F	T

Perhatikanlah bahwa penarikan kesimpulan di dalam argumen ini menggunakan modus ponen. Jadi, kita juga telah memperlihatkan bahwa modus ponen adalah argumen yang sahih. ■

Contoh 1.32

Perlihatkan bahwa penalaran pada argumen berikut:

“Jika air laut surut setelah gempa di laut, maka tsunami datang.
Tsunami datang. Jadi, air laut surut setelah gempa di laut”

tidak benar, dengan kata lain argumennya palsu.

Penyelesaian:

Argumen di atas berbentuk

$$\begin{array}{c} p \rightarrow q \\ q \\ \hline \therefore p \end{array}$$

Dari Tabel 1.15 pada Contoh 1.31 tampak bahwa hipotesis q dan $p \rightarrow q$ benar pada baris ke-3, tetapi pada baris 3 ini konklusi p salah. Jadi, argumen tersebut tidak sah atau palsu, sehingga penalaran menjadi tidak benar.

Kita juga bisa menunjukkan dengan Tabel 1.17 bahwa $[q \wedge (p \rightarrow q)] \rightarrow p$ bukan tautologi, sehingga argumen dikatakan tidak sah. ■

Tabel 1.17 $[q \wedge (p \rightarrow q)] \rightarrow p$ bukan tautologi

p	q	$p \rightarrow q$	$q \wedge (p \rightarrow q)$	$[q \wedge (p \rightarrow q)] \rightarrow p$
T	T	T	T	T
T	F	F	F	T
F	T	T	T	F
F	F	T	F	T

Contoh 1.33

Periksa kesahihan argumen berikut ini:

Jika 5 lebih kecil dari 4, maka 5 bukan bilangan prima.
5 tidak lebih kecil dari 4.

$\therefore 5$ adalah bilangan prima

Penyelesaian:

Misalkan p adalah proposisi “5 lebih kecil dari 4” dan q adalah proposisi “5 adalah bilangan prima”. Maka argumen di atas berbentuk:

$$\begin{array}{c} p \rightarrow \sim q \\ \sim p \\ \hline \therefore q \end{array}$$

Tabel 1.18 memperlihatkan tabel kebenaran untuk kedua hipotesis dan konklusi tersebut. Baris ke-3 dan ke-4 pada tabel tersebut adalah baris di mana $p \rightarrow \sim q$ dan $\sim p$ benar secara bersama-sama, tetapi pada baris ke-4 konklusi q salah (meskipun pada baris ke-3 konklusi q benar). Ini berarti argumen tersebut palsu.

Perhatikanlah bahwa meskipun konklusi dari argumen tersebut kebetulan merupakan pernyataan yang benar (“5 adalah bilangan prima” adalah benar), tetapi konklusi dari argumen ini tidak sesuai dengan bukti bahwa argumen tersebut palsu. ■

Tabel 1.18 Tabel kebenaran untuk $p \rightarrow \sim q$, $\sim p$, dan q

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

Contoh 1.34

Periksa kesahihan argumen berikut ini:

Jika 17 adalah bilangan prima, maka 3 tidak habis membagi 17.
3 habis membagi 17.

∴ 17 bukan bilangan prima

Penyelesaian:

Misalkan p adalah proposisi “17 adalah bilangan prima” dan q adalah proposisi “3 habis membagi 17”. Maka argumen di atas berbentuk:

$$\begin{array}{c} p \rightarrow \sim q \\ q \\ \hline \end{array}$$

∴ $\sim p$

Tabel 1.18 digunakan kembali untuk memperlihatkan tabel kebenaran untuk kedua hipotesis dan konklusi tersebut. Baris ke-3 pada tabel tersebut adalah baris di mana hipotesis $p \rightarrow \sim q$ dan q benar secara bersama-sama. Pada baris ke-3 ini, konklusi $\sim p$ juga benar. Ini berarti argumen tersebut sahih.

Perhatikanlah bahwa meskipun argumen tersebut sahih, tetapi konklusi dari argumen tersebut kebetulan merupakan pernyataan yang salah (“17 bukan bilangan prima” adalah salah). Hal ini disebabkan karena premis yang salah (“3 habis membagi 17”) digunakan di dalam argumen, yang mengakibatkan konklusi dari argumen salah. ■

Contoh 1.34 ini memperlihatkan bahwa argumen yang sahih dapat mengarah ke konklusi yang salah jika satu atau lebih dari proposisi salah digunakan di dalam argumen. Moral dari cerita ini adalah bahwa pada suatu argumen yang benar kita tidak mengatakan bahwa konklusinya benar; kita hanya mengatakan bahwa jika kita menjamin hipotesisnya benar, maka kita juga menjamin konklusinya benar.

Contoh 1.35

Periksa kesahihan argumen berikut ini:

Jika saya menyukai Informatika, maka saya belajar sungguh-sungguh.
Saya belajar sungguh-sungguh atau saya gagal.

- Jika saya gagal, maka saya tidak menyukai Informatika.

Penyelesaian:

Misalkan p adalah proposisi “Saya menyukai Informatika” dan q adalah proposisi “Saya belajar sungguh-sungguh”, dan r adalah proposisi “Saya gagal”. Maka argumen di atas berbentuk:

$$\begin{array}{c} p \rightarrow q \\ q \vee r \\ \hline \therefore r \rightarrow \neg p \end{array}$$

Tabel kebenaran untuk memeriksa kesahihan argumen tersebut ditunjukkan pada Tabel 1.19. Baris ke-1, 2, 6 dan 7 adalah baris di mana premis $p \rightarrow q$ dan $q \vee r$ benar secara bersama-sama, tetapi pada baris ke-1 konklusi $r \rightarrow \neg p$ salah (meskipun pada baris yang 2, 6, dan 7 konklusi tersebut benar), sehingga argumen adalah palsu. ■

Tabel 1.19 Tabel kebenaran untuk $p \rightarrow q$, $q \vee r$, dan $r \rightarrow \neg p$

p	q	r	$p \rightarrow q$	$q \vee r$	$\neg p$	$r \rightarrow \neg p$
T	T	T	T	T	F	F
T	T	F	T	T	F	T
T	F	T	F	T	F	F
T	F	F	F	T	F	T
F	T	T	T	F	T	T
F	T	F	T	T	T	T
F	F	T	T	T	T	T
F	F	F	T	F	T	T

1.12 Aksioma, Teorema, *Lemma*, *Corollary*

Di dalam matematika maupun ilmu komputer, kita sering menemukan kata-kata seperti aksioma, teorema, *lemma*, dan *corolarry*.

Aksioma adalah proposisi yang diasumsikan benar. Aksioma tidak memerlukan pembuktian kebenaran lagi.

Contoh-contoh aksioma:

- (a) Untuk semua bilangan real x dan y , berlaku $x + y = y + x$ (hukum komutatif penjumlahan).
- (b) Jika diberikan dua buah titik yang berbeda, maka hanya ada satu garis lurus yang melalui dua buah titik tersebut.

Teorema adalah proposisi yang sudah terbukti benar. Bentuk khusus dari teorema adalah *lemma* dan *corolarry*. **Lemma** adalah teorema sederhana yang digunakan dalam pembuktian teorema lain. *Lemma* biasanya tidak menarik namun berguna pada pembuktian proposisi yang lebih kompleks, yang dalam hal ini pembuktian tersebut dapat lebih mudah dimengerti bila menggunakan sederetan *lemma*, setiap *lemma* dibuktikan secara individual [ROS03]. **Corollary** adalah teorema yang dapat dibentuk langsung dari teorema yang telah dibuktikan, atau dapat dikatakan *corollary* adalah teorema yang mengikuti dari teorema lain.

Contoh-contoh teorema:

- (a) Jika dua sisi dari sebuah segitiga sama panjang, maka sudut yang berlawanan dengan sisi tersebut sama besar.
- (b) Untuk semua bilangan real x , y , dan z , jika $x \leq y$ dan $y \leq z$, maka $x \leq z$ (hukum transitif).

Contoh *corollary*:

Jika sebuah segitiga adalah sama sisi, maka segitiga tersebut sama sudut. *Corollary* ini mengikuti teorema (a) di atas.

Contoh *lemma*:

Jika n adalah bilangan bulat positif, maka $n - 1$ bilangan positif atau $n - 1 = 0$.

1.13 Ragam Contoh Soal dan Penyelesaian

Untuk lebih memantapkan pemahaman terhadap materi logika proposisi, berikut ini diberikan sejumlah soal dan penyelesaiannya.

Contoh 1.36

Diberikan pernyataan “Tidak benar bahwa dia belajar Algoritma tetapi tidak belajar Matematika”.

- Nyatakan pernyataan di atas dalam notasi simbolik (ekspresi logika)
- Berikan pernyataan yang ekivalen secara logika dengan pernyataan tersebut
(Petunjuk: gunakan hukum De Morgan)

Penyelesaian:

Misalkan

p : Dia belajar Algoritma

q : Dia belajar Matematika

maka,

(a) $\sim(p \wedge \sim q)$

(b) $\sim(p \wedge \sim q) \Leftrightarrow \sim p \vee q$ (Hukum De Morgan)

dengan kata lain: “Dia tidak belajar Algoritma atau belajar Matematika”

■

Contoh 1.37

Untuk menerangkan mutu sebuah hotel, misalkan p : Pelayanannya baik, dan q : Tarif kamarnya murah, r : Hotelnya berbintang tiga.

Terjemahkan proposisi-proposisi berikut dalam notasi simbolik (menggunakan p, q, r):

- Tarif kamarnya murah, tapi pelayanannya buruk.
- Tarif kamarnya mahal atau pelayanannya baik, namun tidak keduanya.
- Salah bahwa hotel berbintang tiga berarti tarif kamarnya murah dan pelayanannya buruk.

Penyelesaian:

(a) $q \wedge \sim p$

(b) $\sim q \oplus p$

(c) $\sim(r \rightarrow (q \wedge \sim p))$

■

Contoh 1.39

Nyatakan pernyataan berikut “Anda tidak dapat terdaftar sebagai pemilih dalam Pemilu jika anda berusia di bawah 17 tahun kecuali kalau anda sudah menikah”.

Penyelesaian:

Misalkan

p : Anda berusia di bawah 17 tahun.

q : Anda sudah menikah.

r : Anda dapat terdaftar sebagai pemilih dalam Pemilu.

maka pernyataan di atas dapat ditulis sebagai

$$(p \wedge \sim q) \rightarrow \sim r$$

■

Contoh 1.40

Diberikan pernyataan “Perlu memiliki *password* yang sah agar anda bisa *log on* ke *server*”

- Nyatakan pernyataan di atas dalam bentuk proposisi “jika p , maka q ”.
- Tentukan ingkaran, konvers, invers, dan kontraposisi dari pernyataan tersebut.

Penyelesaian:

Misalkan

p : Anda bisa *log on* ke *server*

q : Memiliki *password* yang sah

maka

- Jika anda bisa *log on* ke *server* maka anda memiliki *password* yang sah
- 1) Ingkaran:
“Anda bisa *log on* ke *server* dan anda tidak memiliki *password* yang sah”
- 2) Konvers:
“Jika anda memiliki *password* yang sah maka anda bisa *log on* ke *server*”
- 3) Invers:
“Jika anda tidak bisa *log on* ke *server* maka anda tidak memiliki *password* yang sah”
- 4) Kontraposisi :
“Jika anda tidak memiliki *password* yang sah maka anda tidak bisa *log on* ke *server*”

■

Contoh 1.41

Diberikan pernyataan “Untuk mendapatkan satu kupon undian, Anda cukup membeli dua produk senilai Rp 50.000,-”.

- Nyatakan pernyataan di atas dalam bentuk proposisi “jika p , maka q ”.
- Tentukan ingkaran, konvers, invers, dan kontraposisi dari pernyataan tersebut.

Penyelesaian:

Misalkan

- p : Anda mendapatkan satu kupon undian
 q : Anda membeli dua produk senilai Rp 50.000,-

maka

- (a) Jika Anda membeli dua produk senilai Rp. 50.000,-, maka Anda mendapatkan satu kupon undian.
- (b) 1) Ingkaran:
“Anda membeli dua produk senilai Rp. 50.000,- dan Anda tidak mendapatkan satu kupon undian.”
- 2) Konvers:
“Jika Anda mendapatkan satu kupon undian, maka Anda membeli dua produk Rp. 50.000,-”
- 3) Invers:
“Jika Anda tidak membeli dua produk senilai Rp. 50.000,-, maka Anda tidak mendapatkan satu kupon undian.”
- 4) Kontraposisi :
“Jika Anda tidak mendapatkan satu kupon undian, maka Anda tidak membeli dua produk senilai Rp. 50.000,-”

■

Contoh 1.42

Tentukan ingkaran dan kontraposisi dari pernyataan berikut: “Dia tidak pergi ke kampus maupun ke perpustakaan bilamana hari ini hujan”.

Penyelesaian:

Misalkan

- p : Dia pergi ke kampus
 q : Dia pergi ke perpustakaan
 r : Hari ini hujan

Maka kalimat di atas dapat dituliskan dalam bentuk:

$$r \rightarrow (\sim p \wedge \sim q)$$

Untuk menentukan ingkarannya, terapkan hukum-hukum logika sebagai berikut:

$$\begin{aligned} \sim(r \rightarrow (\sim p \wedge \sim q)) &\Leftrightarrow \sim(\sim r \vee (\sim p \wedge \sim q)) \\ &\Leftrightarrow r \wedge \sim(\sim p \wedge \sim q) \\ &\Leftrightarrow r \wedge (p \vee q) \end{aligned}$$

Jadi ingkarannya adalah

“Hari ini hujan, dan dia pergi ke kampus atau ke perpustakaan”

Untuk menentukan kontraposisinya, terapkan hukum-hukum logika sebagai berikut:

$$\sim(\sim p \wedge \sim q) \rightarrow \sim r \Leftrightarrow (p \vee q) \rightarrow \sim r$$

Jadi kontraposisinya adalah

“Jika dia pergi ke kampus atau ke perpustakaan, maka hari ini tidak hujan” ■

Contoh 1.43

Tunjukkan bahwa $[\sim p \wedge (p \vee q)] \rightarrow q$ adalah tautologi.

Penyelesaian:

Buat tabel kebenaran sebagai berikut:

Tabel 1.21 Tabel kebenaran $[\sim p \wedge (p \vee q)] \rightarrow q$

p	q	$\sim p$	$p \vee q$	$\sim p \wedge (p \vee q)$	$[\sim p \wedge (p \vee q)] \rightarrow q$
T	T	F	T	F	T
T	F	F	T	F	T
F	T	T	T	T	T
F	F	T	F	F	T

Dari Tabel 1.21 terlihat bahwa $[\sim p \wedge (p \vee q)] \rightarrow q$ adalah tautologi. ■

Contoh 1.44

Sebagian besar orang percaya bahwa harimau Jawa sudah lama punah. Tetapi, pada suatu hari Amir membuat pernyataan-pernyataan kontroversial sebagai berikut:

- Saya melihat harimau di hutan.
- Jika saya melihat harimau di hutan, maka saya juga melihat srigala.

Misalkan kita diberitahu bahwa Amir kadang-kadang suka berbohong dan kadang-kadang jujur. Gunakan tabel kebenaran untuk memeriksa apakah Amir benar-benar melihat harimau di hutan?

Penyelesaian:

Misalkan

$$\begin{aligned}p &: \text{Amir melihat harimau di hutan} \\q &: \text{Amir melihat srigala}\end{aligned}$$

Pernyataan untuk soal (a) adalah p sedangkan pernyataan untuk (b) adalah $p \rightarrow q$. Tabel kebenaran untuk p dan $p \rightarrow q$ ditunjukkan pada Tabel 1.22.

Tabel 1.22 Tabel kebenaran p dan $p \rightarrow q$

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

Bila Amir dianggap berbohong, maka apa yang dikatakan Amir itu keduanya salah, atau bila dia dianggap jujur maka apa yang dikatakan Amir itu keduanya benar. Tabel 1.22 menunjukkan bahwa mungkin bagi q dan $p \rightarrow q$ benar, tetapi tidak mungkin keduanya salah. Ini berarti Amir mengatakan yang sejurnya, dan kita menyimpulkan bahwa Amir memang benar melihat harimau di hutan.

Anda juga dapat menjawab soal ini tanpa menggunakan tabel kebenaran. Tinjau dua kasus. Kasus pertama, Amir berbohong, maka apa yang dikatakan Amir itu keduanya salah. Ini berarti p salah, dengan demikian implikasi $p \rightarrow q$ pasti benar apa pun nilai kebenaran pernyataan q . Ini jelas kontradiksi. Jadi, pastilah Amir benar (kasus kedua), yang berarti Amir memang benar melihat harimau di hutan. ■

Contoh 1.45

Periksa kesahihan argumen berikut:

$$\begin{array}{c} p \rightarrow \neg q \\ \neg r \rightarrow p \\ \hline \therefore r \end{array}$$

Contoh argumen nyatanya adalah sebagai berikut:

“Jika saya pulang kampung, maka saya tidak bisa mengikuti ujian susulan.
Jika saya tidak lulus ujian, maka saya pulang kampung. Tetapi saya bisa
mengikuti ujian susulan. Oleh karena itu saya lulus ujian.”

Penyelesaian:

Tabel kebenaran untuk memeriksa kesahihan argumen tersebut ditunjukkan pada Tabel 1.23. Baris 5 adalah baris di mana premis $p \rightarrow \neg q$, $\neg r \rightarrow p$, dan q benar secara bersama-sama, dan pada baris ini juga konklusi r benar, sehingga argumen tersebut sahih. ■

Tabel 1.23 Tabel kebenaran untuk $p \rightarrow \neg q$, $\neg r \rightarrow p$, q , dan r

p	q	r	$\neg q$	$p \rightarrow \neg q$	$\neg r$	$\neg r \rightarrow p$
T	T	T	F	F	F	T
T	T	F	F	F	T	T
T	F	T	T	T	F	T

T	F	T	F	T	F	T	T
F	T	T	F	F	T	F	T
F	T	F	T	F	T	T	F
F	F	F	T	T	T	F	T
F	F	F	F	T	T	T	F

Soal Latihan

1. Tentukan pernyataan manakah di bawah ini yang merupakan proposisi? Tentukan nilai kebenaran dari pernyataan yang merupakan proposisi.
 - (a) $3 + 15 = 17$
 - (b) Untuk beberapa bilangan bulat n , $600 = n \cdot 15$
 - (c) $x + y = y + x$ untuk setiap pasangan bilangan riil x dan y
 - (d) Setiap bilangan bulat genap lebih dari empat merupakan penjumlahan dua bilangan prima
 - (e) Tidak ada orangutan hidup di kota
 - (f) Ambil 5 buah buku di atas meja
 - (g) $4 + x = 5$
2. Misalkan p adalah "Iwan bisa berbahasa Inggris", q adalah "Iwan bisa berbahasa Jerman" dan r adalah "Iwan bisa berbahasa Perancis". Terjemahkan kalimat majemuk berikut ke dalam notasi simbolik:
 - (a) Iwan bisa berbahasa Inggris atau Jerman
 - (b) Iwan bisa berbahasa Jerman tetapi tidak bahasa Perancis
 - (c) Iwan bisa berbahasa Inggris atau bahasa Jerman, atau dia tidak bisa berbahasa Perancis atau bahasa Jerman
 - (d) Tidak benar bahwa Iwan bisa berbahasa Inggris atau bahasa Perancis
 - (e) Tidak benar bahwa Iwan bisa berbahasa Inggris atau bahasa Perancis tetapi tidak bahasa Jerman
 - (f) Tidak benar bahwa Iwan tidak bisa berbahasa Inggris, Perancis, maupun Jerman
3. Untuk menerangkan karakteristik mata kuliah X , misalkan p : "Kuliahnya menarik", dan q : "Dosenya enak", r : "Soal-soal ujiannya mudah". Terjemahkan proposisi-proposisi berikut dalam notasi simbolik (menggunakan p , q , r):
 - (d) Kuliahnya tidak menarik, dosenya tidak enak, dan soal-soal ujiannya tidak mudah.
 - (e) Kuliahnya menarik atau soal-soal ujiannya tidak mudah, namun tidak keduanya.
 - (f) Salah bahwa kuliahnya menarik berarti dosenya enak dan soal-soal ujiannya mudah.
4. Diberikan pernyataan "Tidak benar bahwa penjualan merosot maupun pendapatan tidak naik"
 - (a) Nyatakan pernyataan di atas dalam notasi simbolik.

- (b) Berikan pernyataan yang ekivalen secara logika dengan pernyataan tersebut (petunjuk: gunakan Hukum de Morgan).
5. Untuk menerangkan mutu sebuah perangkat lunak yang beredar di pasaran, kita misalkan p adalah pernyataan “Tampilan antarmukanya (*interface*) menarik”, q pernyataan “Cara pengoperasiannya mudah”, dan r pernyataan “Perangkat lunaknya bagus sekali”. Tuliskan pernyataan berikut dalam bentuk simbolik:
- Tidak benar bahwa tampilan antarmukanya menarik maupun cara pengoperasiannya sulit.
 - Tampilan antarmukanya menarik atau cara pengoperasiannya mudah, namun tidak keduanya.
 - Perangkat lunak yang bagus sekali selalu berarti bahwa tampilan antarmukanya menarik dan cara pengoperasiannya mudah, begitu sebaliknya.
6. Nyatakan proposisi berikut dalam notasi simbolik:
- Setiap dokumen dipindai dengan program anti virus bilamana dokumen berasal dari sistem yang tidak dikenal.
 - Setiap dokumen yang berasal dari sistem yang tidak dikenal tetapi ia tidak dipindai dengan program anti virus.
 - Perlu memindai dokumen dengan program anti virus bilamana ia berasal dari sistem yang tidak dikenal.
 - Bila pesan tidak dikirim dari sistem yang tidak diekナル, ia tidak dipindai dengan program anti virus.
7. Misalkan p adalah “Hari ini adalah Hari Rabu”, q adalah “Hujan turun” dan r adalah “Hari ini panas”. Terjemahkan notasi simbolik ini dengan kata-kata:
- $p \vee q$
 - $\sim p \wedge (q \vee r)$
 - $\sim(p \vee q) \wedge r$
 - $(p \wedge q) \wedge \sim(r \vee p)$
 - $(p \wedge (q \wedge r)) \wedge (r \vee (q \vee p))$
 - $\sim q \rightarrow \sim p$
8. Tuliskan tabel kebenaran untuk setiap proposisi berikut:
- $(p \vee q) \wedge \sim p$
 - $\sim(p \wedge q) \vee (\sim q \vee r)$
 - $(\sim p \vee \sim q) \vee p$
 - $\sim(p \wedge q) (r \wedge \sim p)$
 - $(p \vee q) \rightarrow \sim q$
 - $(\sim q \rightarrow p) \rightarrow (p \rightarrow \sim q)$
9. Nyatakan apakah setiap implikasi berikut benar atau salah:
- Jika $2 + 2 = 4$, maka $3 + 3 = 5$
 - Jika $1 + 1 = 2$, maka Tuhan ada
 - Jika $2 + 2 = 4$, maka 4 adalah bilangan prima
 - Jika $3 < 6$, maka $6 < 2$

10. Nyatakan setiap proposisi berikut menjadi proposisi bersyarat “jika p , maka q ”:
- Dian bisa lulus sarjana apabila ia telah menyelesaikan 144 SKS.
 - Sebuah program hanya bisa dibaca jika ia terstruktur dengan baik.
 - Syarat cukup bagi Lukman untuk mengambil kuliah Algoritma dan Pemrograman adalah ia sudah lulus kuliah Matematika Diskrit.
 - Perlu ada salju agar Hesnu bisa bermain ski.
 - Anda hanya mendapat jaminan barang hanya jika anda mengembalikan kartu garansi kurang dari sebulan sejak pembelian.
 - Untuk mendapat gelar doktor, cukup anda kuliah di Universitas X.
 - Perlu mendaki 100 meter lagi untuk mencapai puncak gunung Semeru.
11. Tentukan *konvers*, *invers*, dan *kontraposisi* dari soal nomor 10 di atas.
12. Nyatakan ingkaran, konvers dan kontraposisi dari implikasi berikut:
- Saya masuk kuliah bilamana ada kuis.
 - Sebuah bilangan positif hanya prima jika ia tidak mempunyai pembagi selain 1 dan dirinya sendiri.
 - Dia pergi ke kampus bilamana hari ini tidak mendung maupun hujan.
 - Sebuah program dikatakan bagus hanya jika waktu eksekusinya singkat atau kebutuhan memorinya sedikit
13. Jika pernyataan $p \rightarrow q$ salah, tentukan nilai pernyataan $(\sim p \vee \sim q) \rightarrow q$
14. Jika pernyataan $p \rightarrow q$ benar, dapatkah anda memastikan nilai pernyataan $\sim p \vee (p \leftrightarrow q)$
15. Manakah dari kalimat berikut yang menyatakan “atau” sebagai *inclusive or* atau *exclusive or*?
- Untuk mengambil kuliah Matematika Diskrit, anda harus sudah mengambil kuliah Kalkulus atau Pengantar Teknologi Informasi
 - Sekolah diliburkan jika banjir melebihi 1 meter atau jika hujan masih belum berhenti
 - Jika anda membeli sepeda motor saat ini, anda mendapat potongan Rp 500.000,- atau *voucher* BBM sebesar 2% dari harga motor.
 - Untuk makan malam, tamu boleh memesan 2 macam sup atau 1 macam bistik.
16. Gunakan tabel kebenaran untuk memperlihatkan hukum distributif $p \wedge (q \vee r) \Leftrightarrow (p \wedge q) \vee (q \wedge r)$.
17. Perlihatkan bahwa $(p \rightarrow q) \rightarrow r$ dan $p \rightarrow (q \rightarrow r)$ tidak ekivalen.

18. Gunakan tabel kebenaran untuk menunjukkan bahwa tiap implikasi berikut adalah tautologi:
- $\sim p \rightarrow (p \rightarrow q)$
 - $\sim(p \rightarrow q) \rightarrow \sim q$
 - $(p \wedge q) \rightarrow (p \rightarrow q)$
19. Gunakan hukum-hukum aljabar proposisi untuk menunjukkan bahwa
(i) $(p \wedge q) \rightarrow (p \vee q)$ dan (ii) $[p \wedge (p \rightarrow q)] \rightarrow q$ keduanya adalah tautologi.
20. Ada sebuah kampung yang penduduknya selalu mengatakan hal yang benar atau selalu bohong. Penduduk kampung hanya memberikan jawaban “ya” atau “tidak” terhadap pertanyaan yang diajukan oleh pendatang. Misalkan anda adalah seorang pendatang yang baru sampai ke kampung tersebut dan hendak pergi ke kampung lain. Anda sedang berada pada sebuah pertigaan jalan. Satu cabang jalan menuju kota, sedangkan cabang jalan lainnya menuju ke jurang, namun anda tidak tahu cabang mana yang menuju ke kota tujuan (tidak ada penunjuk arah). Kebetulan di pertigaan tersebut ada seorang warga kampung sedang berdiri, namanya Z. Sebutkan sebuah pertanyaan yang harus anda ajukan ke warga tersebut untuk menentukan cabang jalan mana yang akan anda ambil?

Petunjuk: Misalkan p adalah pernyataan, “Z selalu mengatakan sebenarnya” dan q pernyataan, “Jalan yang berbelok ke kiri menuju kota”. Formulasikan pernyataan A yang tersusun dari p dan q sedemikian rupa sehingga Z akan menjawab pertanyaan “Apakah A benar” dengan “ya” jika dan hanya jika q benar.

21. Periksalah kesahihan argumen-argumen berikut:
- Jika hari panas, Anton mimisan. Hari tidak panas. Oleh karena itu, Anton tidak mimisan.
 - Jika hari panas, Anton mimisan. Anton tidak mimisan. Oleh karena itu, hari tidak panas.
 - Jika Anton mimisan, maka hari panas. Hari tidak panas. Oleh karena itu, Anton mimisan.
 - Jika hari tidak panas, Anton tidak mimisan. Hari panas. Oleh karena itu, Anton mimisan.
 - Jika Anton tidak mimisan, hari tidak panas. Anton mimisan. Oleh karena itu, hari panas.

22. Periksa kesahihan argumen berikut:

Terlambat mengumpulkan tugas lebih baik daripada tidak ada.
Tiada yang lebih baik daripada mendapat nilai E

∴ Terlambat mengumpulkan tugas lebih baik daripada mendapat nilai E

BAB 2

Himpunan

Pelajarilah semesta ini. Jangan merasa kecewa jika dunia tidak mengenal anda, tetapi kecewalah jika anda tidak mengenal dunia.
(Kong Fu Tse - filusuf China)

Dalam kehidupan sehari-hari kita sering membicarakan objek-objek diskrit, misalnya buku, komputer, mahasiswa, nilai ujian, dan lain-lain. Ilmu Komputer atau Informatika merupakan bidang ilmu yang *mengurusi* objek-objek diskrit. Di dalam komputer digital, objek-objek diskrit ini merupakan data masukan untuk program. Komputer digital diprogram untuk menyimpan, mengolah, dan memanipulasi objek-objek diskrit.

Dalam membicarakan objek diskrit, kita sering berhadapan dengan situasi yang berhubungan dengan sekumpulan objek di dalam suatu kelompok atau kelas, dan kita mengacu objek yang termasuk di dalam suatu kelompok. Misalnya, “semua mahasiswa Teknik Informatika ITB Angkatan 2002” adalah sebuah kelompok yang terdiri atas sejumlah mahasiswa ITB Angkatan 2002 dari Departemen Teknik Informatika.

Terminologi dasar tentang sekumpulan objek diskrit adalah **himpunan**. Himpunan digunakan untuk mengelompokkan objek secara bersama-sama. Himpunan merupakan struktur diskrit fundamental yang mendasari struktur diskrit lainnya seperti relasi, kombinasi, dan graf. Banyak konsep ilmu komputer/informatika yang diajukan dalam terminologi himpunan.

2.1 Definisi Himpunan

Cukup banyak definisi himpunan disebutkan dalam berbagai literatur. Definisi himpunan yang kita pakai di sini diambil dari [LIU85].

DEFINISI 2.1 Himpunan (*set*) adalah kumpulan objek-objek yang *berbeda*.

Objek yang terdapat di dalam himpunan disebut **elemen**, **unsur**, atau **anggota**. Kita katakan bahwa himpunan mengandung elemen-elemennya. Kata “berbeda” di dalam definisi di atas adalah penting (sehingga dicetak miring) untuk menekankan maksud bahwa anggota himpunan tidak boleh sama.

2.2 Penyajian Himpunan

Terdapat banyak cara untuk menyajikan himpunan. Di sini dikemukakan 4 cara penyajian, yaitu mengenumerasi elemen-elemennya, menggunakan simbol-simbol baku, menyatakan syarat keanggotaan, dan menggunakan diagram Venn.

1. Enumerasi

Jika sebuah himpunan terbatas dan tidak terlalu besar, kita bisa menyajikan himpunan dengan cara mengenumerasi, artinya menuliskan semua elemen himpunan yang bersangkutan di antara dua buah tanda kurung kurawal. Biasanya suatu himpunan diberi nama dengan menggunakan huruf kapital maupun dengan menggunakan simbol-simbol lainnya.

Contoh 2.1

Himpunan A yang berisi empat anggota 1, 2, 3, dan 4 dapat ditulis sebagai $A = \{1, 2, 3, 4\}$. Perhatikan bahwa himpunan ditentukan oleh anggota-anggotanya dan bukan pada urutan anggota-anggotanya. Urutan anggota di dalam himpunan tidak mempunyai arti apa-apa. Jadi, kita bisa saja menuliskan A sebagai $A = \{2, 4, 1, 3\}$ atau $A = \{4, 3, 2, 1\}$. Karena itu, beberapa literatur menambahkan definisi himpunan sebagai kumpulan objek tak-terurut (*unordered collection*). ■

Contoh 2.2

Himpunan B yang berisi lima bilangan genap positif pertama adalah $B = \{4, 6, 8, 10\}$. ■

Selain masalah urutan, hal penting lain yang perlu diperhatikan adalah penulisan anggota yang berulang. Adalah berlebihan menuliskan A sebagai $A = \{1, 2, 2, 3, 4\}$,

karena dari Definisi 2.1 disebutkan bahwa himpunan mengandung anggota yang berbeda. Jadi, kalau kita ingin menyebutkan kumpulan buku di dalam perpustakaan, yang mana buku berjudul x ada 2 buah, sedangkan buku berjudul y dan z masing-masing 1 buah, maka himpunan $\{x, x, y, z\}$ adalah cara penulisan yang berlebihan. Kita bisa menuliskan himpunan tersebut menjadi $\{x_1, x_2, y, z\}$, yang dalam hal ini x_1 adalah *copy* 1 dari buku x dan x_2 adalah *copy* 2 dari buku x . Buku x_1 dan x_2 adalah objek yang berbeda di dalam himpunan yang terakhir.

Himpunan yang anggotanya boleh barulang adalah himpunan-ganda (multiset). Himpunan-ganda akan dibahas di dalam upa-bab tersendiri pada bagian akhir bab ini. Jika tidak disebutkan lain, maka himpunan yang kita maksudkan di dalam pembahasan adalah himpunan yang anggotanya berbeda satu sama lain.

Contoh 2.3

Meskipun himpunan biasanya digunakan untuk mengelompokkan objek yang mempunyai sifat mirip, tetapi dari definisi himpunan kita mengetahui bahwa sah-sah saja elemen-elemen di dalam himpunan tidak mempunyai hubungan satu sama lain, asalkan *berbeda*. Sebagai contoh, $\{\text{kucing}, a, \text{Amir}, 10, \text{paku}\}$ adalah himpunan yang terdiri dari lima elemen, yaitu kucing, a , Amir, 10, dan paku. ■

Contoh 2.4

Contoh-contoh himpunan lainnya:

$$\begin{aligned} R &= \{ a, b, \{a, b, c\}, \{a, c\} \} \\ C &= \{a, \{a\}, \{\{a\}\} \} \\ K &= \{ \{ \} \} \end{aligned}$$

Perhatikan dari Contoh 2.4, C adalah himpunan yang terdiri dari 3 elemen, yaitu a , $\{a\}$, dan $\{\{a\}\}$. Contoh 2.4 memperlihatkan bahwa suatu himpunan dapat merupakan anggota himpunan lain. Perhatikan juga bahwa K hanya berisi satu elemen, yaitu $\{\}$. Lebih lanjut, $\{\}$ disebut himpunan kosong, sering dilambangkan dengan \emptyset (lihat penjelasannya di dalam upabab 2.4).

Untuk menuliskan himpunan dengan jumlah anggota yang besar dan telah memiliki pola tertentu dapat dilakukan dengan menggunakan tanda '...' (*ellipsis*).

Contoh 2.5

Himpunan alfabet ditulis sebagai $\{ a, b, c, \dots, x, y, z \}$, dan himpunan 100 buah bilangan asli pertama ditulis sebagai $\{1, 2, \dots, 100\}$. Catatan: beberapa literatur menyebutkan bilangan asli (*natural numbers*) dimulai 0.

Untuk menuliskan himpunan yang tidak berhingga banyak anggotanya, kita dapat juga menggunakan tanda '...' (*ellipsis*), seperti pada Contoh 2.6 berikut.

Contoh 2.6

Himpunan bilangan bulat positif ditulis sebagai $\{1, 2, 3, \dots\}$, sedangkan himpunan bilangan bulat ditulis sebagai $\{\dots, -2, -1, 0, 1, 2, \dots\}$. ■

Terhadap suatu himpunan, suatu objek dapat menjadi anggota atau bukan anggota himpunan tersebut. Untuk menyatakan keanggotaan tersebut digunakan notasi berikut:

- $x \in A$ untuk menyatakan x merupakan anggota himpunan A ; dan
 - $x \notin A$ untuk menyatakan x bukan merupakan anggota himpunan A .
-

Contoh 2.7

Misalkan $A = \{1, 2, 3, 4\}$, $R = \{a, b, \{a, b, c\}, \{a, c\}\}$, dan $K = \{\{\}\}$, maka

- $3 \in A$
 - $5 \notin A$
 - $\{a, b, c\} \in R$

 - $\{a\} \notin R$
 - $a \in R$
 - $\{\} \in K$
-

Contoh 2.8

Bila $P_1 = \{a, b\}$, $P_2 = \{\{a, b\}\}$, dan $P_3 = \{\{\{a, b\}\}\}$, maka

- $a \in P_1$
 - $a \notin P_2$
 - $P_1 \in P_2$
 - $P_1 \notin P_3$
 - $P_2 \in P_3$
-

2. Simbol-simbol Baku

Beberapa himpunan yang khusus dituliskan dengan simbol-simbol yang sudah baku. Terdapat sejumlah simbol baku yang berbentuk huruf tebal (*boldface*) yang biasa digunakan untuk mendefinisikan himpunan yang sering digunakan, antara lain:

P = himpunan bilangan bulat positif = { 1, 2, 3, ... }

N = himpunan bilangan asli = { 1, 2, ... }

Z = himpunan bilangan bulat = { ..., -2, -1, 0, 1, 2, ... }

Q = himpunan bilangan rasional

R = himpunan bilangan riil

C = himpunan bilangan kompleks

Kadang-kadang kita berhubungan dengan himpunan-himpunan yang semuanya merupakan bagian dari sebuah himpunan yang universal. Himpunan yang universal ini disebut **semesta** dan disimbolkan dengan U . Himpunan U harus diberikan secara eksplisit atau diarahkan berdasarkan pembicaraan. Sebagai contoh, misalnya $U = \{1, 2, 3, 4, 5\}$ dan A adalah himpunan bagian dari U , dengan $A = \{1, 3, 5\}$.

3. Notasi Pembentuk Himpunan

Cara lain menyajikan himpunan adalah dengan notasi pembentuk himpunan (*set builder*). Dengan cara penyajian ini, himpunan dinyatakan dengan menulis syarat yang harus dipenuhi oleh anggotanya.

Notasi: { x | syarat yang harus dipenuhi oleh x }

Aturan yang digunakan dalam penulisan syarat keanggotaan:

- Bagian di kiri tanda " | " melambangkan elemen himpunan.
- Tanda " | " dibaca *dimana* atau *sedemikian sehingga*.
- Bagian di kanan tanda " | " menunjukkan syarat keanggotaan himpunan.
- Setiap tanda ',' di dalam syarat keanggotaan dibaca sebagai *dan*

Contoh 2.9

- (i) A adalah himpunan bilangan bulat positif yang kecil dari 5, dinyatakan sebagai
$$A = \{ x \mid x \text{ adalah himpunan bilangan bulat positif lebih kecil dari } 5 \}$$

atau dalam notasi yang lebih ringkas:

$$A = \{ x \mid x \in \mathbb{P}, x < 5 \}$$

yang sama dengan $A = \{1, 2, 3, 4\}$.

- (ii) B adalah himpunan bilangan genap positif yang lebih kecil atau sama dengan 8, dinyatakan sebagai
$$B = \{ x \mid x \text{ adalah himpunan bilangan genap positif lebih kecil atau sama dari } 8 \}$$

atau dalam notasi yang lebih ringkas:

$$B = \{ x \mid x/2 \in \mathbb{P}, 2 \leq x \leq 8 \}$$

yang sama dengan $B = \{2, 4, 6, 8\}$

(iii) Notasi pembentuk himpunan sangat berguna untuk menyajikan himpunan yang anggota-anggotanya tidak mungkin dienumerasikan. Misalnya \mathbf{Q} adalah himpunan bilangan rasional, dinyatakan sebagai

$$\mathbf{Q} = \{ a/b \mid a, b \in \mathbf{Z}, b \neq 0 \}$$

(iv) M adalah himpunan mahasiswa yang mengambil kuliah Matematika Diskrit,
 $M = \{ x \mid x \text{ adalah mahasiswa yang mengambil kuliah Matematika Diskrit} \}$

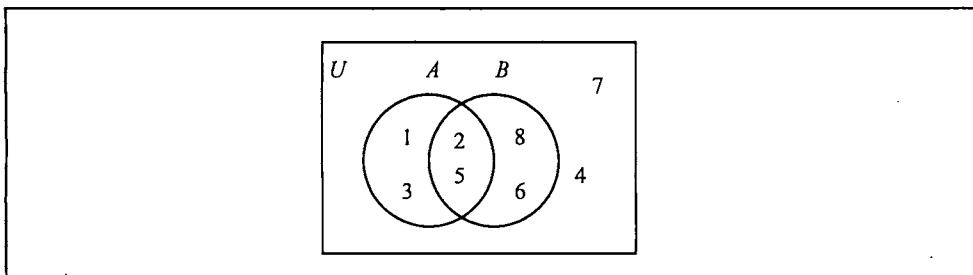


4. Diagram Venn

Diagram Venn menyajikan himpunan secara grafis. Cara penyajian himpunan ini diperkenalkan oleh matematikawan Inggris yang bernama John Venn pada tahun 1881. Di dalam diagram Venn, himpunan semesta (U) digambarkan sebagai suatu segi empat sedangkan himpunan lainnya digambarkan sebagai lingkaran di dalam segi empat tersebut. Anggota-anggota suatu himpunan berada di dalam lingkaran, sedangkan anggota himpunan lain di dalam lingkaran yang lain pula. Ada kemungkinan dua himpunan mempunyai anggota yang sama, dan hal ini digambarkan dengan lingkaran yang saling beririsan. Anggota U yang tidak termasuk di dalam himpunan manapun digambarkan di luar lingkaran.

Contoh 2.10

Misalkan $U = \{1, 2, \dots, 7, 8\}$, $A = \{1, 2, 3, 5\}$ dan $B = \{2, 5, 6, 8\}$. Ketiga himpunan tersebut digambarkan dengan diagram Venn pada Gambar 2.1. Perhatikan bahwa A dan B mempunyai anggota bersama, yaitu 2 dan 5. Anggota U yang lain, yaitu 7 dan 4 tidak termasuk di dalam himpunan A dan B .



Gambar 2.1 Diagram Venn untuk Contoh 1.10

2.3 Kardinalitas

DEFINISI 2.2 Sebuah himpunan dikatakan **berhingga** (*finite set*) jika terdapat n elemen berbeda (*distinct*) yang dalam hal ini n adalah bilangan bulat tak-negatif. Sebaliknya himpunan tersebut dinamakan **tak-berhingga** (*infinite set*).

Misalkan A merupakan himpunan berhingga, maka jumlah elemen berbeda di dalam A disebut **kardinal** dari himpunan A .

Notasi: $n(A)$ atau $|A|$

Catatan:

Di dalam buku ini, kita menggunakan notasi $|A|$ untuk menyatakan kardinalitas himpunan.

Contoh 2.11

Di bawah ini adalah contoh-contoh himpunan berhingga:

- (i) $A = \{x \mid x \text{ merupakan bilangan prima yang lebih kecil dari } 20\}$, maka $|A| = 8$, dengan elemen-elemen A adalah 2, 3, 5, 7, 11, 13, 17, 19
- (ii) $B = \{\text{kucing}, a, \text{Amir}, 10, \text{paku}\}$ maka $|B| = 5$, dengan elemen-elemen B (yang berbeda) adalah kucing, a , Amir, 10, dan paku.
- (iii) $C = \{a, \{a\}, \{\{a\}\}\}$, maka $|C| = 3$, dengan elemen-elemen A (yang berbeda) adalah a , $\{a\}$, dan $\{\{a\}\}$.
- (iv) $D = \{x \mid x \text{ adalah faktor dari } 12\}$, maka $|D| = 6$, dengan elemen-elemen D adalah 1, 2, 3, 4, 6, dan 12.
- (v) $E = \{x \mid x \text{ adalah bilangan bulat positif kurang dari } 1\}$, maka $|E| = 0$, karena tidak ada bilangan positif yang kurang dari 1.
- (vi) $F = \{x \mid x \text{ adalah kucing di Bandung}\}$, ini juga adalah himpunan berhingga, meskipun kita sangat sulit menghitung jumlah kucing di Bandung, tetapi ada jumlah tertentu yang berhingga kucing di Bandung.

Himpunan yang tidak berhingga mempunyai kardinal tidak berhingga pula. Sebagai contoh, himpunan bilangan riil mempunyai jumlah anggota tidak berhingga, maka $|\mathbb{R}| = \infty$, begitu juga himpunan bilangan bulat tak-negatif, himpunan garis yang melalui titik pusat koordinat, himpunan titik di sepanjang garis $y = 2x + 3$, dan lain-lain.

2.4 Himpunan Kosong

DEFINISI 2.2. Himpunan yang tidak memiliki satupun elemen atau himpunan dengan kardinal = 0 disebut **himpunan kosong (empty set)**.

Notasi : \emptyset atau { }

Contoh 2.12

- (i) $E = \{ x \mid x < x \}$, maka $|E| = 0$
- (ii) $P = \{ \text{orang Indonesia yang pernah ke bulan} \}$, maka $|P| = 0$
- (iii) $A = \{x \mid x \text{ adalah akar-akar persamaan kuadrat } x^2 + 5x + 10 = 0\}$, $|A| = 0$

Perhatikan bahwa himpunan $\{\{ \}\}$ dapat juga ditulis sebagai $\{\emptyset\}$, begitu pula himpunan $\{\{ \}, \{\{ \}\}\}$ dapat juga ditulis sebagai $\{\emptyset, \{\emptyset\}\}$. Perhatikan juga bahwa $\{\emptyset\}$ bukan himpunan kosong karena ia memuat satu elemen yaitu \emptyset .

Istilah seperti *kosong*, *hampa*, *nihil*, ketiganya mengacu pada himpunan yang tidak mengandung elemen, tetapi istilah *nol* tidak sama dengan ketiga istilah di atas, sebab nol menyatakan sebuah bilangan tertentu.

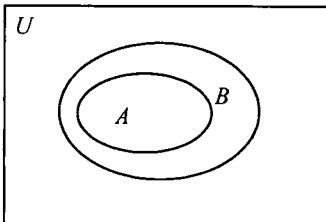
2.5 Himpunan Bagian (*Subset*)

Sebuah himpunan dapat merupakan bagian dari himpunan lain. Anggota yang dikandung di dalam himpunan tersebut juga terkandung di dalam himpunan yang lain.

DEFINISI 2.3. Himpunan A dikatakan **himpunan bagian (subset)** dari himpunan B jika dan hanya jika setiap elemen A merupakan elemen dari B . Dalam hal ini, B dikatakan **superset** dari A .

Notasi: $A \subseteq B$

Dengan menggunakan diagram Venn, $A \subseteq B$ lebih mudah dimengerti maksudnya. $A \subseteq B$ digambarkan dengan diagram Venn pada Gambar 2.2.



Gambar 2.2 Diagram Venn untuk $A \subseteq B$

Contoh 2.13

- (i) $\{1, 2, 3\} \subseteq \{1, 2, 3, 4, 5\}$
- (ii) $\{1, 2, 3\} \subseteq \{1, 2, 3\}$
- (iii) $\mathbf{N} \subseteq \mathbf{Z} \subseteq \mathbf{R} \subseteq \mathbf{C}$
- (iv) Jika $A = \{(x, y) | x + y < 4, x \geq 0, y \geq 0\}$ dan $B = \{(x, y) | 2x + y < 4, x \geq 0 \text{ dan } y \geq 0\}$, maka $B \subseteq A$.

Contoh 2.14

$A = \{p, q, r\}$ bukan himpunan bagian dari $B = \{m, p, q, t, u\}$ karena $r \in A$ tetapi $r \notin B$.

TEOREMA 2.1. Untuk sembarang himpunan A berlaku hal-hal sebagai berikut:

- (a) A adalah himpunan bagian dari A itu sendiri (yaitu, $A \subseteq A$).
- (b) Himpunan kosong merupakan himpunan bagian dari A ($\emptyset \subseteq A$).
- (c) Jika $A \subseteq B$ dan $B \subseteq C$, maka $A \subseteq C$

Bukti untuk teorema 2.1(a) mudah ditunjukkan, karena elemen-elemen dari A adalah anggota dari A . Untuk membuktikan teorema 2.1(b) juga tidak sulit, karena kita harus memperlihatkan bahwa implikasi “Jika $x \in \emptyset$, maka $x \in A$ ” selalu benar (sesuai dengan definisi himpunan bagian). Kita hanya perlu menyatakan bahwa bagian hipotesis (yaitu, $x \in \emptyset$) selalu bernilai salah karena \emptyset tidak mempunyai anggota. Oleh karena itu, implikasi tersebut akan selalu bernilai benar tanpa bergantung kepada himpunan A .

Bukti untuk Teorema 2.1(c) diperlihatkan dari definisi bahwa jika jika setiap elemen dari himpunan A adalah elemen dari himpunan B , dan setiap elemen dari himpunan B adalah elemen dari himpunan C , maka jelas setiap elemen dari A juga adalah elemen dari C .

Dari Teorema 2.1(a) dan teorema 2.1(b) yang menyatakan bahwa $\emptyset \subseteq A$ dan $A \subseteq A$, maka \emptyset dan A disebut himpunan bagian tak sebenarnya (*improper subset*) dari himpunan A . Sebagai contoh, jika $A = \{1, 2, 3\}$, maka $\{1, 2, 3\}$ dan \emptyset adalah *improper subset* dari A , sedangkan himpunan bagian sebenarnya (*proper subset*) dari A adalah $\{1\}$, $\{2\}$, $\{3\}$, $\{1, 2\}$, $\{1, 3\}$, dan $\{2, 3\}$.

Perhatikanlah bahwa penulisan $A \subseteq B$ berbeda dengan $A \subset B$. Jika kita ingin menekankan bahwa A adalah himpunan bagian dari B tetapi $A \neq B$ maka kita menulis $A \subset B$, dan kita katakan bahwa A adalah himpunan bagian sebenarnya (*proper subset*) dari himpunan B [ROS03]. Sebagai contoh, himpunan $\{1\}$ dan $\{2, 3\}$ merupakan *proper subset* dari $\{1, 2, 3\}$. Sebaliknya, pernyataan $A \subseteq B$ digunakan untuk menyatakan bahwa A adalah himpunan bagian (*subset*) dari B yang memungkinkan $A = B$.

Perhatikan juga bahwa himpunan $\{\emptyset\}$ bukan merupakan himpunan bagian dari himpunan $\{\{\emptyset\}\}$, namun ia merupakan anggota himpunan $\{\{\emptyset\}\}$.

Contoh 2.15

Tunjukkan bahwa $A = \{a, b, c\}$ adalah himpunan bagian sebenarnya dari $B = \{a, b, c, d, e, f\}$.

Penyelesaian:

Untuk menunjukkan bahwa A adalah himpunan bagian sebenarnya (*proper subset*) dari B , perlihatkan bahwa setiap elemen di dalam A juga elemen di dalam B dan sekurang-kurangnya ada 1 elemen B yang tidak terdapat di dalam A .

Setiap elemen dari A adalah juga elemen dari B sehingga $A \subseteq B$. Sebaliknya, $d \in B$ tetapi $d \notin A$, oleh karena itu $A \neq B$. Dengan demikian, A adalah himpunan bagian sebenarnya dari B , kita tuliskan $A \subset B$. ■

Contoh 2.16

Misalkan $X = \{4, 5, 6\}$ dan $Z = \{4, 5, 6, 7, 8\}$. Tentukan semua kemungkinan himpunan Y sedemikian hingga $X \subset Y$ dan $Y \subset Z$, yaitu X adalah *proper subset* dari Y dan Y adalah *proper subset* dari Z .

Penyelesaian:

Y harus mengandung semua elemen X dan sekurang-kurangnya satu elemen dari Z . Dengan demikian, $Y = \{4, 5, 6, 7\}$ atau $Y = \{4, 5, 6, 8\}$. Y tidak boleh memuat 7 dan 8 sekaligus karena Y adalah *proper subset* dari Z . ■

2.6 Himpunan yang Sama

Dua buah himpunan mungkin saja sama, yaitu semua anggota di dalam kedua himpunan tersebut sama, meskipun urutannya di dalam himpunan tidak sama. Kita mendefinisikan kesamaan dua buah himpunan melalui definisi 2.4 berikut ini.

DEFINISI 2.4. Himpunan A dikatakan sama dengan himpunan B jika dan hanya jika keduanya mempunyai elemen yang sama. Dengan kata lain, A sama dengan B jika A adalah himpunan bagian dari B dan B adalah himpunan bagian dari A . Jika tidak demikian, maka kita katakan A tidak sama dengan B .

Notasi : $A = B \leftrightarrow A \subseteq B$ dan $B \subseteq A$

Contoh 2.17

- (i) Jika $A = \{ 0, 1 \}$ dan $B = \{ x \mid x(x - 1) = 0 \}$, maka $A = B$
 - (ii) Jika $A = \{ 3, 5, 8, 5 \}$ dan $B = \{ 5, 3, 8 \}$, maka $A = B$
 - (iii) Jika $A = \{ 3, 5, 8, 5 \}$ dan $B = \{ 3, 8 \}$, maka $A \neq B$
-

Tiga hal yang perlu dicatat dalam memeriksa kesamaan dua buah himpunan:

1. Urutan elemen di dalam himpunan tidak penting.
Jadi, $\{1, 2, 3\} = \{3, 2, 1\} = \{1, 3, 2\}$
2. Pengulangan elemen tidak mempengaruhi kesamaan dua buah himpunan.
Jadi, $\{1, 1, 1, 1\} = \{1, 1\} = \{1\}$
 $\{1, 2, 3\} = \{1, 2, 1, 3, 2, 1\}$
3. Untuk tiga buah himpunan, A , B , dan C berlaku aksioma berikut:
 - (a) $A = A$, $B = B$, dan $C = C$
 - (b) jika $A = B$, maka $B = A$
 - (c) jika $A = B$ dan $B = C$, maka $A = C$

2.7 Himpunan yang Ekivalen

Dua buah himpunan dapat mempunyai kardinal yang sama meskipun anggota kedua himpunan tersebut tidak sama. Kita katakan kedua himpunan tersebut ekivalen.

DEFINISI 2.5. Himpunan A dikatakan ekivalen dengan himpunan B jika dan hanya jika kardinal dari kedua himpunan tersebut sama.

Notasi : $A \sim B \leftrightarrow |A| = |B|$

Contoh 2.18

Jika $A = \{ 1, 3, 5, 7 \}$ dan $B = \{ a, b, c, d \}$, maka $A \sim B$ sebab $|A| = |B| = 4$

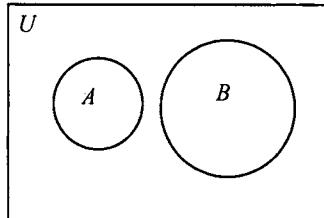
2.8 Himpunan Saling Lepas

Dua buah himpunan mungkin saja tidak memiliki anggota yang sama satu buah pun. Kedua himpunan tersebut dikatakan saling lepas (*disjoint*).

DEFINISI 2.6. Dua himpunan A dan B dikatakan saling lepas jika keduaanya tidak memiliki elemen yang sama.

Notasi : $A // B$

Diagram Venn yang menggambarkan dua himpunan yang saling lepas ditunjukkan pada Gambar 2.3.



Gambar 2.3 Diagram Venn untuk $A // B$

Contoh 2.19

Jika $A = \{ x \mid x \in P, x < 8 \}$ dan $B = \{ 10, 20, 30, \dots \}$, maka $A // B$.

2.9 Himpunan Kuasa

Satu terminologi yang banyak ditemui dalam literatur ilmu komputer adalah himpunan kuasa (*power set*). Himpunan kuasa dari suatu himpunan mengandung semua himpunan bagian dari himpunan yang dimaksud.

DEFINISI 2.7. Himpunan kuasa (*power set*) dari himpunan A adalah suatu himpunan yang elemennya merupakan semua himpunan bagian dari A , termasuk himpunan kosong dan himpunan A sendiri.

Notasi : $P(A)$ atau 2^A

Catatan:

Pada kebanyakan literatur, notasi untuk himpunan kuasa adalah “ \wp ”, tetapi untuk lebih memudahkan penulisan kita menggantinya dengan huruf “ P ” (dari huruf pertama kata “*power*”)

Contoh 2.20

Jika $A = \{ 1, 2 \}$, maka $P(A) = \{ \emptyset, \{ 1 \}, \{ 2 \}, \{ 1, 2 \} \}$

Contoh 2.21

Himpunan kuasa dari himpunan kosong adalah $P(\emptyset) = \{ \emptyset \}$, dan himpunan kuasa dari himpunan $\{ \emptyset \}$ adalah $P(\{ \emptyset \}) = \{ \emptyset, \{ \emptyset \} \}$.

Berapa banyak anggota himpunan kuasa dari sembarang himpunan A ? Jika $|A| = n$, maka $|P(A)| = 2^n$. Kardinalitas himpunan kuasa ini dapat dibuktikan sebagai berikut:

Bukti:

Susun elemen-elemen A sebagai barisan (a_1, a_2, \dots, a_n) .

Bentuk himpunan bagian A_i dari A dan juga bentuk barisan-barisan biner

$$E_i = (e_{1i}, e_{2i}, \dots, e_{ni})$$

dengan

$$e_{ji} = 1 \text{ bila } a_j \text{ ada di dalam } A_i$$

$$e_{ji} = 0 \text{ jika } a_j \text{ tidak ada di dalam } A_i.$$

Banyaknya kombinasi E_i yang mungkin muncul adalah $2^n - 1$. Mengingat himpunan kosong juga merupakan himpunan bagian A , maka terbukti bahwa jumlah himpunan bagian dari himpunan A sama dengan $2^n - 1 + 1 = 2^n$.

2.10 Operasi Terhadap Himpunan

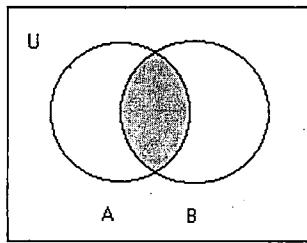
Terhadap dua buah himpunan atau lebih, kita dapat melakukan operasi untuk menghasilkan himpunan lain. Jenis operasi yang lazim digunakan terhadap himpunan adalah operasi irisan (*intersection*), gabungan (*union*), komplementen, selisih (*difference*), perkalian kartesian (*cartesian product*), dan beda-setangkup (*symmetric difference*)

a. IRISAN (*intersection*)

DEFINISI 2.8. Irisan (*intersection*) dari himpunan A dan B adalah sebuah himpunan yang setiap elemennya merupakan elemen dari himpunan A dan himpunan B .

Notasi : $A \cap B = \{ x \mid x \in A \text{ dan } x \in B \}$

Diagram Venn untuk $A \cap B$ ditunjukkan pada Gambar 2.4.



Gambar 2.4 Diagram Venn untuk $A \cap B$ (daerah $A \cap B$ diarsir)

Jika dua himpunan saling lepas, maka irisannya adalah himpunan kosong, karena tidak ada elemen yang sama yang terdapat di dalam kedua himpunan tersebut.

Contoh 2.22

Tinjau (i), (ii), dan (iii) di bawah ini.

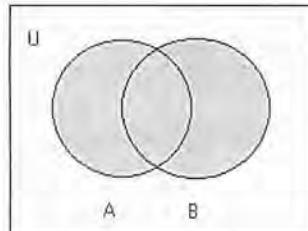
- (i) Jika $A = \{2, 4, 6, 8, 10\}$ dan $B = \{4, 10, 14, 18\}$, maka $A \cap B = \{4, 10\}$
- (ii) Jika $A = \{(x, y) \mid x + y = 7, x, y \in R\}$ dan $B = \{(x, y) \mid x - y = 3, x, y \in R\}$, maka $A \cap B = \{(5, 2)\}$, yang merupakan titik potong garis $x + y = 7$ dan $x - y = 3$.
- (iii) Jika $A = \{3, 5, 9\}$ dan $B = \{-2, 6\}$, maka $A \cap B = \emptyset$. Artinya: $A // B$

b. GABUNGAN (*union*)

DEFINISI 2.9. Gabungan (*union*) dari himpunan A dan B adalah himpunan yang setiap anggotanya merupakan anggota himpunan A atau himpunan B .

Notasi : $A \cup B = \{ x \mid x \in A \text{ atau } x \in B \}$

Diagram Venn untuk $A \cup B$ ditunjukkan pada Gambar 2.5.



Gambar 2.5 Diagram Venn untuk $A \cup B$ (daerah $A \cup B$ diarsir)

Contoh 2.23

Tinjau (i) dan (ii) di bawah ini.

- (i) Jika $A = \{ 2, 5, 8 \}$ dan $B = \{ 7, 5, 22 \}$, maka $A \cup B = \{ 2, 5, 7, 8, 22 \}$
(ii) $A \cup \emptyset = A$

c. KOMPLEMEN (*complement*)

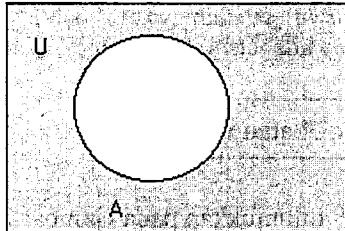
DEFINISI 2.10. Komplemen dari suatu himpunan A terhadap suatu himpunan semesta U adalah suatu himpunan yang elemennya merupakan elemen U yang bukan elemen A .

Notasi : $\overline{A} = \{ x \mid x \in U \text{ dan } x \notin A \}$

Diagram Venn untuk \overline{A} ditunjukkan pada Gambar 2.6.

Catatan:

Beberapa literatur menuliskan lambang komplemen sebagai A^C atau A' .



Gambar 2.6 Diagram Venn untuk \overline{A} (daerah \overline{A} diarsir)

Contoh 2.24

Misalkan $U = \{1, 2, 3, \dots, 9\}$,

- (i) jika $A = \{1, 3, 7, 9\}$, maka $\overline{A} = \{2, 4, 6, 8\}$
- (ii) jika $A = \{x \mid x/2 \in P, x < 9\}$, maka $\overline{A} = \{1, 3, 5, 7, 9\}$

Contoh 2.25

Misalkan:

A = himpunan semua mobil buatan dalam negeri

B = himpunan semua mobil impor

C = himpunan semua mobil yang dibuat sebelum tahun 1990

D = himpunan semua mobil yang nilai jualnya kurang dari Rp 100 juta

E = himpunan semua mobil milik mahasiswa universitas tertentu

- (i) pernyataan “semua mobil mahasiswa di universitas ini produksi dalam negeri atau diimpor dari luar negeri” dapat dinyatakan dalam notasi himpunan sebagai $(E \cap A) \cup (E \cap B)$ atau $E \cap (A \cup B)$
- (ii) pernyataan “semua mobil produksi dalam negeri yang dibuat sebelum tahun 1990 yang nilai jualnya kurang dari Rp 100 juta” dapat dinyatakan dalam notasi himpunan sebagai $A \cap C \cap D$
- (iii) pernyataan “semua mobil impor buatan setelah tahun 1990 yang mempunyai nilai jual lebih dari Rp 100 juta” dapat dinyatakan dalam notasi himpunan sebagai $\overline{C} \cap \overline{D} \cap B$

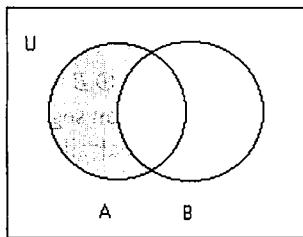
d. SELISIH (*difference*)

DEFINISI 2.11. Selisih dari dua himpunan A dan B adalah suatu himpunan yang elemennya merupakan elemen dari A tetapi bukan elemen dari B . Selisih antara A dan B dapat juga dikatakan sebagai komplemen himpunan B relatif terhadap himpunan A .

Notasi : $A - B = \{ x \mid x \in A \text{ dan } x \notin B \} = A \cap \bar{B}$

Perhatikan bahwa komplemen dari sembarang himpunan A terhadap semesta U dapat juga didefinisikan sebagai $\bar{A} = U - A$.

Diagram Venn untuk $A - B$ ditunjukkan pada Gambar 2.7.



Gambar 2.7 Diagram Venn untuk $A - B$ (daerah $A - B$ diarsir)

Contoh 2.26

Tinjau (i), (ii), dan (iii) di bawah ini.

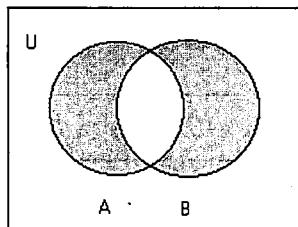
- (i) Jika $A = \{ 1, 2, 3, \dots, 10 \}$ dan $B = \{ 2, 4, 6, 8, 10 \}$, maka $A - B = \{ 1, 3, 5, 7, 9 \}$ dan $B - A = \emptyset$.
 - (ii) $\{1, 3, 5\} - \{1, 2, 3\} = \{5\}$, tetapi $\{1, 2, 3\} - \{1, 3, 5\} = \{2\}$
 - (iii) A = himpunan fungsi menerus (kontinu) dan terbatas di dalam selang $[0,1]$.
 B = himpunan fungsi *differentiable* di dalam selang $[0,1]$.
 $B - A$ = himpunan fungsi *differentiable* tak terbatas di dalam selang $[0, 1]$.
-

e. BEDA SETANGKUP (*Symmetric Difference*)

DEFINISI 2.12. Beda setangkup dari himpunan A dan B adalah suatu himpunan yang elemennya ada pada himpunan A atau B , tetapi tidak pada keduanya.

Notasi: $A \oplus B = (A \cup B) - (A \cap B) = (A - B) \cup (B - A)$

Diagram Venn untuk $A \oplus B$ diarsir ditunjukkan pada Gambar 2.8.



Gambar 2.8 Diagram Venn untuk $A \oplus B$ (daerah $A \oplus B$ diarsir)

Contoh 2.27

- (i) Jika $A = \{ 2, 4, 6 \}$ dan $B = \{ 2, 3, 5 \}$, maka $A \oplus B = \{ 3, 4, 5, 6 \}$
- (ii) A = himpunan segitiga sama kaki, B = himpunan segitiga siku-siku
 $A \oplus B$ = himpunan segitiga sama kaki yang tidak siku-siku dan segitiga siku-siku yang tidak sama kaki

Contoh 2.28

Misalkan

U = himpunan mahasiswa

P = himpunan mahasiswa yang nilai ujian UTS di atas 80

Q = himpunan mahasiswa yang nilain ujian UAS di atas 80

Seorang mahasiswa mendapat nilai A jika nilai UTS dan nilai UAS keduanya di atas 80, mendapat nilai B jika salah satu ujian di atas 80, dan mendapat nilai C jika kedua ujian di bawah 80. Maka, dalam notasi himpunan

- (i) pernyataan "semua mahasiswa yang mendapat nilai A" adalah $P \cap Q$
- (ii) pernyataan "semua mahasiswa yang mendapat nilai B" adalah $P \oplus Q$
- (iii) pernyataan "semua mahasiswa yang mendapat nilai C" adalah $U - (P \cup Q)$

TEOREMA 2.2. Beda setangkup memenuhi hukum-hukum berikut:

- (a) $A \oplus B = B \oplus A$ (hukum komutatif)
- (b) $(A \oplus B) \oplus C = A \oplus (B \oplus C)$ (hukum asosiatif)

f. PERKALIAN KARTESIAN (*cartesian product*)

DEFINISI 2.13. Perkalian kartesian dari himpunan A dan B adalah himpunan yang elemennya semua pasangan berurutan (*ordered pairs*) yang dibentuk dari komponen pertama dari himpunan A dan komponen kedua dari himpunan B .

Notasi: $A \times B = \{(a, b) \mid a \in A \text{ dan } b \in B\}$

Contoh 2.29

Tinjau (i) dan (ii) di bawah ini.

- (i) Misalkan $C = \{1, 2, 3\}$, dan $D = \{a, b\}$, maka perkalian kartesian C dan D adalah $C \times D = \{(1, a), (1, b), (2, a), (2, b), (3, a), (3, b)\}$
- (ii) Misalkan $A = B = \text{himpunan semua bilangan riil}$, maka $A \times B = \text{himpunan semua titik di bidang datar}$
-

Catatlah bahwa:

1. Jika A dan B merupakan himpunan berhingga, maka: $|A \times B| = |A| \cdot |B|$.
 2. Pasangan berurutan (a, b) berbeda dengan (b, a) , dengan kata lain $(a, b) \neq (b, a)$.
 3. Perkalian kartesian tidak komutatif, yaitu $A \times B \neq B \times A$ dengan syarat A atau B tidak kosong. Pada Contoh 2.29(i) di atas, $D \times C = \{(a, 1), (a, 2), (a, 3), (b, 1), (b, 2), (b, 3)\} \neq C \times D$.
 4. Jika $A = \emptyset$ atau $B = \emptyset$, maka $A \times B = B \times A = \emptyset$
-

Contoh 2.30

Misalkan

$A = \text{himpunan makanan} = \{s = \text{soto}, g = \text{gado-gado}, n = \text{nasi goreng}, m = \text{mie rebus}\}$

$B = \text{himpunan minuman} = \{c = \text{coca-cola}, t = \text{teh}, d = \text{es dawet}\}$

Berapa banyak kombinasi makanan dan minuman yang dapat disusun dari kedua himpunan di atas? Jawabnya adalah $|A \times B| = |A| \cdot |B| = 4 \cdot 3 = 12$ kombinasi dan minuman, yaitu $\{(s, c), (s, t), (s, d), (g, c), (g, t), (g, d), (n, c), (n, t), (n, d), (m, c), (m, t), (m, d)\}$.

Contoh 2.31

Daftarkan semua anggota himpunan berikut:

- (a) $P(\emptyset)$ (b) $\emptyset \times P(\emptyset)$ (c) $\{\emptyset\} \times P(\emptyset)$ (d) $P(P(\{3\}))$

Penyelesaian:

- (a) $P(\emptyset) = \{\emptyset\}$
 (b) $\emptyset \times P(\emptyset) = \emptyset$ (ket: jika $A = \emptyset$ atau $B = \emptyset$ maka $A \times B = \emptyset$)
 (c) $\{\emptyset\} \times P(\emptyset) = \{\emptyset\} \times \{\emptyset\} = \{(\emptyset, \emptyset)\}$
 (d) $P(P(\{3\})) = P(\{\emptyset, \{3\}\}) = \{\emptyset, \{\emptyset\}, \{\{3\}\}, \{\emptyset, \{3\}\}\}$
-



2.11 Perampatan Operasi Himpunan

Operasi himpunan dapat dilakukan terhadap 2 atau lebih himpunan. Dalam hal ini kita melakukan perampatan (*generalization*) operasi himpunan dengan menggunakan dasar perampatan yang ada pada operasi aritmatika biasa.

Misalkan $A_1, A_2, A_3, \dots, A_n$ merupakan himpunan, maka:

$$A_1 \cap A_2 \cap \dots \cap A_n = \bigcap_{i=1}^n A_i$$

$$A_1 \cup A_2 \cup \dots \cup A_n = \bigcup_{i=1}^n A_i$$

$$A_1 \times A_2 \times \dots \times A_n = \times_{i=1}^n A_i$$

$$A_1 \oplus A_2 \oplus \dots \oplus A_n = \bigoplus_{i=1}^n A_i$$

Notasi perampatan di atas dapat mempermudah penulisan ekspresi yang panjang, misalnya:

$$A \cap (B_1 \cup B_2 \cup \dots \cup B_n) = (A \cap B_1) \cup (A \cap B_2) \cup \dots \cup (A \cap B_n)$$

menjadi:

$$A \cap (\bigcup_{i=1}^n B_i) = \bigcup_{i=1}^n (A \cap B_i)$$

Contoh 2.32

Misalkan:

$$\begin{aligned} A_1 &= \{0, 2, 3\} \\ A_2 &= \{1, 2, 3, 6\} \\ A_3 &= \{-1, 0, 3, 9\} \end{aligned}$$

maka,

$$\bigcap_{i=1}^3 A_i = \{3\} \quad \text{dan} \quad \bigcup_{i=1}^3 A_i = \{-1, 0, 1, 2, 3, 6, 9\}$$

Contoh 2.33

Misalkan $A = \{1, 2\}$, $B = \{a, b\}$, dan $C = \{\alpha, \beta\}$, maka

$$A \times B \times C = \{(1, a, \alpha), (1, a, \beta), (1, b, \alpha), (1, b, \beta), (2, a, \alpha), (2, a, \beta), (2, b, \alpha), (2, b, \beta)\}$$

2.12 Hukum-hukum Aljabar Himpunan

Terdapat beberapa sifat yang berlaku pada operasi antara dua himpunan atau lebih. Sifat-sifat tersebut dinyatakan dalam kesamaan himpunan (*set identities*). Kesamaan tersebut diberi nama “hukum” yang menyatakan bahwa bila dua himpunan atau lebih dioperasikan, maka hukum-hukum yang mengatur operasi tersebut berlaku. Cukup banyak hukum yang terdapat pada himpunan (yang ditemukan pada sejumlah referensi), namun Tabel 2.1 di bawah ini hanya mendaftarkan 11 buah hukum yang penting saja. Beberapa hukum tersebut mirip dengan hukum aljabar pada sistem bilangan riil seperti $a(b + c) = ab + bc$, yaitu hukum distributif, sehingga kadang-kadang hukum-hukum pada himpunan dinamakan juga **hukum-hukum aljabar himpunan**.

Pembuktian hukum-hukum di atas tidak diberikan di sini, dan diserahkan kepada anda sebagai latihan. Lihat upabab 2.17 mengenai metode-metode pembuktian untuk himpunan. Perhatikanlah bahwa terdapat kemiripan antara hukum-hukum himpunan di atas dengan hukum-hukum logika (lihat upabab 1.4). Notasi \wedge , \vee , F , T , pada hukum-hukum logika masing-masing berkoresponden dengan notasi \cap , \cup , \emptyset , dan U pada himpunan.

Tabel 2.1 Hukum-hukum aljabar Himpunan

1. Hukum identitas: (i) $A \cup \emptyset = A$ (ii) $A \cap U = A$	2. Hukum <i>null/dominasi</i> : (i) $A \cap \emptyset = \emptyset$ (ii) $A \cup U = U$
3. Hukum komplemen: (i) $A \cup \bar{A} = U$ (ii) $A \cap \bar{A} = \emptyset$	4. Hukum idempoten: (i) $A \cup A = A$ (ii) $A \cap A = A$

5. Hukum involusi: $\overline{(\overline{A})} = A$	6. Hukum penyerapan (absorpsi): (i) $A \cup (A \cap B) = A$ (ii) $A \cap (A \cup B) = A$
7. Hukum komutatif: (i) $A \cup B = B \cup A$ (ii) $A \cap B = B \cap A$	8. Hukum asosiatif: (i) $A \cup (B \cup C) = (A \cup B) \cup C$ (ii) $A \cap (B \cap C) = (A \cap B) \cap C$
9. Hukum distributif: (i) $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$ (ii) $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$	10. Hukum De Morgan: (i) $\overline{A \cap B} = \overline{A} \cup \overline{B}$ (ii) $\overline{A \cup B} = \overline{A} \cap \overline{B}$
11. Hukum 0/1 (atau hukum komplemen 2) (i) $\overline{\emptyset} = U$ (ii) $\overline{U} = \emptyset$	

2.13 Prinsip Dualitas

Prinsip dualitas banyak ditemukan pada beberapa situasi. Prinsip ini menyatakan bahwa dua konsep yang berbeda dapat dipertukarkan namun tetap memberikan jawaban yang benar. Misalnya di negara Amerika Serikat kemudi mobil terletak di depan bagian kiri, sedangkan di negara Inggris (juga di Indonesia) kemudi mobil terletak di depan bagian kanan. Kedua letak kemudi ini menimbulkan peraturan yang berbeda pada kedua negara [LIU85]:

- (a) di Amerika Serikat,
 - mobil harus berjalan di bagian *kanan* jalan,
 - pada jalur yang berlajur banyak, lajur *kiri* untuk mendahului,
 - bila lampu merah menyala, mobil belok *kanan* boleh langsung
- (b) di Inggris,
 - mobil harus berjalan di bagian *kiri* jalan,
 - pada jalur yang berlajur banyak, lajur *kanan* untuk mendahului,
 - bila lampu merah menyala, mobil belok *kiri* boleh langsung

Jadi, konsep kiri dan kanan dapat dipertukarkan pada kedua negara tersebut sehingga peraturan yang berlaku di Amerika Serikat menjadi berlaku pula di Inggris. Prinsip inilah yang dinamakan dengan prinsip **dualitas**.

Prinsip dualitas juga ditemukan di dalam teori himpunan. Di dalam Tabel 2.1 kita dapat melihat bahwa beberapa sifat operasi himpunan merupakan analog satu sama lain. Sebagai contoh, pada hukum komplemen, $A \cup \bar{A} = U$ analog dengan $A \cap \bar{A} = \emptyset$, begitu juga pada hukum asosiatif, $A \cap (B \cap C) = (A \cap B) \cap C$ analog dengan $A \cup (B \cup C) = (A \cup B) \cup C$. Hukum yang kedua diperoleh dari hukum yang pertama dengan cara mengganti tanda \cap dengan \cup , \cup dengan \cap , \emptyset dengan U , U dengan \emptyset , dan membiarkan komplemen tetap seperti dinyatakan sebelumnya.

DEFINISI 2.14 (Prinsip Dualitas pada Himpunan). Misalkan S adalah suatu kesamaan yang melibatkan himpunan (*set identity*) dan operasi-operasi seperti \cup , \cap , dan komplemen. Jika S^* diperoleh dari S dengan mengganti \cup menjadi \cap , \cap menjadi \cup , \emptyset menjadi U , dan U menjadi \emptyset , sedangkan komplemen dibiarkan seperti semula, maka kesamaan S^* juga benar dan disebut dual dari kesamaan S .

Contoh 2.34

Dual dari $(A \cap B) \cup (A \cap \bar{B}) = A$ adalah $(A \cup B) \cap (A \cup \bar{B}) = A$. Dualitas ini berarti bahwa $(A \cap B) \cup (A \cap \bar{B}) = A$ adalah kesamaan yang benar, dan dualnya, $(A \cup B) \cap (A \cup \bar{B}) = A$, juga benar. ■

Prinsip dualitas merupakan prinsip yang penting di dalam aljabar himpunan. Kita dapat menggunakan prinsip ini untuk menurunkan kesamaan himpunan (*set identities*) lain yang mengandung operator \cup dan \cap , dan himpunan \emptyset dan U atau membuktikan dual dari kesamaan himpunan lain. Tabel 2.2 memperlihatkan bahwa hukum-hukum aljabar himpunan merupakan contoh dualitas.

Tabel 2.2 Dualitas dari hukum-hukum aljabar Himpunan

1. Hukum identitas: $A \cup \emptyset = A$	Dualnya: $A \cap U = A$
2. Hukum <i>null/dominasi</i> : $A \cap \emptyset = \emptyset$	Dualnya: $A \cup U = U$
3. Hukum komplemen: $A \cup \bar{A} = U$	Dualnya: $A \cap \bar{A} = \emptyset$
4. Hukum idempoten: $A \cup A = A$	Dualnya: $A \cap A = A$

5. Hukum penyerapan: $A \cup (A \cap B) = A$	Dualnya: $A \cap (A \cup B) = A$
6. Hukum komutatif: $A \cup B = B \cup A$	Dualnya: $A \cap B = B \cap A$
7. Hukum asosiatif: $A \cup (B \cup C) = (A \cup B) \cup C$	Dualnya: $A \cap (B \cap C) = (A \cap B) \cap C$
8. Hukum distributif: $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$	Dualnya: $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$
9. Hukum De Morgan: $\overline{A \cup B} = \overline{A} \cap \overline{B}$	Dualnya: $\overline{A \cap B} = \overline{A} \cup \overline{B}$
10. Hukum 0/1 $\overline{\emptyset} = U$	Dualnya: $\overline{U} = \emptyset$

2.14 Prinsip Inklusi-Eksklusi

Berapa banyak anggota di dalam gabungan dua buah himpunan A dan B ? Penggabungan dua buah himpunan menghasilkan himpunan baru yang elemen-elemennya berasal dari himpunan A dan himpunan B . Himpunan A dan himpunan B mungkin saja memiliki elemen-elemen yang sama. Banyaknya elemen bersama antara A dan B adalah $|A \cap B|$. Setiap unsur yang sama itu telah dihitung dua kali, sekali pada $|A|$ dan sekali pada $|B|$, meskipun ia seharusnya dianggap sebagai satu buah elemen di dalam $|A \cup B|$. Karena itu, jumlah elemen hasil penggabungan seharusnya adalah jumlah elemen di masing-masing himpunan dikurangi dengan jumlah elemen di dalam irisan mereka, atau

$$|A \cup B| = |A| + |B| - |A \cap B| \quad (2.1)$$

Prinsip ini dikenal dengan nama prinsip **inklusi-eksklusi**. Sejumlah *lemma* dan teorema yang berkaitan dengan prinsip ini dituliskan sebagai berikut:

LEMMA 2.1. Misalkan A dan B adalah himpunan berhingga yang saling lepas (*disjoint*), maka $|A \cup B| = |A| + |B|$.

TEOREMA 2.3. Misalkan A dan B adalah himpunan berhingga, maka $|A \cup B|$ berhingga dan $|A \cup B| = |A| + |B| - |A \cap B|$.

Dengan cara yang sama, kita dapat menghitung jumlah elemen hasil operasi beda setangkup:

$$|A \oplus B| = |A| + |B| - 2|A \cap B| \quad (2.2)$$

Contoh 2.35

Berapa banyaknya bilangan bulat antara 1 dan 100 yang habis dibagi 3 atau 5?

Penyelesaian:

Misalkan,

A = himpunan bilangan bulat yang habis dibagi 3,

B = himpunan bilangan bulat yang habis dibagi 5,

$A \cap B$ = himpunan bilangan bulat yang habis dibagi 3 dan 5 (yaitu himpunan bilangan bulat yang habis dibagi oleh KPK – Kelipatan Persekutuan Terkecil – dari 3 dan 5, yaitu 15),

yang ditanyakan adalah $|A \cup B|$.

Terlebih dahulu kita harus menghitung

$$|A| = \lfloor 100/3 \rfloor = 33, \quad |B| = \lfloor 100/5 \rfloor = 20, \quad |A \cap B| = \lfloor 100/15 \rfloor = 6$$

untuk mendapatkan

$$|A \cup B| = |A| + |B| - |A \cap B| = 33 + 20 - 6 = 47$$

Jadi, ada 47 buah bilangan yang habis dibagi 3 atau 5. ■

Prinsip inklusi-eksklusi dapat dirampatkan untuk operasi lebih dari dua buah himpunan. Untuk tiga buah himpunan A , B , dan C , berlaku teorema berikut:

TEOREMA 2.4. Misalkan A , B , dan C adalah himpunan berhingga, maka $|A \cup B \cup C|$ berhingga dan

$$|A \cup B \cup C| = |A| + |B| + |C| - |A \cap B| - |A \cap C| - |B \cap C| + |A \cap B \cap C|$$

dan untuk r buah himpunan berlaku teorema berikut:

TEOREMA 2.4. Misalkan A_1, A_2, \dots, A_r adalah himpunan berhingga, maka berlaku

$$\begin{aligned} |A_1 \cup A_2 \cup \dots \cup A_r| &= \sum_i |A_i| - \sum_{1 \leq i \leq j \leq r} |A_i \cap A_j| + \sum_{1 \leq i \leq j \leq k \leq r} |A_i \cap A_j \cap A_k| \\ &\quad + \dots + (-1)^{r-1} |A_1 \cap A_2 \cap \dots \cap A_r| \end{aligned}$$

Contoh 2.36

Sebanyak 1232 orang mahasiswa mengambil kuliah Bahasa Inggris, 879 orang mengambil kuliah Bahasa Perancis, dan 114 mengambil kuliah Bahasa Jerman. Sebanyak 103 orang mengambil kuliah Bahasa Inggris dan Perancis, 23 orang mengambil kuliah Bahasa Inggris dan Jerman, dan 14 orang mengambil kuliah Bahasa Perancis dan Bahasa Jerman. Jika 2092 orang mengambil paling sedikit satu buah kuliah Bahasa Inggris, Bahasa Perancis, dan Bahasa Jerman, berapa banyak mahasiswa yang mengambil kuliah ketiga buah bahasa tersebut?

Penyelesaian:

Misalkan,

I = himpunan mahasiswa yang mengambil kuliah Bahasa Inggris,

P = himpunan mahasiswa yang mengambil kuliah Bahasa Perancis,

J = himpunan mahasiswa yang mengambil kuliah Bahasa Jerman.

maka,

$$|I| = 1232, \quad |P| = 879, \quad |J| = 114$$

$$|I \cap P| = 103, \quad |I \cap J| = 23, \quad |P \cap J| = 14$$

dan

$$|I \cup P \cup J| = 2092$$

Penyulihan nilai-nilai di atas pada persamaan

$$|I \cup P \cup J| = |I| + |P| + |J| - |I \cap P| - |I \cap J| - |P \cap J| + |I \cap P \cap J|$$

memberikan

$$2092 = 1232 + 879 + 114 - 103 - 23 - 14 + |I \cap P \cap J|$$

sehingga

$$|I \cap P \cap J| = 7$$

Jadi, ada 7 orang mahasiswa yang mengambil ketiga buah kuliah Bahasa Inggris, Perancis, dan Jerman. ■

2.15 Partisi

Tinjau sekumpulan mahasiswa di sebuah kelas. Bagaimana cara dosen membagi mahasiswa menjadi sejumlah kelompok? Dosen dapat membagi himpunan mahasiswa menjadi beberapa buah himpunan bagian, yang dalam hal ini setiap himpunan bagian mungkin berisi 1 orang mahasiswa, 2 orang mahasiswa, dan

seterusnya, bahkan kosong. Tidak ada mahasiswa yang sama berada di dalam dua atau lebih himpunan bagian yang berbeda. Gabungan dari seluruh himpunan bagian itu adalah seluruh mahasiswa di dalam kelas.

DEFINISI 2.15. Partisi dari sebuah himpunan A adalah sekumpulan himpunan bagian tidak kosong A_1, A_2, \dots dari A sedemikian sehingga:

- (a) $A_1 \cup A_2 \cup \dots = A$, dan
- (b) himpunan bagian A_i saling lepas, yaitu $A_i \cap A_j = \emptyset$ untuk $i \neq j$.

Contoh 2.37

Misalkan $A = \{1, 2, 3, 4, 5, 6, 7, 8\}$, maka $\{\{1\}, \{2, 3, 4\}, \{7, 8\}, \{5, 6\}\}$ adalah partisi dari A . ■

Catatlah bahwa partisi membagi himpunan A menjadi beberapa buah “blok”. Pada contoh di atas, himpunan A dibagi menjadi 4 buah blok, yaitu $\{1\}$, $\{2, 3, 4\}$, $\{7, 8\}$, dan $\{5, 6\}$. Jika himpunan A terbatas jumlah elemennya, maka jumlah partisi yang dapat dibentuk tidak lebih banyak dari $|A|$.

2.16 Pembuktian Proposisi Himpunan

Proposisi himpunan adalah pernyataan yang menggunakan notasi himpunan. Pernyataan dapat berupa kesamaan (*set identity*), misalnya $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$ adalah sebuah kesamaan himpunan, atau dapat berupa implikasi seperti “Jika $A \cap B = \emptyset$ dan $A \subseteq (B \cup C)$ maka selalu berlaku bahwa $A \subseteq C$ ”.

Terdapat beberapa metode untuk membuktikan kebenaran proposisi himpunan. Untuk suatu proposisi himpunan, kita dapat membuktikannya dengan beberapa metode yang menghasilkan kesimpulan yang sama. Di bawah ini dikemukakan beberapa metode pembuktian proposisi perihal himpunan.

1. Pembuktian dengan menggunakan diagram Venn

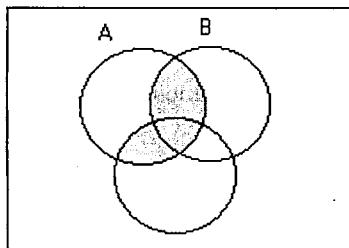
Buatlah diagram Venn untuk bagian ruas kiri keramaan dan diagram Venn untuk ruas kanan kesamaan. Jika diagram Venn keduanya sama, berarti kesaman tersebut benar.

Contoh 2.38

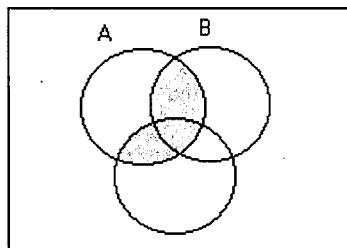
Misalkan A , B , dan C adalah himpunan. Buktikan $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$ dengan diagram Venn.

Penyelesaian:

Diagram Venn untuk ruas kiri dan ruas kanan masing-masing ditunjukkan pada Gambar 2.9. Dari diagram Venn untuk masing-masing ruas di atas, keduanya memberikan area arsiran yang sama. Jadi, terbukti bahwa $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$. ■



$$A \cap (B \cup C)$$



$$(A \cap B) \cup (A \cap C)$$

Gambar 2.9 Diagram Venn untuk pembuktian $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$

Dengan diagram Venn, pembuktian dapat dilakukan dengan cepat. Ini adalah kelebihan metode ini. Namun, kekurangannya, diagram Venn hanya dapat digunakan jika himpunan yang digambarkan tidak banyak jumlahnya. Selain itu, metode ini *mengilustrasikan* ketimbang membuktikan fakta. Banyak matematikawan tidak menganggapnya sebagai metode yang valid untuk pembuktian secara formal. Oleh karena itu, pembuktian dengan diagram Venn kurang dapat diterima.

2. Pembuktian dengan menggunakan tabel keanggotaan

Kesamaan himpunan juga dapat dibuktikan dengan menggunakan **tabel keanggotaan** (*membership tables*). Kita menggunakan angka 1 untuk menyatakan bahwa suatu elemen adalah anggota himpunan, dan 0 untuk menyatakan bukan himpunan (nilai ini dapat dianalogikan *true* dan *false*).

Contoh 2.39

Misalkan A , B , dan C adalah himpunan. Buktikan bahwa $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$. Tabel keanggotaan untuk kesamaan tersebut adalah seperti di bawah ini. Karena kolom $A \cap (B \cup C)$ dan kolom $(A \cap B) \cup (A \cap C)$ sama, maka kesamaan tersebut benar. ■

A	B	C	$B \cup C$	$A \cap (B \cup C)$	$A \cap B$	$A \cap C$	$(A \cap B) \cup (A \cap C)$
0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	1	0	0	0	0
0	1	1	1	0	0	0	0
1	0	0	0	0	0	0	0
1	0	1	1	1	0	1	1
1	1	0	1	1	1	0	1
1	1	1	1	1	1	1	1

Penjelasan: pandang baris kedua dari tabel di atas:

A	B	C	$B \cup C$	$A \cap (B \cup C)$	$A \cap B$	$A \cap C$	$(A \cap B) \cup (A \cap C)$
0	0	1	1	0	0	0	0

Arti dari baris tersebut adalah: misalkan $x \notin A$ (nilai 0), $x \notin B$ (nilai 0), $x \notin C$ (nilai 0), maka x pasti $\in B \cup C$ (nilai 1), tetapi $x \notin A \cap (B \cup C)$ (nilai 0), $x \notin A \cap B$ (nilai 0), $x \notin A \cap C$ (nilai 0), dan $x \notin ((A \cap B) \cup (A \cap C))$ (nilai 0).

3. Pembuktian dengan menggunakan aljabar himpunan.

Aljabar himpunan mengacu pada hukum-hukum yang dikemukakan pada upabab 2.12, termasuk di dalamnya teorema-teorema (yang ada buktinya), definisi suatu operasi himpunan dan penerapan prinsip dualitas.

Contoh 2.40

Misalkan A dan B himpunan. Buktikan bahwa $(A \cap B) \cup (A \cap \bar{B}) = A$

Penyelesaian:

$$\begin{aligned}
 (A \cap B) \cup (A \cap \bar{B}) &= A \cap (B \cup \bar{B}) && (\text{Hukum distributif}) \\
 &= A \cap U && (\text{Hukum komplemen}) \\
 &= A && (\text{Hukum identitas})
 \end{aligned}$$

■

Contoh 2.41

Misalkan A dan B himpunan. Buktikan bahwa $A \cup (B - A) = A \cup B$

Penyelesaian:

$$\begin{aligned}
 A \cup (B - A) &= A \cup (B \cap \bar{A}) && (\text{Definisi operasi selisih}) \\
 &= (A \cup B) \cap (A \cup \bar{A}) && (\text{Hukum distributif}) \\
 &= (A \cup B) \cap U && (\text{Hukum komplemen}) \\
 &= A \cup B && (\text{Hukum identitas})
 \end{aligned}$$

■

Contoh 2.42

Misalkan A dan B himpunan. Tunjukkan bahwa $(A - B) - C = (A - C) - B$

Penyelesaian:

$$\begin{aligned}
 (A - B) - C &= (A \cap \bar{B}) - C && (\text{Definisi operasi selisih}) \\
 &= (A \cap \bar{B}) \cap \bar{C} && (\text{Definisi operasi selisih}) \\
 &= (A \cap \bar{C}) \cap \bar{B} && (\text{Hukum asosiatif}) \\
 &= (A - C) \cap \bar{B} && (\text{Definisi operasi selisih}) \\
 &= (A - C) - B && (\text{Definisi operasi selisih})
 \end{aligned}$$

■

Contoh 2.43

Tunjukkan bahwa $A \cup (\overline{A \cup B}) = A \cup \overline{B}$

Penyelesaian:

$$\begin{aligned} A \cup (\overline{A \cup B}) &= A \cup (\overline{A} \cap \overline{B}) && \text{(Hukum De Morgan)} \\ &= (A \cup \overline{A}) \cap (\overline{A} \cup \overline{B}) && \text{(Hukum distributif)} \\ &= U \cap (\overline{A} \cup \overline{B}) && \text{(Hukum komplemen)} \\ &= A \cup \overline{B} && \text{(Hukum identitas)} \end{aligned}$$

Contoh 2.44

Buktikan untuk sembarang himpunan A dan B , bahwa

- (i) $A \cup (\overline{A} \cap B) = A \cup B$ dan
(ii) $A \cap (\overline{A} \cup B) = A \cap B$

Penyelesaian:

$$\begin{aligned} \text{(i) } A \cup (\overline{A} \cap B) &= (A \cup (A \cap B)) \cup (\overline{A} \cap B) && \text{(Hukum penyerapan)} \\ &= A \cup \{(A \cap B) \cup (\overline{A} \cap B)\} && \text{(Hukum asosiatif)} \\ &= A \cup \{(A \cup \overline{A})\} \cap B && \text{(Hukum distributif)} \\ &= A \cup (U \cap B) && \text{(Hukum komplemen)} \\ &= A \cup B && \text{(Hukum identitas)} \\ \\ \text{(ii) } A \cap (\overline{A} \cup B) &= (A \cap \overline{A}) \cup (A \cap B) && \text{(Hukum distributif)} \\ &= \emptyset \cup (A \cap B) && \text{(Hukum komplemen)} \\ &= A \cap B && \text{(Hukum identitas)} \end{aligned}$$

atau, dapat juga melalui dualitas dari (i) sebagai berikut:

$$\begin{aligned} A \cap (\overline{A} \cup B) &= \{A \cap (A \cup B)\} \cap (\overline{A} \cup B) && \text{(Hukum penyerapan)} \\ &= A \cap \{(A \cup B) \cap (\overline{A} \cup B)\} && \text{(Hukum asosiatif)} \\ &= A \cap \{(A \cap \overline{A})\} \cup B && \text{(Hukum distributif)} \\ &= A \cap (\emptyset \cup B) && \text{(Hukum komplemen)} \\ &= A \cap B && \text{(Hukum identitas)} \end{aligned}$$

Contoh 2.45

Jika A dan B masing-masing adalah himpunan, buktikan bahwa

$$(A \oplus B) \cap A = A \cap \overline{B}$$

Penyelesaian:

$$\begin{aligned}
 (A \oplus B) \cap A &= [(A \cap \bar{B}) \cup (B \cap \bar{A})] \cap A && (\text{Definisi operasi beda-setangkup}) \\
 &= [(A \cap \bar{B}) \cap A] \cup [(B \cap \bar{A}) \cap A] && (\text{Hukum distributif}) \\
 &= [(A \cap A) \cap \bar{B}] \cup [(A \cap \bar{A}) \cap B] && (\text{Hukum asosiatif}) \\
 &= (A \cap \bar{B}) \cup [(A \cap \bar{A}) \cap B] && (\text{Hukum idempoten}) \\
 &= (A \cap \bar{B}) \cup (\emptyset \cap B) && (\text{Hukum komplemen}) \\
 &= (A \cap \bar{B}) \cup \emptyset && (\text{Hukum Null}) \\
 &= A \cap \bar{B} && (\text{Hukum identitas})
 \end{aligned}$$

Contoh 2.46

Buktikan hukum penyerapan: (a) $A \cup (A \cap B) = A$ dan (b) $A \cap (A \cup B) = A$

Penyelesaian:

$$\begin{aligned}
 (a) A \cup (A \cap B) &= (A \cap U) \cup (A \cap B) && (\text{Hukum identitas}) \\
 &= A \cap (U \cup B) && (\text{Hukum distributif}) \\
 &= A \cap U && (\text{Hukum identitas}) \\
 &= A && (\text{Hukum identitas})
 \end{aligned}$$

(b) Bukti mengikuti prinsip dualitas dari jawaban (a)

$$\begin{aligned}
 A \cap (A \cup B) &= (A \cup \emptyset) \cap (A \cup B) && (\text{Hukum identitas}) \\
 &= A \cup (\emptyset \cap B) && (\text{Hukum distributif}) \\
 &= A \cup \emptyset && (\text{Hukum identitas}) \\
 &= A && (\text{Hukum identitas})
 \end{aligned}$$

4. Pembuktian dengan menggunakan definisi.

Metode ini digunakan untuk membuktikan proposisi himpunan yang tidak berbentuk kesamaan, tetapi proposisi yang berbentuk implikasi. Biasanya di dalam implikasi tersebut terdapat notasi himpunan bagian (\subseteq atau \subset).

Contoh 2.47

Misalkan A dan B himpunan. Jika $A \cap B = \emptyset$ dan $A \subseteq (B \cup C)$ maka $A \subseteq C$. Buktikan!

Penyelesaian:

- (i) Dari definisi himpunan bagian, $P \subseteq Q$ jika setiap $x \in P$ juga $\in Q$. Misalkan $x \in A$. Karena $A \subseteq (B \cup C)$, maka dari definisi himpunan bagian, x juga $\in (B \cup C)$. Dari definisi operasi gabungan (\cup), $x \in (B \cup C)$ berarti $x \in B$ atau $x \in C$.

(ii) Karena $x \in A$ dan $A \cap B = \emptyset$, maka $x \notin B$

Dari (i) dan (ii), $x \in C$ harus benar. Karena $\forall x \in A$ juga berlaku $x \in C$, maka dapat disimpulkan $A \subseteq C$. ■

Contoh 2.48

Misalkan A , B , dan C adalah tiga buah himpunan. Buktikan jika $A \subseteq B$ maka $A \cap C \subseteq B \cap C$.

Penyelesaian:

- (i) Karena $A \cap C \subseteq B \cap C$, maka dari definisi himpunan bagian, $\forall x \in (A \cap C)$ juga adalah $\in (B \cap C)$
- (ii) Karena $A \cap C$, maka dari definisi operasi irisan, jika $x \in (A \cap C)$, maka $x \in A$ dan $x \in C$.
- (iii) Karena $A \subseteq B$, maka dari definisi himpunan bagian, $\forall y \in A$ juga $\in B$. Jika simbol y diganti dengan x , maka $\forall x \in A$ juga $\in B$.

Dari (ii) telah diketahui $x \in A$ dan $x \in C$. Dari (iii) telah diketahui $x \in B$ jika $x \in A$. Maka, $x \in A$ dan $x \in C$, yang berarti $x \in (B \cap C)$.

Dari (ii) dan (iii) dapat disimpulkan bahwa $A \cap C \subseteq B \cap C$. ■

2.17 Himpunan Ganda

Dari definisi himpunan, himpunan adalah kumpulan elemen yang berbeda. Namun pada beberapa situasi, adakalanya elemen himpunan tidak seluruhnya berbeda, misalnya himpunan nama-nama mahasiswa di sebuah kelas. Nama-nama mahasiswa di dalam sebuah kelas mungkin ada yang sama, karena itu ada perulangan elemen yang sama di dalam himpunan tersebut. Himpunan yang elemennya boleh berulang (tidak harus berbeda) disebut **himpunan ganda (multiset)**. Contohnya, $\{a, a, a, b, b, c\}$, $\{2, 2, 2\}$, $\{2, 3, 4\}$, $\{\}$ adalah himpunan ganda. **Multiplisitas** dari suatu elemen pada himpunan ganda adalah jumlah kemunculan elemen tersebut di dalam himpunan ganda. Sebagai contoh: Jika $M = \{0, 1, 01, 1, 0, 001, 0001, 00001, 0, 0, 1\}$, maka multiplisitas elemen 0 adalah 4. Cara lain menyatakan himpunan-ganda adalah dengan menggunakan multiplisitasnya, misalnya $\{3 \cdot a, 2 \cdot b, 1 \cdot c\}$, yang sama dengan $\{a, a, a, b, b, c\}$.

Himpunan (*set*) merupakan contoh khusus dari suatu himpunan ganda, yang dalam hal ini multiplisitas dari setiap elemennya adalah 1 atau 0.

Kardinalitas dari suatu himpunan ganda didefinisikan sebagai kardinalitas himpunan padanannya (ekivalen), dengan mengasumsikan elemen-elemen di dalam himpunan ganda semuanya berbeda.

Operasi pada himpunan ganda sedikit berbeda dengan operasi pada himpunan biasa. Untuk himpunan ganda, definisi operasi himpunan adalah sebagai berikut:

DEFINISI 2.16. Misalkan P dan Q adalah himpunan ganda:

1. $P \cup Q$ adalah suatu himpunan ganda yang multiplisitas elemennya sama dengan multiplisitas maksimum elemen tersebut pada himpunan P dan Q .

Contoh: Jika $P = \{ a, a, a, c, d, d \}$ dan $Q = \{ a, a, b, c, c \}$, maka $P \cup Q = \{ a, a, a, b, c, c, d, d \}$

2. $P \cap Q$ adalah suatu himpunan ganda yang multiplisitas elemennya sama dengan multiplisitas minimum elemen tersebut pada himpunan P dan Q .

Contoh: Jika $P = \{ a, a, a, c, d, d \}$ dan $Q = \{ a, a, b, c, c \}$ maka $P \cap Q = \{ a, a, c \}$

3. $P - Q$ adalah suatu himpunan ganda yang multiplisitas elemennya sama dengan:

- multiplisitas elemen tersebut pada P dikurangi multiplisitasnya pada Q , jika selisihnya positif
- 0, jika selisihnya nol atau negatif.

Contoh: Jika $P = \{ a, a, a, b, b, c, d, d, e \}$ dan $Q = \{ a, a, b, b, b, c, c, d, d, f \}$ maka $P - Q = \{ a, e \}$

4. $P + Q$, yang didefinisikan sebagai jumlah (*sum*) dua buah himpunan ganda, adalah suatu himpunan ganda yang multiplisitas elemennya sama dengan penjumlahan dari multiplisitas elemen tersebut pada P dan Q . Catatan: Beda setangkup tidak didefinisikan pada himpunan ganda.

Contoh: Jika $P = \{ a, a, b, c, c \}$ dan $Q = \{ a, b, b, d \}$ maka $P + Q = \{ a, a, a, b, b, b, c, c, d \}$

2.18 Tipe Set dalam Bahasa Pascal

Bahasa Pascal menyediakan tipe data khusus untuk himpunan, yang bernama set. Tipe set menyatakan himpunan kuasa dari tipe ordinal.

Contohnya:

```
type
  HurufBesar = 'A'..'Z';
  Huruf = set of HurufBesar;
var
  HurufKu : Huruf;
```

Nilai untuk peubah HurufKu dapat diisi dengan pernyataan berikut:

```
HurufKu:=['A', 'C', 'D'];
HurufKu:=['M'];
HurufKu:[]; { himpunan kosong }
```

Operasi yang dapat dilakukan pada tipe himpunan adalah operasi gabungan, irisan, dan selisih seperti pada contoh berikut:

```
HurufKu:=['A', 'C', 'D'] + ['C', 'D', 'E']; {gabungan}
HurufKu:=['A', 'C', 'D'] * ['C', 'D', 'E']; {irisan}
HurufKu:=['A', 'C', 'D'] - ['C', 'D', 'E']; {selisih}
```

Uji keanggotaan sebuah elemen di dalam himpunan dilakukan dengan menggunakan operator *in* seperti contoh berikut:

```
if 'A' in HurufKu then
...
...
```

Di dalam kelas pemrograman *Delphi*, *set* sering digunakan untuk mengindikasikan *flag*. Misalnya himpunan *icon* untuk *window*:

```
type
TBorderIcon=(biSystemMenu,biMinimize, biMaximize);
Hurst = set of TBorderIcon;
```

2.19 Pengantar Logika dan Himpunan *Fuzzy*

(upabab ini hanya sekadar pengetahuan tambahan)

Logika *fuzzy* pertama kali dikembangkan oleh Lotfi A. Zadeh, seorang ilmuwan Amerika Serikat berkebangsaan Iran dari Universitas California di Berkeley, melalui tulisannya pada tahun 1965. Meskipun logika *fuzzy* dikembangkan di Amerika, namun ia lebih populer dan banyak diaplikasikan secara luas oleh praktisi Jepang dengan mengadaptasikannya ke bidang kendali (*control*). Makanya, tidak heran, kalau saat ini banyak dijual produk elektronik buatan Jepang yang menerapkan prinsip logika *fuzzy*, seperti mesin cuci, AC, dan lain-lain.

Mengapa logika *fuzzy* yang ditemukan di Amerika malah lebih banyak ditemukan aplikasinya di negara Jepang? Salah satu penjelasannya kultur orang Barat yang cenderung memandang suatu persoalan sebagai hitam-putih, ya-tidak, bersalah-tidak bersalah, sukses-gagal, atau yang setara dengan dunia logika biner Aristoteles, sedangkan kultur orang Timur lebih dapat menerima dunia "abu-abu" atau *fuzzy*.

Logika *fuzzy* umumnya diterapkan pada masalah-masalah yang mengandung unsur ketidakpastian (*uncertainty*).

Contoh a: Seseorang dikatakan “tinggi” jika tinggi badannya di atas 1,7 meter. Apakah orang yang mempunyai tinggi badan 1,6999 meter atau 1,65 meter termasuk kategori orang tinggi? Menurut persepsi manusia, orang yang mempunyai tinggi badan sekitar 1,7 meter dikatakan “kurang lebih tinggi” atau “agak tinggi”.

Contoh b: Kecepatan “pelan” didefinisikan di bawah 20 km/jam. Bagaimana dengan kecepatan 20,001 km/jam, apakah masih dapat dikatakan pelan? Kita, manusia, mungkin mengatakan bahwa kecepatan 20,001 km/jam itu “agak pelan”.

Kedua contoh di atas memperlihatkan bahwa ketidakpastian dalam kasus ini disebabkan oleh kaburnya pengertian “agak”, “kurang lebih”, “sedikit”, dan sebagainya.

Logika *fuzzy* dikembangkan dari teori himpunan *fuzzy*. Sementara, himpunan yang telah kita bahas sebelum ini adalah himpunan klasik yang seringkali disebut **himpunan tegas** (*crisp set*). Keanggotaan suatu unsur di dalam himpunan dinyatakan secara tegas, apakah objek tersebut anggota himpunan atau bukan. Untuk sembarang himpunan A , sebuah unsur x adalah anggota himpunan apabila x terdapat atau terdefinisi di dalam A . Sebagai contoh, misalkan $A = \{0, 4, 7, 8, 11\}$, maka jelaslah $7 \in A$, tetapi $5 \notin A$.

Kita sudah mempelajari bahwa himpunan dapat disajikan dalam berbagai cara. Selain keempat cara penyajian yang sudah disebutkan di dalam upabab 2.2, ada cara lain untuk menyajikan himpunan, yaitu dengan menggunakan **fungsi karakteristik**. Fungsi karakteristik, dilambangkan dengan χ , mendefinisikan apakah suatu unsur dari semesta pembicaraan merupakan anggota suatu himpunan atau bukan, yaitu

$$\chi_A(x) = \begin{cases} 1 & , x \in A \\ 0 & , x \notin A \end{cases}$$

Jadi, χ_A memetakan X ke himpunan $\{0, 1\}$, yang dalam hal ini X adalah semesta pembicaraan.

Contoh 2.49.

Misalkan $X = \{1, 2, 3, 4, 5, 6\}$ dan $A \subseteq X$, yang dalam hal ini $A = \{1, 2, 5\}$. Dengan fungsi karakteristik, kita menyatakan A sebagai

$$A = \{(1,1), (2,1), (3,0), (4,0), (5,1), (6,0)\}$$

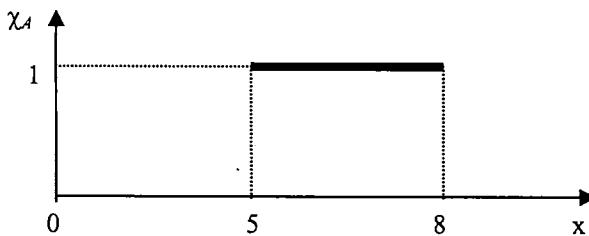
Keterangan: $(2,1)$ berarti $\chi_A(2) = 1$; $(4,0)$ berarti $\chi_A(4) = 0$,

Contoh 2.50.

Misalkan $X = \{x \mid 0 \leq x \leq 10, x \in \mathbf{R}\}$. Misalkan $A \subseteq X$, yang dalam hal ini $A = \{x \mid 5 \leq x \leq 8, x \in \mathbf{R}\}$. Maka, kita dapat menyatakan bahwa

$$\begin{aligned}\chi_A(6) &= 0 \\ \chi_A(4,8) &= 0 \\ \chi_A(7) &= 1 \\ \chi_A(8,654) &= 1\end{aligned}$$

Grafik pada Gambar 2.10 memperlihatkan hubungan antara nilai-nilai x di dalam X dan fungsi karakteristiknya. Garis tebal antara 5 dengan 8 menyatakan bahwa nilai-nilai x di dalam selang tersebut secara tegas mempunyai keanggotaan 1, sedangkan di luar selang tersebut mempunyai nilai keanggotaan 0.



Gambar 2.10 Grafik fungsi karakteristik $A = \{x \mid 5 \leq x \leq 8, x \in \mathbf{R}\}$

Sekarang, tinjau V = himpunan kecepatan pelan (yaitu $v \leq 20$ km/jam). Apakah kecepatan $v = 20,01$ km/jam termasuk ke dalam himpunan kecepatan pelan? Menurut himpunan tegas, $20,01$ km/jam $\notin V$, tetapi menurut himpunan *fuzzy*, $20,01$ km/jam tidak ditolak ke dalam himpunan V , tetapi diturunkan derajat keanggotaannya.

Di dalam teori himpunan *fuzzy*, keanggotaan suatu elemen di dalam himpunan dinyatakan dengan **derajat keanggotaan** (*membership values*) yang nilainya terletak di dalam selang $[0, 1]$.

Derajat keanggotaan ditentukan dengan **fungsi keanggotaan**:

$$\mu_A : X \rightarrow [0, 1]$$

bandingkan fungsi keanggotaan pada teori himpunan tegas:

$$\chi_A : X \rightarrow \{0, 1\}$$

Arti derajat keanggotaan adalah sebagai berikut:

- (a) jika $\mu_A(x) = 1$, maka x adalah anggota penuh dari himpunan A
- (b) jika $\mu_A(x) = 0$, maka x bukan anggota himpunan A
- (c) jika $\mu_A(x) = \mu$, dengan $0 < \mu < 1$, maka x adalah anggota himpunan A dengan derajat keanggotaan sebesar μ .

Cara Mendefinisikan Himpunan *Fuzzy*

Misalkan himpunan *fuzzy* A didefinisikan pada semesta pembicaraan $X = \{x_1, x_2, \dots, x_n\}$. Di bawah ini dijelaskan tiga cara mendefinisikan himpunan *fuzzy*:

Cara 1: Sebagai himpunan pasangan berurutan

$$A = \{ (x_1, \mu_A(x_1)), (x_2, \mu_A(x_2)), \dots, (x_n, \mu_A(x_n)) \}$$

Contoh 2.51.

Misalkan

$X = \{ \text{becak}, \text{sepeda motor}, \text{mobil kodok(VW)}, \text{mobil kijang}, \text{mobil carry} \}$

A = himpunan kendaraan yang nyaman dipakai untuk bepergian jarak jauh oleh keluarga besar (terdiri dari ayah, ibu, dan empat orang anak)

Didefinisikan bahwa,

$$x_1 = \text{becak}, \mu_A(x_1) = 0$$

$$x_2 = \text{sepeda motor}, \mu_A(x_2) = 0.1$$

$$x_3 = \text{mobil kodokbecak}, \mu_A(x_3) = 0.5$$

$$x_4 = \text{mobil kijang}, \mu_A(x_4) = 1.0$$

$$x_5 = \text{mobil carry}, \mu_A(x_5) = 0.8$$

maka, dalam himpunan *fuzzy*,

$$A = \{ (\text{becak}, 0), (\text{sepeda motor}, 0.1), (\text{mobil kodok}, 0.5), \\ (\text{mobil kijang}, 0.5), (\text{mobil carry}, 0.8) \}$$

Cara pertama ini hanya dapat digunakan apabila anggota himpunan *fuzzy* bernilai diskrit. Untuk anggota himpunan *fuzzy* yang bernilai riil, kita tidak dapat menggunakannya.

Cara 2: Dinyatakan dengan menyebut fungsi keanggotaan.

Cara ini digunakan bila anggota himpunan *fuzzy* bernilai menerus (riil).

Contoh 2.52.

Misalkan

$A = \text{himpunan bilangan riil yang dekat } 2$

maka, dalam himpunan *fuzzy*,

$$A = \{(x, \mu_A(x)) \mid \mu_A(x) = 1/(1 + (x - 2)^2)\}$$

Cara 3: Dengan menuliskan sebagai

$$A = \{\mu_A(x_1)/x_1 + \mu_A(x_2)/x_2 + \dots + \mu_A(x_n)/x_n\} = \left\{ \sum_{i=1}^n \mu_A(x_i)/x_i \right\}$$

untuk X diskrit, atau

$$A = \left\{ \int_X \mu_A(x)/x \right\}$$

untuk X menerus (*continue*). Lambang \int bukan berarti integral seperti di dalam kalkulus.

Contoh 2.53.

(i) diskrit

$X = \text{himpunan bilangan bulat positif}$

$$A = \text{bilangan bulat yang dekat } 10 = \{0.1/7 + 0.5/8 + 1.0/10, 0.8/11 + 0.5/12 + 0.1/13\}$$

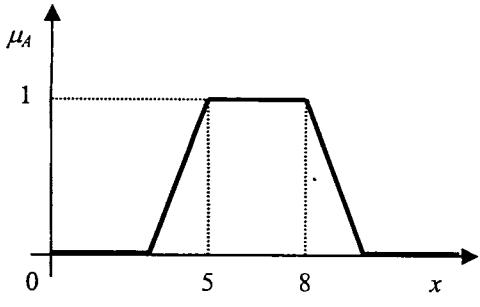
(ii) menerus

$X = \text{himpunan bilangan riil positif}$

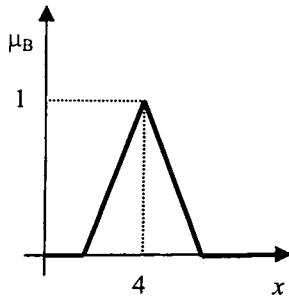
$$A = \text{bilangan riil yang dekat } 10 = \int 1/(1 + (x - 10)^2)/x$$

Operasi Himpunan *Fuzzy*

Misalkan himpunan *fuzzy* A dan himpunan *fuzzy* B masing-masing memiliki fungsi keanggotaan yang grafiknya adalah seperti pada Gambar 2.11.



(a)



(b)

Gambar 2.11 Grafik fungsi keanggotaan himpunan fuzzy A dan B .

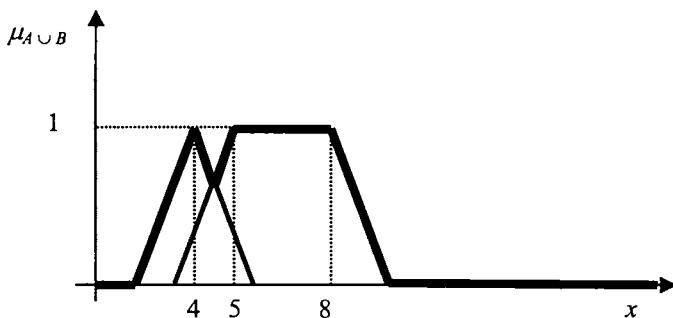
Operasi-operasi pada himpunan fuzzy didefinisikan sebagai berikut:

1. Gabungan

$$A \cup B \rightarrow \mu_{A \cup B} = \mu_A(x) \vee \mu_B(x) = \max(\mu_A(x), \mu_B(x))$$

$A \cup B$ diartikan sebagai “ x dekat A atau x dekat B ”.

Grafik fungsi keanggotaan $A \cup B$ digambarkan pada Gambar 2.12 (garis yang lebih tebal menunjukkan derajat keanggotaan hasil gabungan)

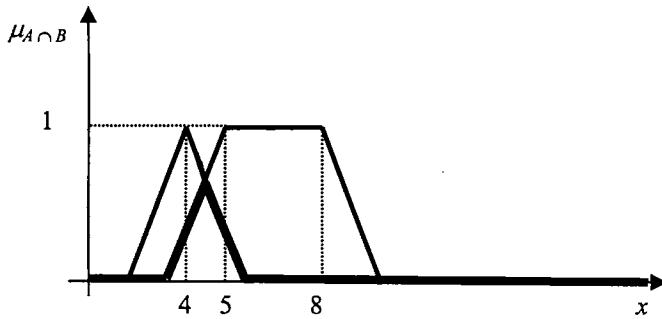
Gambar 2.12 Grafik fungsi keanggotaan himpunan $A \cup B$.

2. Irisan

$$A \cap B \rightarrow \mu_{A \cap B} = \mu_A(x) \wedge \mu_B(x) = \min(\mu_A(x), \mu_B(x))$$

$A \cap B$ diartikan sebagai “ x dekat A dan x dekat B ”.

Grafik fungsi keanggotaan $A \cap B$ digambarkan pada Gambar 2.13 (garis yang lebih tebal menunjukkan derajat keanggotaan hasil irisan).



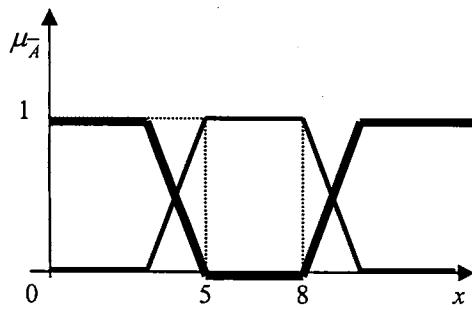
Gambar 2.13 Grafik fungsi keanggotaan himpunan $A \cap B$.

3. Komplemen

$$\overline{A} \rightarrow \mu_{\overline{A}} = 1 - \mu_A(x)$$

\overline{A} diartikan sebagai “ x tidak dekat A ”.

Grafik fungsi keanggotaan \overline{A} digambarkan pada Gambar 2.14 garis yang lebih tebal menunjukkan derajat keanggotaan hasil komplemen).



Gambar 2.14 Grafik fungsi keanggotaan himpunan \overline{A} .

Logika Fuzzy

Pada logika klasik, nilai kebenaran proposisi adalah 1 (*true*) atau 0 (*false*). Tetapi pada logika *fuzzy*, nilai kebenaran proposisi adalah nilai riil di dalam selang [0,1]. Misalkan p adalah proposisi yang didefinisikan pada himpunan *fuzzy* A , maka nilai kebenaran proposisi p adalah $T(p)$:

$$T(p) = \mu_A(x), \quad 0 \leq \mu_A \leq 1$$

Jadi, nilai kebenaran $p : x \in A$ sama dengan derajat keanggotaan x di dalam A .

Dua bentuk proposisi di dalam logika *fuzzy*:

1. Proposisi atomik, berbentuk “ x is A ” yang dalam hal ini, x adalah peubah linguistik dan A adalah terma/nilai linguistik

Contoh proposisi (dalam Bahasa Inggris): “*man is old*”

Jika $x = 50$ dan fungsi keanggotaan *old* adalah

$$\mu_{\text{old}} = \begin{cases} 0 & , x \leq 45 \\ (x - 45)/15 & , 45 < x < 60 \\ 1 & , x \geq 60 \end{cases}$$

maka nilai kebenaran “*42 is old*” adalah $(50 - 45)/15 = 1/3 = 0.333$

2. Proposisi majemuk, berbentuk:

“ x is A or y is B ”

“ x is A and y is B ”

Contoh (dalam Bahasa Inggris): “*temperature is cold or it is rainy*”

2.20 Ragam Contoh Soal dan Penyelesaian

Untuk lebih memantapkan pemahaman terhadap materi himpunan (tidak termasuk himpunan *fuzzy*), berikut ini diberikan sejumlah soal dan penyelesaiannya.

Contoh 2.54.

Misalkan A adalah himpunan. Periksalah apakah setiap pernyataan di bawah ini benar atau salah dan jika salah, bagaimana seharusnya:

- (a) $A \cap P(A) = P(A)$
- (b) $\{A\} \cup P(A) = P(A)$
- (c) $A - P(A) = A$
- (d) $\{A\} \in P(A)$
- (e) $A \subseteq P(A)$

Penyelesaian:

- (a) salah, seharusnya $A \cap P(A) = \emptyset$
 - (b) benar
 - (c) benar
 - (d) salah, seharusnya $\{A\} \subseteq P(A)$
 - (e) salah, seharusnya $A \in P(A)$
-

Contoh 2.55

[LIP00] Misalkan $A = \{1, 2, 3\}$ dan $B = \{1, 2, 3, 4, 5\}$. Tentukan semua kemungkinan himpunan C sedemikian sehingga $A \subset C$ dan $C \subset B$, yaitu A adalah *proper subset* dari C dan C adalah *proper subset* dari B .

Penyelesaian:

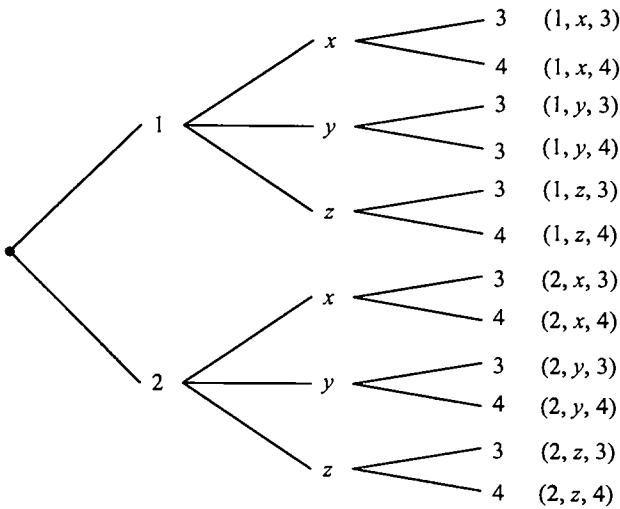
C harus mengandung semua elemen $A = \{1, 2, 3\}$ dan sekurang-kurangnya satu elemen dari B . Dengan demikian, $C = \{1, 2, 3, 4\}$ atau $C = \{1, 2, 3, 5\}$. C tidak boleh memuat 4 dan 5 sekaligus karena C adalah *proper subset* dari B .

Contoh 2.56.

Diberikan $A = \{1, 2\}$, $B = \{x, y, z\}$, dan $C = \{3, 4\}$. Tentukan $|A \times B \times C|$ dan $A \times B \times C$.

Penyelesaian:

$A \times B \times C$ terdiri dari semua tripel (a, b, c) yang dalam hal ini $a \in A$, $b \in B$, dan $c \in C$. Kardinalitas $A \times B \times C$, yaitu $|A \times B \times C| = |A| \cdot |B| \cdot |C| = 2 \cdot 3 \cdot 2 = 12$. Semua elemen dari $A \times B \times C$ dapat diperoleh secara sistematis dengan bantuan pohon berikut:



Jadi, $A \times B \times C = \{(1, x, 3), (1, x, 4), (1, y, 3), (1, y, 4), (1, z, 3), (1, z, 4), (2, x, 3), (2, x, 4), (2, y, 3), (2, y, 4), (2, z, 3), (2, z, 4)\}$. ■

Contoh 2.57

Jika A dan B masing-masing adalah himpunan, tunjukkan secara aljabar bahwa $(A \oplus B) \cap A$ dapat dinyatakan dengan $A - B$.

Penyelesaian:

$$\begin{aligned}
 (A \oplus B) \cap A &= ((A - B) \cup (B - A)) \cap A && (\text{Definisi operasi beda setangkup}) \\
 &= ((A \cap \bar{B}) \cup (B \cap \bar{A})) \cap A && (\text{Definisi selisih}) \\
 &= A \cap ((A \cap \bar{B}) \cup (B \cap \bar{A})) && (\text{Hukum Komutatif}) \\
 &= (A \cap (A \cap \bar{B})) \cup (A \cap (B \cap \bar{A})) && (\text{Hukum Distributif}) \\
 &= (A \cap (A \cap \bar{B})) \cup ((B \cap \bar{A}) \cap A) && (\text{Hukum Komutatif}) \\
 &= ((A \cap A) \cap \bar{B}) \cup (B \cap (\bar{A} \cap A)) && (\text{Hukum Asosiatif}) \\
 &= (A \cap \bar{B}) \cup (B \cap (\bar{A} \cap A)) && (\text{Hukum Idempoten}) \\
 &= (A \cap \bar{B}) \cup (B \cap \emptyset) && (\text{Hukum Komplemen}) \\
 &= (A \cap \bar{B}) \cup \emptyset && (\text{Hukum Null}) \\
 &= (A \cap \bar{B}) && (\text{Hukum Identitas}) \\
 &= A - B && (\text{Definisi Selisih})
 \end{aligned}$$

Contoh 2.58

Misalkan A , B , dan C adalah himpunan. Buktikan secara aljabar himpunan bahwa $A - (B \cup C) = (A - B) \cap (A - C)$.

Penyelesaian:

$$\begin{aligned}
 A - (B \cup C) &= A \cap \overline{B \cup C} && (\text{Definisi Selisih}) \\
 &= A \cap (\overline{B} \cup \overline{C}) && (\text{Hukum De Morgan}) \\
 &= (A \cap \overline{B}) \cap (A \cap \overline{C}) && (\text{Hukum Distributif}) \\
 &= (A - B) \cap (A - C) && (\text{Definisi operasi selisih})
 \end{aligned}$$

Contoh 2.59

Misalkan A adalah himpunan bagian dari himpunan semesta (U). Tuliskan hasil dari operasi beda-setangkup berikut?

$$(a) A \oplus U \quad (b) A \oplus \overline{A} \quad (c) \overline{A} \oplus U$$

Penyelesaian:

$$\begin{aligned}
 (a) A \oplus U &= (A - U) \cup (U - A) && (\text{Definisi operasi beda setangkup}) \\
 &= (\emptyset) \cup (A) && (\text{Definisi opearsi selisih}) \\
 &= \overline{A} && (\text{Hukum Identitas})
 \end{aligned}$$

$$\begin{aligned}
 (b) A \oplus \bar{A} &= (A - \bar{A}) \cup (\bar{A} - A) && \text{(Definisi operasi beda setangkup)} \\
 &= (A \cap A) \cup (\bar{A} \cap \bar{A}) && \text{(Definisi operasi selisih)} \\
 &= A \cup \bar{A} && \text{(Hukum Idempoten)} \\
 &= U && \text{(Hukum Komplemen)}
 \end{aligned}$$

$$\begin{aligned}
 (c) \quad \overline{A} \oplus U &= (\overline{A} \cup U) - (\overline{A} \cap U) && \text{(Definisi operasi beda setangkup)} \\
 &= U - \overline{A} && \text{(Hukum Null dan Hukum Identitas)} \\
 &= A && \text{(Definisi operasi selisih)}
 \end{aligned}$$

Contoh 2.60

Di antara bilangan bulat antara 101 – 600 (termasuk 101 dan 600 itu sendiri), berapa banyak bilangan yang tidak habis dibagi oleh 4 atau 5 namun tidak keduanya?

Penyelesaian:

Diketahui:

$$\begin{aligned} |U| &= 500 \\ |A| &= \lfloor 600/4 \rfloor - \lfloor 100/4 \rfloor = 150 - 25 = 125 \\ |B| &= \lfloor 600/5 \rfloor - \lfloor 100/5 \rfloor = 120 - 20 = 100 \\ |A \cap B| &= \lfloor 600/20 \rfloor - \lfloor 100/20 \rfloor = 30 - 5 = 25 \end{aligned}$$

yang ditanyakan $|\overline{A \oplus B}| = ?$

Hitung terlebih dahulu

$$|A \oplus B| = |A| + |B| - 2|A \cap B| = 125 + 100 - 50 = 175$$

untuk mendapatkan

$$|\overline{A \oplus B}| = U - |A \oplus B| = 500 - 175 = 325$$

Soal Latihan

1. Tentukan apakah pernyataan di bawah ini benar atau salah:
 - (a) $\{\emptyset\} \subseteq \{\emptyset\}$
 - (b) $\emptyset \in \{\emptyset\}$
 - (c) $\{\emptyset\} \in \{\emptyset\}$
 - (d) $\{a, b\} \subseteq \{a, b, \{\{a, b\}\}\}$
 - (e) Jika $A \subseteq B$ dan $B \in C$, maka $A \in C$
 - (f) Jika $A \in B$ dan $B \subseteq C$, maka $A \in C$.
 - (g) Jika $A = \{\emptyset, \{\emptyset\}\}$, maka $\emptyset \in 2^A$
 - (h) Jika $A = \{\emptyset, \{\emptyset\}\}$, maka $\{\{\emptyset\}\} \subseteq 2^A$
 - (i) $\emptyset \subseteq \emptyset$
 - (j) $\emptyset \in \emptyset$
 - (k) $\{\emptyset\} \in \emptyset$
 - (l) $\{a, b\} \subseteq \{a, b, c, \{\{a, b, c\}\}\}$
 - (m) $\{a, b\} \in \{a, b, c, \{\{a, b, c\}\}\}$
 - (n) $\{a, b\} \in \{a, b, \{\{a, b\}\}\}$
 - (o) jika $A \in B$ dan $B \subseteq C$, maka $A \subseteq C$
 - (p) jika $A \subseteq B$ dan $B \in C$, maka $A \subseteq C$
 - (q) $x \in \{x\}$
 - (r) $\{x\} \subseteq \{x\}$
 - (s) $\{x\} \in \{x\}$
 - (t) $\{x\} \in \{\{x\}\}$
 - (u) $\emptyset \subseteq \{x\}$
 - (v) $\emptyset \in \{x\}$
2. Jika $A = \{a, b, \{a, c\}, \emptyset\}$ dan $B = \{a, \{a\}, d, e\}$, tentukan himpunan berikut:
 - (a) $A - \emptyset$
 - (b) $A - \{\emptyset\}$
 - (c) $\{\{a, c\}\} - A$
 - (d) $A \oplus B$
 - (e) $\{a\} - \{A\}$
 - (f) $P(A-B)$
 - (g) $\emptyset - A$
 - (h) B^2
 - (i) $A \cup (B \cap A)$
 - (j) $A \cap P(A)$
3. (a) Tentukan himpunan kuasa dari himpunan $\{\emptyset, \{\emptyset\}\}$
(b) Berapa banyak elemen pada himpunan $P(\{\emptyset, a, \{a\}, \{\{a\}\}\})$?
4. Misalkan $A = \{\emptyset\}$ dan $B = P(P(A))$.
 - (a) Apakah $\emptyset \in B$? $\emptyset \subseteq B$?
 - (b) Apakah $\{\emptyset\} \in B$? $\{\emptyset\} \subseteq B$?
 - (c) Apakah $\{\{\emptyset\}\} \in B$? $\{\{\emptyset\}\} \subseteq B$?

5. Misalkan A adalah himpunan. Periksalah apakah setiap pernyataan di bawah ini benar atau salah dan jika salah, bagaimana seharusnya:
- $A \cap P(A) = A$
 - $A - P(A) = A$
 - $\{A\} \in P(A)$
 - $A \subseteq P(A)$
6. Didefinisikan A, B, C, D , dan E sebagai berikut:
 $A = \{1, 2, 3, 4\}$, $B = \{1, 2, \{2\}, \{\{4\}\}\}$,
 $C = \{1, \{1, 2\}, \{\{1, 2, 3\}\}\}$, $D = \{1, 2, 2, 1\}$.
- Untuk tiap W, X, Y, Z yang didefinisikan di bawah ini, nyatakan apakah ia adalah elemen atau himpunan bagian dari tiap-tiap himpunan A, B, C, D .
- $W = \{1, 3, 5\}$ $X = \{1, 2, 3\}$ $Y = \{4\}$ $Z = \{2\}$
7. Diketahui $A = \{+, -\}$, $B = \{00, 01, 10, 11\}$.
- Daftarkan $A \times B$
 - Berapa banyak elemen A^4 dan $(A \times B)^3$?
8. Periksalah apakah setiap pernyataan di bawah ini benar atau salah. Jelaskan secara ringkas jawaban anda.
- $A \cap P(A) = P(A)$
 - $A \cap P(A) = A$
 - $\{A\} \cup P(A) = P(A)$
 - $\{A\} \cap P(A) = A$
 - $A - P(A) = A$
9. Temukan dua buah himpunan sedemikian sehingga $A \in B$ dan $A \subseteq B$.
10. Misalkan A, B , dan C adalah himpunan. Tunjukkan bahwa
- $(A - C) \cap (C - B) = \emptyset$
 - $(B - A) \cup (C - A) = (B \cup C) - A$
 - $(A - B) - C \subseteq A - C$
11. Apa yang dapat dikatakan tentang himpunan A dan B jika kesamaan berikut benar?
- $A - B = B - A$
 - $A \cap B = B \cap A$
12. Dapatkah disimpulkan $A = B$ jika A, B , dan C adalah himpunan sedemikian sehingga
- $A \cup C = B \cup C$
 - $A \cap C = B \cap C$

13. Misalkan A , B , dan C adalah himpunan yang tidak kosong sedemikian sehingga $A \subseteq B$, $B \subseteq C$, dan $C \subseteq A$. Apakah yang dapat disimpulkan dari pernyataan tersebut.
14. Misalkan A , B , dan C adalah himpunan. Tunjukkan bahwa $(A - B) - C = (A - C) - (B - C)$.
15. Misalkan A , B , dan C himpunan. Tunjukkan bahwa $(A - B) - C = A - (B \cup C)$.
16. Buktikan hukum identitas: (i) $A \cup \emptyset = A$ dan (ii) $A \cap U = A$.
17. Buktikan bahwa jika $A \cup B \subseteq A \cap B$ maka $A = B$.
18. Buktikan untuk himpunan A , B , dan C , bahwa
(a) $A \cap (B - C) = (A \cap B) - (A \cap C)$
(b) $A - (B - C) = (A - B) \cup (A \cap C)$
(c) jika $A \cap B \subseteq C$, maka $B - C \subseteq \overline{A}$
19. Misalkan A , B , dan C adalah himpunan. Pada kondisi manakah pernyataan di bawah ini benar?
(a) $(A - B) \cap (A - C) = \emptyset$
(b) $(A - B) \oplus (A - C) = \emptyset$
20. *Successor* dari himpunan A didefinisikan sebagai $A \cup \{A\}$. Tentukan *successor* dari himpunan
(a) $\{1, 2, 3\}$ (b) \emptyset (c) $\{\emptyset\}$ (d) $\{\emptyset, \{\emptyset\}\}$
21. Jika diketahui $(A \cap C) \subseteq (B \cap C)$ dan $(A \cap C) \subseteq (B \cap C)$, tunjukkan bahwa $A \subseteq B$.
22. Jika diketahui $A \cap B = A \cap C$, apakah berarti bahwa $B = C$?
23. Misalkan A himpunan mahasiswa tahun pertama, B himpunan mahasiswa tahun kedua, C himpunan mahasiswa Jurusan Matematika, D himpunan mahasiswa jurusan Teknik Informatika, E himpunan mahasiswa yang mengambil kuliah Matematika Diskrit, F himpunan mahasiswa yang menonton pertunjukan pantomim pasa Senin malam, G himpunan mahasiswa yang begadang sampai lewat tengah malam pada hari Senin malam. Nyatakan pernyataan berikut dalam notasi teori himpunan:
(a) Semua mahasiswa tahun kedua jurusan Teknik Informatika mengambil kuliah Matematika Diskrit.

- (b) Hanya mereka yang mengambil kuliah Matematika Diskrit atau yang yang pergi nonton pertunjukan pantomim yang begadang sampai lewat tengah malam pada hari Senin malam.
- (c) Mahasiswa yang mengambil kuliah Matematika Diskrit tidak ada yang pergi nonton pertunjukan pantomim pada Senin malam. (Penyebabnya adalah tugas pekerjaan rumah yang sangat banyak dalam kuliah Matematika Diskrit).
- (d) Pertunjukan pantomim itu hanya untuk mahasiswa tahun pertama dan mahasiswa tahun kedua.
- (e) Semua mahasiswa tahun kedua yang bukan dari Jurusan Matematika atau pun Jurusan Teknik Informatika pergi nonton pertunjukan pantomim.
24. Di antara bilangan bulat 1 sampai 300 (termasuk 1 dan 300 sendiri), berapa banyak yang tidak habis dibagi 3 atau 5?
25. Di antara bilangan bulat 1 - 300, berapa banyak bilangan yang habis dibagi 3 tetapi tidak habis dibagi 5 maupun 7?
26. Di antara 100 mahasiswa, 32 orang mempelajari matematika, 20 orang mempelajari fisika, 45 orang mempelajari biologi, 15 mempelajari matematika dan biologi, 7 mempelajari matematika dan fisika, 10 mempelajari fisika dan biologi, dan 30 tidak mempelajari satu pun di antara ketiga bidang tersebut.
- (a) Hitunglah banyaknya mahasiswa yang mempelajari ketiga bidang tersebut.
- (b) Hitunglah banyaknya mahasiswa yang mempelajari hanya satu di antara ketiga bidang tersebut.
27. Enam puluh ribu suporter sepakbola yang mendukung pertandingan di kandang sendiri membeli habis semua cindera mata untuk mobil mereka. Secara keseluruhan laku terjual 20000 stiker, 36000 bendera kecil, dan 12000 gantungan kunci. Kita diberitahu bahwa 52000 suporter membeli sedikitnya satu cindera mata dan tidak seorangpun membeli suatu cindera mata lebih dari satu. Selain itu, 6000 suporter membeli bendera kecil dan gantungan kunci, 9000 membeli bendera kecil dan stiker, dan 5000 membeli gantungan kunci dan stiker.
- (a) Berapa banyak suporter yang membeli ketiga macam cindera mata di atas?
- (b) Berapa banyak suporter yang membeli tepat satu cindera mata?
28. Di antara 50 mahasiswa di dalam kelas, 26 orang memperoleh nilai A dari ujian pertama dan 21 orang memperoleh nilai A dari ujian kedua. Jika 17 orang mahasiswa tidak memperoleh nilai A dari ujian pertama maupun ujian kedua, berapa banyak mahasiswa yang memperoleh dua kali nilai A dari kedua ujian itu?

29. Dalam suatu survei pada 60 orang, didapatkan bahwa 25 orang membaca majalah *Tempo*, 26 orang membaca majalah *Gatra*, dan 26 orang membaca majalah *Intisari*. Juga terdapat 9 orang yang membaca *Tempo*, dan *Intisari*, 11 orang membaca *Tempo* dan *Gatra*, 8 orang membaca *Gatra* dan *Intisari*, dan 8 orang tidak membaca majalah satupun.
Tentukan jumlah orang yang membaca ketiga majalah tersebut.
Tentukan jumlah orang yang benar-benar membaca satu majalah.
30. Tentukan semua partisi dari himpunan $B = \{\emptyset, 1, 2, \{2\}, \{\{4\}\}\}$.
31. Diketahui multiset $P = \{0, 0, 1, 1, 1, 1, 2, 2, 3\}$ dan $Q = \{0, 1, 2, 3, 3, 3, 3, 4, 4\}$. Tentukan (a) $P \cup Q$ (b) $P \cap Q$ (c) $P - Q$ (d) $P + Q$
32. Salah satu kemungkinan mendefinisikan beda setangkup anta dua himpunan-ganda P dan Q , dilambangkan dengan $P \oplus Q$, ialah membuat multiplisitas setiap unsur di dalam $P \oplus Q$ sama dengan nilai mutlak selisih antara multiplisitas unsur tersebut di dalam P dan di dalam Q . Ketidaktaat-asasan (*inconsistency*) yang bagaimanakah yang mungkin timbul dengan definisi demikian ini? Petunjuk: perhatikan himpunan ganda $(P \oplus Q) \oplus R$ dan himpunan-ganda $P \oplus (Q \oplus R)$.

BAB 3

Matriks, Relasi, dan Fungsi

Sebuah masalah yang telah jelas digambarkan berarti telah terselesaikan sebagian.
(C.F. Kettering)

Hubungan (*relationship*) antara elemen himpunan dengan elemen himpunan lainnya sering dijumpai pada banyak masalah. Misalnya hubungan antara mahasiswa dengan mata kuliah yang diambil, hubungan antara orang dengan kerabatnya, hubungan antara bilangan genap dengan bilangan yang habis dibagi 2, dan sebagainya. Di dalam ilmu komputer, contoh hubungan itu misalnya hubungan antara program komputer dengan peubah yang digunakan, hubungan antara bahasa pemrograman dengan pernyataan (*statement*) yang sah, hubungan antara *plaintext* dan *chipertext* pada bidang kriptografi, dan sebagainya.

Hubungan antara elemen himpunan dengan elemen himpunan lain dinyatakan dengan struktur yang disebut **relasi**. Di dalam bab ini kita akan membicarakan relasi dan sifat-sifatnya, serta jenis khusus relasi yang disebut **fungsi**.

Sebelum membahas relasi dan fungsi, Bab 4 ini akan dimulai dengan pokok bahasan **matriks**. Di dalam matematika diskrit matriks digunakan untuk merepresentasikan

struktur diskrit. Struktur diskrit adalah struktur matematika abstrak yang digunakan untuk merepresentasikan objek-objek diskrit dan hubungan antara objek-objek tersebut. Struktur diskrit yang direpresentasikan dengan matriks antara lain relasi, graf, dan pohon.

3.1 Matriks

Materi matriks mungkin sudah tidak asing bagi mahasiswa karena matriks sudah dipelajari sejak di bangku sekolah menengah. Upabab ini hanya merangkum kembali materi yang berkaitan dengan matriks dan relevan dengan matematika diskrit.

DEFINISI 3.1. Matriks adalah susunan skalar elemen-elemen dalam bentuk baris dan kolom. Matriks A yang berukuran dari m baris dan n kolom ($m \times n$) adalah:

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

Entri a_{ij} disebut elemen matriks pada baris ke- i dan kolom ke- j . Jika $m = n$, maka matriks tersebut dinamakan juga matriks bujursangkar (*square matrix*). Menuliskan matriks dalam bentuk persegi panjang di atas adalah boros tempat, oleh karena itu kita lazim menuliskan matriks dengan notasi ringkas $A = [a_{ij}]$.

Contoh 3.2

Di bawah ini adalah sebuah matriks yang berukuran 3×4 :

$$A = \begin{bmatrix} 2 & 5 & 0 & 6 \\ 8 & 7 & 5 & 4 \\ 3 & 1 & 1 & 8 \end{bmatrix}$$

Matriks di atas disusun oleh 3 baris elemen, yaitu $(2, 5, 0, 6)$, $(8, 7, 5, 4)$, $(3, 1, 1, 8)$, atau susunan dalam bentuk kolom-kolom:

$$\begin{bmatrix} 2 \\ 8 \\ 3 \end{bmatrix}, \begin{bmatrix} 5 \\ 7 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 5 \\ 1 \end{bmatrix}, \text{ dan } \begin{bmatrix} 6 \\ 4 \\ 8 \end{bmatrix}.$$

Beberapa Matriks Khusus

Terdapat beberapa matriks khusus yang ditemukan dalam pembahasan matematika, antara lain matriks diagonal, matriks identitas, dan matriks *transpose*.

1. Matriks diagonal

Matriks diagonal adalah matriks bujursangkar dengan $a_{ij} = 0$ untuk $i \neq j$. Dengan kata lain, seluruh elemen yang tidak terdapat pada posisi $i \neq j$ bernilai 0.

Contoh 3.2

Di bawah ini adalah contoh-contoh matriks diagonal yang berukuran 3×3 :

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix}, \quad \begin{bmatrix} 2 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

2. Matriks identitas

Matriks identitas, dilambangkan dengan I , adalah matriks diagonal dengan semua elemen diagonal = 1.

Contoh 3.3

Di bawah ini adalah contoh-contoh matriks I , masing masing 3×3 dan 4×4 .

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3. Matriks segitiga atas/bawah

Matriks segitiga atas/bawah adalah matrik jika elemen-elemen di atas/di bawah diagonal bernilai 0, yaitu $a_{ij} = 0$ jika $i < j$ ($i > j$).

Contoh 3.4

Di bawah ini adalah contoh matriks segitiga. Yang pertama matriks segitiga atas dan yang kedua matriks segitiga bawah.

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 5 & 7 & 0 & 0 \\ 6 & 0 & 3 & 0 \\ 2 & 4 & -2 & 6 \end{bmatrix}, \quad \begin{bmatrix} 2 & 6 & 6 & -4 \\ 0 & 3 & 7 & 3 \\ 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 8 \end{bmatrix}$$



4. Matriks transpose

Matriks *transpose* adalah matriks yang diperoleh dengan mempertukarkan baris-baris dan kolom-kolom. Misalkan $A = [a_{ij}]$ berukuran $m \times n$, maka *transpose* dari matriks A , ditulis A^T , adalah matriks $n \times m$ yang dalam hal ini jika $A^T = [b_{ij}]$, maka $b_{ij} = a_{ji}$ untuk $i = 1, 2, \dots, n$ dan $j = 1, 2, \dots, m$.

Contoh 3.5

Di bawah ini adalah sebuah matriks A dan transpose-nya, A^T .

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}, \quad A^T = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}$$



5. Matriks setangkup (symmetry)

A adalah matriks setangkup atau simetri jika $A^T = A$, yaitu jika $a_{ij} = a_{ji}$ untuk setiap i dan j . Dengan kata lain, pada matriks setangkup elemen di bawah diagonal adalah hasil pencerminan dari elemen di atas diagonal terhadap sumbu diagonal matriks.

Contoh 3.6

Di bawah ini adalah contoh-contoh matriks setangkup.

$$\begin{bmatrix} 1 & 5 & 6 & 2 \\ 5 & 7 & 0 & 4 \\ 6 & 0 & 3 & -2 \\ 2 & 4 & -2 & 6 \end{bmatrix}, \quad \begin{bmatrix} 2 & 6 & 6 & -4 \\ 6 & 3 & 7 & 3 \\ 6 & 7 & 0 & 2 \\ -4 & 3 & 2 & 8 \end{bmatrix}$$



6. Matriks 0/1 (zero-one)

Matriks 0/1 adalah matriks yang setiap elemennya hanya bernilai 0 atau 1. Matriks ini banyak digunakan untuk merepresentasikan relasi keterhubungan (lihat upabab 3.3 mengenai representasi relasi).

Contoh 3.7

Di bawah ini adalah contoh matriks 0/1:

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

■

Operasi Aritmetika Matriks

Operasi aritmetika yang biasa dilakukan terhadap matriks adalah operasi penjumlahan dan perkalian dua buah matriks, serta perkalian matriks dengan sebuah skalar.

1. Penjumlahan dua buah matriks

Dua buah matriks dapat dijumlahkan jika ukuran keduanya sama. Misalkan $A = [a_{ij}]$ dan $B = [b_{ij}]$ yang masing-masing berukuran $m \times n$. Jumlah A dan B , dilambangkan dengan $A + B$, menghasilkan matriks $C = [c_{ij}]$ yang berukuran $m \times n$, yang dalam hal ini $c_{ij} = a_{ij} + b_{ij}$ untuk setiap i dan j .

Catatan:

operasi pengurangan sama dengan operasi penjumlahan, tetapi dengan mengganti operator + dengan operator -.

Contoh 3.8

Penjumlahan dua buah matriks:

$$\begin{bmatrix} 1 & 2 & 3 \\ 0 & 5 & -2 \\ 4 & 7 & 8 \end{bmatrix} + \begin{bmatrix} 5 & 6 & 8 \\ 7 & -3 & 9 \\ 6 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 1+5 & 2+6 & 3+8 \\ 0+7 & 5-3 & -2+9 \\ 4+6 & 7+2 & 8+1 \end{bmatrix} = \begin{bmatrix} 6 & 8 & 11 \\ 7 & 2 & 7 \\ 10 & 9 & 9 \end{bmatrix}$$

■

2. Perkalian dua buah matriks

Dua buah matriks dapat dikalikan jika jumlah kolom matriks pertama sama dengan jumlah baris matriks kedua. Misalkan $A = [a_{ij}]$ adalah matriks $m \times n$ dan $B = [b_{ij}]$ adalah matriks $n \times p$. Maka, perkalian A dan B , dilambangkan dengan AB , menghasilkan matriks $C = [c_{ij}]$ yang berukuran $m \times p$, yang dalam hal ini

$$c_{ij} = a_{i1}b_{1j} + a_{i2}b_{2j} + \cdots + a_{in}b_{nj} = \sum_{k=1}^n a_{ik}b_{kj}$$

Contoh 3.9

Perkalian dua buah matriks:

$$\begin{bmatrix} 1 & 3 \\ 2 & -1 \end{bmatrix} \begin{bmatrix} 2 & 0 & -4 \\ 3 & -2 & 6 \end{bmatrix} = \begin{bmatrix} (1)(2) + (3)(3) & (1)(0) + 3(-2) & (1)(-4) + (1)(6) \\ (2)(2) + (-1)(3) & (2)(0) + (-2)(-2) & (2)(-4) + (-1)(6) \end{bmatrix}$$
$$= \begin{bmatrix} 11 & -6 & 14 \\ 1 & 2 & -14 \end{bmatrix}$$

Sifat-sifat operasi perkalian matriks:

1. Perkalian matriks tidak komutatif, yaitu $AB \neq BA$.
2. Hukum asosiatif berlaku pada operasi matriks: $(AB)C = A(BC)$
3. Hukum distributif berlaku pada operasi matriks:
 - (i) $A(B + C) = AB + AC$ (hukum distributif kiri)
 - (ii) $(B + C)A = BA + CA$ (hukum distributif kanan)
4. Perkalian matriks dengan matriks identitas I tidak mengubah matriks, yaitu $AI = IA = A$.
5. Perpangkatan matriks didefinisikan sebagai berikut: $A^0 = I$, $A^k = \underbrace{AAA\cdots A}_{k \text{ kali}}$.
6. A adalah matriks ortogonal jika $AA^T = A^TA = I$.

3. Perkalian matriks dengan skalar.

Misalkan k adalah sebuah skalar. Perkalian matriks A dengan skalar k adalah mengalikan setiap elemen matriks dengan k .

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

$$kA = \begin{bmatrix} ka_{11} & ka_{12} & \cdots & ka_{1n} \\ ka_{21} & ka_{22} & \cdots & ka_{2n} \\ \vdots & \vdots & & \vdots \\ ka_{m1} & ka_{m2} & \cdots & ka_{mn} \end{bmatrix}$$

Contoh 3.10

Misalkan

$$A = \begin{bmatrix} 2 & 1 & 0 \\ 3 & 7 & 5 \\ -2 & 0 & 4 \end{bmatrix} \text{ dan } k = 3,$$

maka

$$3A = \begin{bmatrix} 3 \times 2 & 3 \times 1 & 3 \times 0 \\ 3 \times 3 & 3 \times 7 & 3 \times 5 \\ 3 \times (-2) & 3 \times 0 & 3 \times 4 \end{bmatrix} = \begin{bmatrix} 6 & 3 & 0 \\ 9 & 21 & 15 \\ -6 & 0 & 12 \end{bmatrix}$$

■

3.2 Relasi

Di dalam Bab 2 mengenai himpunan kita sudah mengenal pasangan terurut (*ordered pairs*). Cara yang paling mudah menyatakan hubungan antara elemen dari dua himpunan adalah dengan himpunan pasangan terurut. Himpunan pasangan terurut diperoleh dari perkalian kartesian (*cartesian product*) antara dua himpunan.

Kita tulis kembali definisi perkalian kartesian. Perkalian kartesian (*cartesian products*) dari himpunan A dan B adalah himpunan yang elemennya semua pasangan terurut (*ordered pairs*) yang mungkin terbentuk dengan komponen pertama dari himpunan A dan komponen kedua dari himpunan B .

Notasi: $A \times B = \{(a, b) \mid a \in A \text{ dan } b \in B\}$

Relasi antara himpunan A dan B –disebut **relasi biner**– didefinisikan sebagai berikut:

DEFINISI 3.2. Relasi biner R antara A dan B adalah himpunan bagian dari $A \times B$.

Notasi: $R \subseteq (A \times B)$.

Jika $(a, b) \in R$, kita gunakan notasi $a R b$ yang artinya a dihubungkan dengan b oleh R , dan jika $(a, b) \notin R$, kita gunakan notasi $a \not R b$ yang artinya a tidak dihubungkan oleh b oleh relasi R . Himpunan A disebut **daerah asal** (*domain*) dari R , dan himpunan B disebut **daerah hasil** (*range* atau *codomain*) dari R .

Contoh 3.11

Misalkan $A = \{\text{Amir, Budi, Cecep}\}$ adalah himpunan nama mahasiswa, dan $B = \{\text{IF221, IF251, IF342, IF323}\}$ adalah himpunan kode mata kuliah di Jurusan Teknik Informatika. Perkalian kartesian antara A dan B menghasilkan himpunan pasangan terurut yang jumlah anggotanya adalah $|A| \cdot |B| = 3 \cdot 4 = 12$ buah, yaitu

$$A \times B = \{(\text{Amir, IF221}), (\text{Amir, IF251}), (\text{Amir, IF342}), (\text{Amir, IF323}), (\text{Budi, IF221}), (\text{Budi, IF251}), (\text{Budi, IF342}), (\text{Budi, IF323}), (\text{Cecep, IF221}), (\text{Cecep, IF251}), (\text{Cecep, IF342}), (\text{Cecep, IF323})\}$$

Misalkan R adalah relasi yang menyatakan mata kuliah yang diambil oleh mahasiswa pada Semester Ganjil, yaitu

$$R = \{(Amir, IF251), (Amir, IF323), (Budi, IF221), (Budi, IF251), (Cecep, IF323)\}$$

Kita dapat melihat bahwa $R \subseteq (A \times B)$, A adalah daerah asal R , dan B adalah daerah hasil R . Oleh karena $(Amir, IF251) \in R$, kita dapat menuliskan Amir R IF251, tetapi $(Amir, IF342) \notin R$ sehingga kita menuliskan Amir R IF342. ■

Contoh 3.12

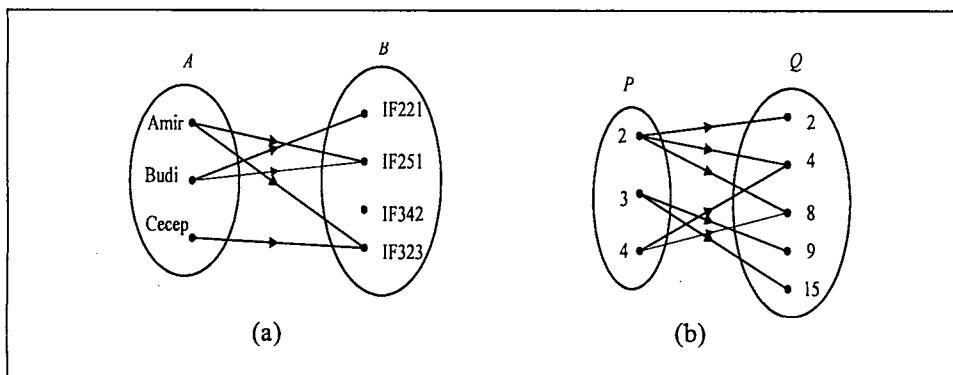
Misalkan $P = \{2, 3, 4\}$ dan $Q = \{2, 4, 8, 9, 15\}$. Jika kita definisikan relasi R dari P ke Q dengan

$$(p, q) \in R \text{ jika } p \text{ habis membagi } q$$

maka kita peroleh

$$R = \{(2, 2), (2, 4), (4, 4), (2, 8), (4, 8), (3, 9), (3, 15)\}$$
 ■

Pasangan terurut pada relasi dari himpunan A ke himpunan B dapat digambarkan dengan diagram panah. Gambarkan dua buah cakram/lingkaran, lalu tuliskan elemen-elemen A dan B pada masing-masing cakram. Jika $(a, b) \in R$, gambarkan panah dari a ke b yang menyatakan a berelasi dengan b . Diagram panah untuk masing-masing relasi pada Contoh 3.11 dan Contoh 3.12 diperlihatkan pada Gambar 3.1(a) dan (b).



Gambar 3.1 Diagram panah masing-masing untuk (a) Contoh 3.11 dan (b) Contoh 3.12

Daerah asal dan daerah hasil relasi bisa saja merupakan himpunan yang sama. Ini berarti relasi hanya didefinisikan pada sebuah himpunan. Misalnya R adalah relasi yang didefinisikan pada himpunan orang yang dalam hal ini $(x, y) \in R$ jika x adalah ibu dari y . Relasi yang didefinisikan hanya pada sebuah himpunan

adalah relasi yang khusus. Definisi relasi khusus ini dikemukakan dengan definisi berikut:

DEFINISI 3.3. Relasi pada himpunan A adalah relasi dari $A \times A$.

Dengan kata lain, relasi pada himpunan A adalah himpunan bagian dari $A \times A$. Contoh 3.13 mengilustrasikan relasi semacam ini.

Contoh 3.13

Misalkan R adalah relasi pada $A = \{2, 3, 4, 8, 9\}$ yang didefinisikan oleh $(x, y) \in R$ jika x adalah faktor prima dari y . Maka

$$R = \{(2, 2), (2, 4), (2, 8), (3, 3), (3, 9)\}$$

3.3 Representasi Relasi

Selain dinyatakan sebagai himpunan pasangan terurut, ada banyak cara lain untuk merepresentasikan atau menyajikan relasi. Di bawah ini disajikan 3 cara yang lazim dipakai untuk merepresentasikan relasi, yaitu dengan tabel, matriks, dan graf berarah.

3.3.1 Representasi Relasi dengan Tabel

Relasi biner dapat direpresentasikan sebagai tabel. Kolom pertama tabel menyatakan daerah asal, sedangkan kolom kedua menyatakan daerah hasil.

Relasi R pada Contoh 3.11 dapat dinyatakan dengan Tabel 3.1, sedangkan relasi R pada Contoh 3.12 dinyatakan dengan Tabel 3.2.

Tabel 3.1

A	B
Amir	IF251
Amir	IF323
Budi	IF221
Budi	IF251
Cecep	IF323

Tabel 3.2

P	Q
2	2
2	4
4	4
2	8
4	8
3	9
3	15

Kita tidak merepresentasikan relasi pada sebuah himpunan dengan tabel, karena tidak lazim dilakukan.

3.3.2 Representasi Relasi dengan Matriks

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

$$M = \begin{bmatrix} & b_1 & b_2 & \cdots & b_n \\ a_1 & m_{11} & m_{12} & \cdots & m_{1n} \\ a_2 & m_{21} & m_{22} & \cdots & m_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_m & m_{m1} & m_{m2} & \cdots & m_{mn} \end{bmatrix}$$

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 . Matriks representasi relasi merupakan contoh matriks *zero-one*.

Relasi R pada Contoh 3.11 dapat dinyatakan dengan matriks

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

yang dalam hal ini, $a_1 = \text{Amir}$, $a_2 = \text{Budi}$, $a_3 = \text{Cecep}$, dan $b_1 = \text{IF221}$, $b_2 = \text{IF251}$, $b_3 = \text{IF342}$, dan $b_4 = \text{IF323}$.

Relasi R pada Contoh 3.12 dapat dinyatakan dengan matriks

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

yang dalam hal ini, $a_1 = 2$, $a_2 = 3$, $a_3 = 4$, dan $b_1 = 2$, $b_2 = 4$, $b_3 = 8$, $b_4 = 9$, $b_5 = 15$.

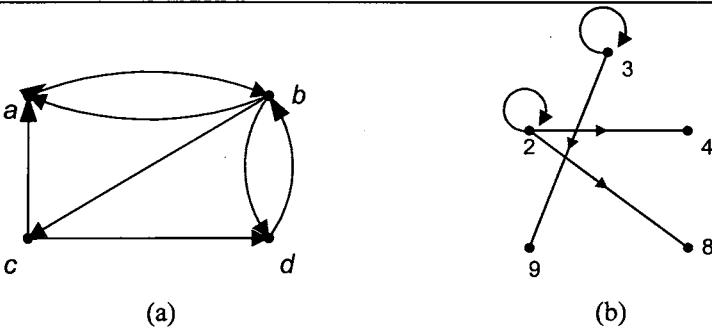
Relasi R pada sebuah himpunan adalah unik, yaitu matriks bujursangkar. Relasi pada sebuah himpunan pada Contoh 3.13 dapat dinyatakan dengan matriks

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

yang dalam hal ini, $a_1 = 2$, $a_2 = 3$, $a_3 = 4$, $a_4 = 8$, $a_5 = 9$, dan $b_1 = 2$, $b_2 = 3$, $b_3 = 4$, $b_4 = 8$, $b_5 = 9$.

3.3.3 Representasi Relasi dengan Graf Berarah

Relasi pada sebuah himpunan dapat direpresentasikan secara grafis dengan **graf berarah** (*directed graph* atau *digraph*) (Graf akan dibahas secara khusus di dalam Bab 8). Tiap elemen himpunan dinyatakan dengan sebuah titik (disebut juga simpul atau *vertex*), dan tiap pasangan terurut dinyatakan dengan busur (*arc*) yang arahnya ditunjukkan dengan sebuah panah. Dengan kata lain, jika $(a, b) \in R$, maka sebuah busur dibuat dari simpul a ke simpul b . Simpul a disebut **simpul asal** (*initial vertex*) dan simpul b disebut **simpul tujuan** (*terminal vertex*).



Gambar 3.2. (a) Representasi graf untuk relasi $R = \{(a, a), (a, b), (b, a), (b, c), (b, d), (c, a), (c, d), (d, b)\}$
(b) Representasi graf untuk relasi $R = \{(2, 2), (2, 4), (2, 8), (3, 3), (3, 9)\}$ pada Contoh 3.13.

Pasangan terurut (a, a) dinyatakan dengan busur dari simpul a ke simpul a sendiri. Busur semacam itu disebut **gelang** atau **kalang** (*loop*).

Sebagai contoh, misalkan $R = \{(a, a), (a, b), (b, a), (b, c), (b, d), (c, a), (c, d), (d, b)\}$ adalah relasi pada himpunan $\{a, b, c, d\}$. R direpresentasikan dengan graf berarah pada Gambar 3.2 (a). Perhatikan bahwa a mempunyai busur ke simpul a lagi. Busur yang mempunyai simpul asal sama dengan simpul tujuan dinamakan **kalang** (*loop*).

Relasi R pada Contoh 3.13 direpresentasikan dengan graf berarah pada Gambar 3.2(b).

Representasi dengan graf berarah umumnya digunakan untuk relasi pada sebuah himpunan. Relasi biner dari himpunan A ke himpunan B sebenarnya dapat juga direpresentasikan dengan graf berarah seperti diagram panah pada Gambar 3.1(a). Setiap elemen pada himpunan A dan dinyatakan dengan sebuah simpul pada dan pada sisi yang lain setiap elemen pada himpunan B dinyatakan dengan sebuah simpul. Beberapa literatur matematika tidak mendefinisikan representasi graf berarah untuk relasi biner. Buku ini pun hanya menggunakan graf berarah untuk merepresentasikan relasi pada sebuah himpunan.

3.4 Relasi Inversi

Jika R adalah relasi pada himpunan orang-orang di mana $(a, b) \in R$ jika a adalah ayah dari b , maka kita dapat membuat kebalikan relasinya, yaitu (b, a) yang menyatakan b adalah anak dari a . Relasi baru tersebut dinamakan inversi dari relasi semula. Begitu pula relasi “lebih besar dari” mempunyai inversi “lebih kecil dari”, relasi “lebih tua dari” mempunyai inversi “lebih muda dari”.

Secara umum, jika diberikan relasi R dari himpunan A ke himpunan B , kita bisa mendefinisikan relasi baru dari B ke A dengan cara membalik urutan dari setiap pasangan terurut di dalam R . Relasi Definisi relasi inversi adalah sebagai berikut:

DEFINISI 3.4. Misalkan R adalah relasi dari himpunan A ke himpunan B . Inversi dari relasi R , dilambangkan dengan R^{-1} , adalah relasi dari B ke A yang didefinisikan oleh

$$R^{-1} = \{(b, a) \mid (a, b) \in R\}$$

Contoh 3.14

Misalkan $P = \{2, 3, 4\}$ dan $Q = \{2, 4, 8, 9, 15\}$. Jika kita definisikan relasi R dari P ke Q dengan

$$(p, q) \in R \text{ jika } p \text{ habis membagi } q$$

maka kita peroleh

$$R = \{(2, 2), (2, 4), (4, 4), (2, 8), (4, 8), (3, 9), (3, 15)\}$$

R^{-1} adalah *invers* dari relasi R , yaitu relasi dari Q ke P dengan

$$(q, p) \in R^{-1} \text{ jika } q \text{ adalah kelipatan dari } p$$

maka kita peroleh

$$R^{-1} = \{(2, 2), (4, 2), (4, 4), (8, 2), (8, 4), (9, 3), (15, 3)\}$$

Jika M adalah matriks yang merepresentasikan relasi R ,

$$M = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 \end{bmatrix}$$

maka matriks yang merepresentasikan relasi R^{-1} , misalkan N , diperoleh dengan melakukan transpose terhadap matriks M ,

$$N = M^T = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

3.5 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. Hasil operasi tersebut juga berupa relasi. Dengan kata lain, jika R_1 dan R_2 masing-masing adalah relasi dari himpunan A ke himpunan B , maka operasi $R_1 \cap R_2$, $R_1 \cup R_2$, $R_1 - R_2$, dan $R_1 \oplus R_2$ juga adalah relasi dari A ke B .

Contoh 3.15

Misalkan $A = \{a, b, c\}$ dan $B = \{a, b, c, d\}$. Relasi $R_1 = \{(a, a), (b, b), (c, c)\}$ dan relasi $R_2 = \{(a, a), (a, b), (a, c), (a, d)\}$ adalah relasi dari A ke B . Kita dapat mengkombinasikan kedua buah relasi tersebut untuk memperoleh/

$$\begin{aligned} R_1 \cap R_2 &= \{(a, a)\} \\ R_1 \cup R_2 &= \{(a, a), (b, b), (c, c), (a, b), (a, c), (a, d)\} \\ R_1 - R_2 &= \{(b, b), (c, c)\} \\ R_2 - R_1 &= \{(a, b), (a, c), (a, d)\} \\ R_1 \oplus R_2 &= \{(b, b), (c, c), (a, b), (a, c), (a, d)\} \end{aligned}$$

Jika relasi R_1 dan R_2 masing-masing dinyatakan dengan matriks M_{R1} dan M_{R2} , maka matriks yang menyatakan gabungan dan irisan dari kedua relasi tersebut adalah

$$M_{R1 \cup R2} = M_{R1} \vee M_{R2} \quad \text{dan} \quad M_{R1 \cap R2} = M_{R1} \wedge M_{R2}$$

yang dalam hal ini, operator “ \vee ” berarti “atau” dan “ \wedge ” berarti “dan”. Contoh 3.16 berikut mengilustrasikan perhitungan relasi dengan menggunakan matriks.

Contoh 3.16

Misalkan bahwa relasi R_1 dan R_2 pada himpunan A dinyatakan oleh matriks

$$R_1 = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} \quad \text{dan} \quad R_2 = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

maka matriks yang menyatakan $R_1 \cup R_2$ dan $R_1 \cap R_2$ adalah

$$M_{R_1 \cup R_2} = M_{R_1} \vee M_{R_2} = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix}; \quad M_{R_1 \cap R_2} = M_{R_1} \wedge M_{R_2} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

3.6 Komposisi Relasi

Cara lain mengkombinasikan relasi adalah dengan mengkomposisikan dua buah relasi atau lebih. Definisi dari komposisi dua buah relasi didefinisikan sebagai berikut.

DEFINISI 3.5. Misalkan R adalah relasi dari himpunan A ke himpunan B , dan S adalah relasi dari himpunan B ke himpunan C . Komposisi R dan S , dinotasikan dengan $S \circ R$, adalah relasi dari A ke C yang didefinisikan oleh

$$S \circ R = \{(a, c) \mid a \in A, c \in C, \text{ dan untuk beberapa } b \in B, (a, b) \in R \text{ dan } (b, c) \in S\}$$

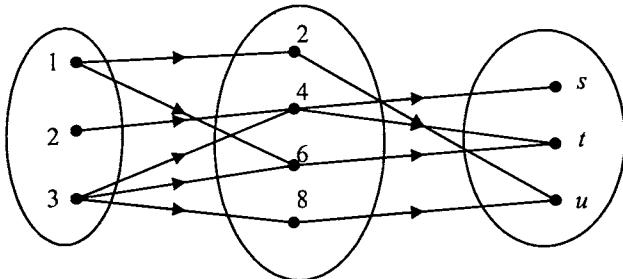
Dengan kata lain, menurut Definisi 3.5, kita menerapkan relasi R lebih dahulu, baru kemudian relasi S .

Contoh 3.17

Misalkan $R = \{(1, 2), (1, 6), (2, 4), (3, 4), (3, 6), (3, 8)\}$ adalah relasi dari himpunan $\{1, 2, 3\}$ ke himpunan $\{2, 4, 6, 8\}$ dan $S = \{(2, u), (4, s), (4, t), (6, t), (8, u)\}$ adalah relasi dari himpunan $\{2, 4, 6, 8\}$ ke himpunan $\{s, t, u\}$. Maka komposisi relasi R dan S adalah

$$S \circ R = \{(1, u), (1, t), (2, s), (2, t), (3, s), (3, t), (3, u)\}$$

Komposisi relasi R dan S lebih jelas lagi jika diperagakan dengan diagram panah (Gambar 3.3). ■



Gambar 3.3 Diagram panah yang memperlihatkan komposisi relasi R dan S pada Contoh 3.17.

Jika relasi R_1 dan R_2 masing-masing dinyatakan dengan matriks M_{R_1} dan M_{R_2} , maka matriks yang menyatakan komposisi dari kedua relasi tersebut adalah

$$M_{R_2 \circ R_1} = M_{R_1} \cdot M_{R_2}$$

yang dalam hal ini operator “.” sama seperti pada perkalian matriks biasa, tetapi dengan mengganti tanda kali dengan “ \wedge ” dan tanda tambah dengan “ \vee ”.

Contoh 3.18

Misalkan bahwa relasi R_1 dan R_2 pada himpunan A dinyatakan oleh matriks

$$R_1 = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad \text{dan} \quad R_2 = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 1 \end{bmatrix}$$

maka matriks yang menyatakan $R_2 \circ R_1$ adalah

$$\begin{aligned} M_{R_2 \circ R_1} = M_{R_1} \cdot M_{R_2} &= \begin{bmatrix} (1 \wedge 0) \vee (0 \wedge 0) \vee (1 \wedge 1) & (1 \wedge 1) \vee (0 \wedge 0) \vee (1 \wedge 0) & (1 \wedge 0) \vee (0 \wedge 1) \vee (1 \wedge 1) \\ (1 \wedge 0) \vee (1 \wedge 0) \vee (0 \wedge 1) & (1 \wedge 1) \vee (1 \wedge 0) \vee (0 \wedge 0) & (1 \wedge 0) \vee (1 \wedge 1) \vee (0 \wedge 1) \\ (0 \wedge 0) \vee (0 \wedge 0) \vee (0 \wedge 1) & (0 \wedge 1) \vee (0 \wedge 0) \vee (0 \wedge 0) & (0 \wedge 0) \vee (0 \wedge 1) \vee (0 \wedge 1) \end{bmatrix} \\ &= \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix} \end{aligned}$$

Simbol R^n digunakan untuk mendefinisikan komposisi relasi dengan dirinya sendiri sebanyak n kali, yaitu

$$R^n = R \circ R \circ \dots \circ R \quad (\text{sebanyak } n \text{ kali})$$

dan

$$M_{R^n} = M_R^{[n]}$$

Oleh karena

$$R^{n+1} = R^n \circ R$$

maka

$$M_{R^{n+1}} = M_R^{[n]} \cdot M_R$$

Contoh 3.19

Misalkan $R = \{(1, 1), (1, 3), (2, 2), (3, 1), (3, 2)\}$ adalah relasi pada himpunan $A = \{1, 2, 3\}$. Tentukan R^2 .

Penyelesaian:

Karena $R^2 = R \circ R$, maka

$$R \circ R = \{(1, 1), (1, 3), (1, 2), (2, 2), (3, 1), (3, 3), (3, 2)\}$$

Bila diselesaikan dengan menggunakan matriks, maka matriks yang merepresentasikan relasi R adalah

$$M_R = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & 0 \end{bmatrix}$$

sehingga

$$M_{R^2} = M_R^{[2]} = M_R \cdot M_R = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

Periksalah bahwa matriks terakhir ini merepresentasikan $\{(1, 1), (1, 2), (1, 3), (2, 2), (3, 1), (3, 2), (3, 3)\}$ seperti jawaban di atas.

3.7 Sifat-sifat Relasi

Relasi pada sebuah himpunan adalah kasus yang paling sering dijumpai. Relasi ini mempunyai beberapa sifat. Sifat-sifat yang paling penting didefinisikan di bawah ini.

1. Refleksif (*reflexive*)

DEFINISI 3.6. Relasi R pada himpunan A disebut refleksif jika $(a, a) \in R$ untuk setiap $a \in A$.

Dengan kata lain, Definisi 3.6 menyatakan bahwa di dalam relasi refleksif setiap elemen di dalam A berhubungan dengan dirinya sendiri. Definisi 3.6 ini juga menyatakan bahwa relasi R pada himpunan A tidak refleksif jika ada $a \in A$ sedemikian sehingga $(a, a) \notin R$. Contoh-contoh berikut dapat memperjelas sifat relasi refleksif.

Contoh 3.20

Misalkan $A = \{1, 2, 3, 4\}$, dan relasi R di bawah ini didefinisikan pada himpunan A , maka

- (a) Relasi $R = \{(1, 1), (1, 3), (2, 1), (2, 2), (3, 3), (4, 2), (4, 3), (4, 4)\}$ bersifat refleksif karena terdapat elemen relasi yang berbentuk (a, a) , yaitu $(1, 1), (2, 2), (3, 3)$, dan $(4, 4)$.
- (b) Relasi $R = \{(1, 1), (2, 2), (2, 3), (4, 2), (4, 3), (4, 4)\}$ tidak bersifat refleksif karena $(3, 3) \notin R$. ■

Contoh 3.21

Relasi “habis membagi” pada himpunan bilangan bulat positif bersifat refleksif karena setiap bilangan bulat positif selalu habis membagi dirinya sendiri, sehingga $(a, a) \in R$ untuk setiap $a \in A$. ■

Contoh 3.22

Tiga buah relasi di bawah ini menyatakan relasi pada himpunan bilangan bulat positif N .

$$R : x \text{ lebih besar dari } y, \quad S : x + y = 5, \quad T : 3x + y = 10$$

Tidak satupun dari ketiga relasi di atas yang refleksif karena, misalkan $(2, 2)$ bukan anggota R, S , maupun T . ■

Ditinjau dari representasi relasi, relasi yang bersifat refleksif mempunyai matriks yang elemen diagonal utamanya semua bernilai 1, atau $m_{ii} = 1$, untuk $i = 1, 2, \dots, n$,

$$\begin{bmatrix} 1 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \\ & & & & 1 \end{bmatrix}$$

sedangkan graf berarah dari relasi yang bersifat refleksif dicirikan adanya gelang pada setiap simpulnya.

2. Setangkup (*symmetric*) dan Tolak-setangkup (*antisymmetric*)

DEFINISI 3.7. Relasi R pada himpunan A disebut **setangkup** jika $(a, b) \in R$, maka $(b, a) \in R$, untuk semua $a, b \in A$.

Dengan kata lain, Definisi 3.7 menyatakan bahwa relasi R pada himpunan A tidak setangkup jika $(a, b) \in R$ sedemikian sehingga $(b, a) \notin R$. Sebagai contoh, misalkan A adalah himpunan mahasiswa di sebuah universitas dan R adalah relasi pada A sedemikian sehingga $(a, b) \in R$ jika dan hanya jika a satu fakultas dengan b . Jelas bahwa b juga sefakultas dengan a . Jadi, R setangkup. Contoh lain, misalkan T adalah relasi pada himpunan bilangan bulat positif sedemikian sehingga $(a, b) \in T$ jika dan hanya jika $a \geq b$. Jelas T tidak setangkup, karena, misalnya $(6, 5) \in T$ tetapi $(5, 6) \notin T$.

DEFINISI 3.8. Relasi R pada himpunan A disebut **tolak-setangkup** jika $(a, b) \in R$ dan $(b, a) \in R$ maka $a = b$, untuk semua $a, b \in A$.

Definisi 3.8 menyatakan bahwa jika $(a, b) \in R$, maka $(b, a) \notin R$ kecuali $a = b$. Definisi 3.8 juga menyatakan bahwa relasi R pada himpunan A tidak tolak-setangkup jika ada elemen berbeda a dan b sedemikian sehingga $(a, b) \in R$ dan $(b, a) \in R$. Sebagai ilustrasi, misalkan A adalah himpunan tes yang diadakan untuk masuk sebuah perusahaan (misalnya tes tertulis, tes kesehatan, tes wawancara, dsb). Misalkan R adalah relasi pada A sedemikian sehingga $(a, b) \in R$ jika tes a dilakukan sebelum tes b . Jelas, jika tes a dilakukan sebelum tes b , tes b tidak mungkin dilakukan sebelum tes a untuk dua tes a dan b yang berbeda. Dengan kata lain, $(b, a) \notin R$ kecuali $a = b$. Jadi, R adalah relasi tolak-setangkup.

Perhatikanlah bahwa istilah setangkup dan tolak-setangkup tidaklah berlawanan, karena suatu relasi dapat memiliki kedua sifat itu sekaligus. Namun, relasi tidak dapat memiliki kedua sifat tersebut sekaligus jika ia mengandung beberapa pasangan terurut berbentuk (a, b) yang mana $a \neq b$ [ROS03].

Contoh-contoh berikut dapat memperjelas sifat relasi setangkup dan tolak-setangkup.

Contoh 3.23

Misalkan $A = \{1, 2, 3, 4\}$, dan relasi R di bawah ini didefinisikan pada himpunan A , maka

- Relasi $R = \{(1, 1), (1, 2), (2, 1), (2, 2), (2, 4), (4, 2), (4, 4)\}$ bersifat setangkup karena jika $(a, b) \in R$ maka $(b, a) \in R$. Di sini $(1, 2)$ dan $(2, 1) \in R$, begitu juga $(2, 4)$ dan $(4, 2) \in R$.
- Relasi $R = \{(1, 1), (2, 3), (2, 4), (4, 2)\}$ tidak setangkup karena $(2, 3) \in R$, tetapi $(3, 2) \notin R$.
- Relasi $R = \{(1, 1), (2, 2), (3, 3)\}$ tolak-setangkup karena $(1, 1) \in R$ dan $1 = 1$, $(2, 2) \in R$ dan $2 = 2$, $(3, 3) \in R$ dan $3 = 3$. Perhatikan bahwa R juga setangkup.

- (d) Relasi $R = \{(1, 1), (1, 2), (2, 2), (2, 3)\}$ tolak-setangkup karena $(1, 1) \in R$ dan $1 = 1$ dan, $(2, 2) \in R$ dan $2 = 2$ dan. Perhatikan bahwa R tidak setangkup.
- (e) Relasi $R = \{(1, 1), (2, 4), (3, 3), (4, 2)\}$ tidak tolak-setangkup karena $2 \neq 4$ tetapi $(2, 4)$ dan $(4, 2)$ anggota R . Relasi R pada (a) dan (b) di atas juga tidak tolak-setangkup.
- (f) Relasi $R = \{(1, 2), (2, 3), (1, 3)\}$ tidak setangkup tetapi tolak-setangkup

Contoh 3.24

Relasi "habis membagi" pada himpunan bilangan bulat positif tidak setangkup karena jika a habis membagi b , b tidak habis membagi a , kecuali jika $a = b$. Sebagai contoh, 2 habis membagi 4, tetapi 4 tidak habis membagi 2. Karena itu, $(2, 4) \in R$ tetapi $(4, 2) \notin R$. Relasi "habis membagi" tolak-setangkup karena jika a habis membagi b dan b habis membagi a maka $a = b$. Sebagai contoh, 4 habis membagi 4. Karena itu, $(4, 4) \in R$ dan $4 = 4$.

Contoh 3.25

Tiga buah relasi di bawah ini menyatakan relasi pada himpunan bilangan bulat positif N .

$$R : x \text{ lebih besar dari } y, \quad S : x + y = 6, \quad T : 3x + y = 10$$

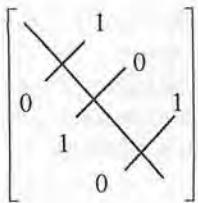
R bukan relasi setangkup karena, misalkan 5 lebih besar dari 3 tetapi 3 tidak lebih besar dari 5. S relasi setangkup karena $(4, 2)$ dan $(2, 4)$ adalah anggota S . T tidak setangkup karena, misalkan $(3, 1)$ adalah anggota T tetapi $(1, 3)$ bukan anggota T . S bukan relasi tolak-setangkup karena, misalkan $(4, 2) \in S$ dan $(4, 2) \in S$ tetapi $4 \neq 2$. Relasi R dan T keduanya tolak-setangkup (tunjukkan!).

Ditinjau dari representasi relasi, relasi yang bersifat setangkup mempunyai matriks yang elemen-elemen di bawah diagonal utama merupakan pencerminan dari elemen-elemen di atas diagonal utama, atau $m_{ij} = m_{ji} = 1$, untuk $i = 1, 2, \dots, n$. Matriks dari relasi setangkup diperlihatkan seperti di bawah ini:

$$\begin{bmatrix} & & 1 \\ & \times & \\ 1 & & \\ & \times & \\ & & 0 \end{bmatrix}$$

Sedangkan graf berarah dari relasi yang bersifat setangkup dicirikan oleh: jika ada busur dari a ke b , maka juga ada busur dari b ke a .

Matriks dari relasi tolak-setangkup mempunyai sifat yaitu jika $m_{ij} = 1$ dengan $i \neq j$, maka $m_{ji} = 0$. Dengan kata lain, matriks dari relasi tolak-setangkup adalah jika salah satu dari $m_{ij} = 0$ atau $m_{ji} = 0$ bila $i \neq j$. Matriks relasi tolak-setangkup diperlihatkan seperti berikut ini:



Sedangkan graf berarah dari relasi yang bersifat tolak-setangkup dicirikan oleh: jika dan hanya jika tidak pernah ada dua busur dalam arah berlawanan antara dua simpul berbeda.

3. Menghantar (*transitive*)

DEFINISI 3.9. Relasi R pada himpunan A disebut menghantar jika $(a, b) \in R$ dan $(b, c) \in R$, maka $(a, c) \in R$, untuk semua $a, b, c \in A$.

Sebagai ilustrasi, misalkan A adalah himpunan orang, dan R adalah relasi pada A sedemikian sehingga $(a, b) \in R$ jika dan hanya jika b adalah keturunan a . Jelas, jika b adalah keturunan a , yaitu $(a, b) \in R$, dan c adalah keturunan b , yaitu $(b, c) \in R$, maka c juga keturunan a , yaitu $(a, c) \in R$. Jadi, R adalah relasi menghantar. Tetapi, jika T adalah relasi pada A sedemikian sehingga $(a, b) \in T$ jika a adalah ibu dari b , maka T tidak menghantar.

Contoh-contoh berikut dapat memperjelas sifat relasi menghantar dan tidak menghantar.

Contoh 3.26

Misalkan $A = \{1, 2, 3, 4\}$, dan relasi R di bawah ini didefinisikan pada himpunan A , maka

- (a) $R = \{(2, 1), (3, 1), (3, 2), (4, 1), (4, 2), (4, 3)\}$ bersifat menghantar. Periksa dengan membuat tabel berikut:

Pasangan berbentuk

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

- (b) $R = \{(1, 1), (2, 3), (2, 4), (4, 2)\}$ tidak manghantar karena $(2, 4)$ dan $(4, 2) \in R$, tetapi $(2, 2) \notin R$, begitu juga $(4, 2)$ dan $(2, 3) \in R$, tetapi $(4, 3) \notin R$.

- (c) Relasi $R = \{(1, 1), (2, 2), (3, 3), (4, 4)\}$ jelas menghantar (tunjukkan!).
- (d) Relasi $R = \{(1, 2), (3, 4)\}$ menghantar karena tidak ada $(a, b) \in R$ dan $(b, c) \in R$ sedemikian sehingga $(a, c) \in R$.
- (e) Relasi yang hanya berisi satu elemen seperti $R = \{(4, 5)\}$ selalu menghantar (alasannya sama seperti jawaban (d) di atas).

Contoh 3.27

Relasi “habis membagi” pada himpunan bilangan bulat positif bersifat menghantar. Misalkan bahwa a habis membagi b dan b habis membagi c . Maka terdapat bilangan positif m dan n sedemikian sehingga $b = ma$ dan $c = nb$. Di sini $c = nma$, sehingga a habis membagi c . Jadi, relasi “habis membagi” bersifat menghantar.

Contoh 3.28

Tiga buah relasi di bawah ini menyatakan relasi pada himpunan bilangan bulat positif \mathbb{N} .

$$R : x \text{ lebih besar dari } y, \quad S : x + y = 6, \quad T : 3x + y = 10$$

R adalah relasi menghantar karena jika $x > y$ dan $y > z$ maka $x > z$. S tidak menghantar karena, misalkan $(4, 2)$ dan $(2, 4)$ adalah anggota S tetapi $(4, 4) \notin S$. Sebaliknya, $T = \{(1, 7), (2, 4), (3, 1)\}$ tidak menghantar.

Ditinjau dari representasi relasi, relasi yang bersifat menghantar tidak mempunyai ciri khusus pada matriks representasinya, tetapi sifat menghantar pada graf berarah ditunjukkan oleh: jika ada busur dari a ke b dan dari b ke c , maka juga terdapat busur berarah dari a ke c .

Suatu relasi dapat mengandung beberapa sifat sekaligus atau sama sekali tidak mengandung sifat apa pun sama sekali. Contoh 3.29 dan Contoh 3.30 di bawah ini mengilustrasikan hal ini.

Contoh 3.29

Misalkan R adalah relasi yang didefinisikan pada himpunan bilangan bulat (*integer*), yang dalam hal ini $(x, y) \in R$ jika x adalah kelipatan dari y . Tentukan apakah R refleksif, setangkup, tolak-setangkup, dan/atau menghantar dengan menyebutkan masing-masing alasannya.

Penyelesaian:

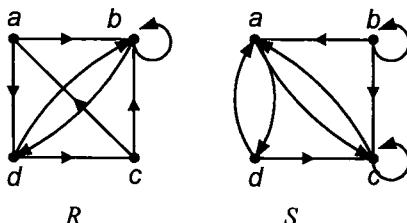
Berdasarkan pernyataan “ $(x, y) \in R$ jika x adalah kelipatan dari y ” kita dapat menuliskannya sebagai $x = ky$, yang dalam hal ini $k = 0, 1, 2, \dots$. Jadi $(x, y) = (ky, y) \in R$. R bersifat refleksif, tolak-setangkup dan menghantar, tetapi R tidak setangkup:

- (i) R refleksif sebab untuk $k = 1$, $(y, y) \in R$. Misalnya, $(4, 4), (5, 5)$, dst adalah $\in R$.
- (ii) R tidak setangkup sebab $x \geq y$ sehingga tidak mungkin y adalah kelipatan dari x kecuali jika $x = y$.

- (iii) R tolak-setangkup karena $(x, y) \in R$ dan $(y, x) \in R$ jika $x = y$.
(iv) R menghantar sebab jika $x = ky$ dan $y = mz$, maka di sini $x = kmz$, sehingga x juga kelipatan z . Contohnya, jika $(8, 4) \in R$ dan $(4, 2) \in R$, maka $(8, 2) \in R$

Contoh 3.30

Diketahui dua buah relasi, R dan S , yang masing-masing didefinisikan pada himpunan $A = \{a, b, c, d\}$. Masing-masing relasi direpresentasikan dalam graf berarah berarah berikut ini:



Tentukan apakah R dan S refleksif, setangkup, tolak-setangkup, dan/atau menghantar. Jelaskan alasannya.

Penyelesaian:

Relasi R :

- (i) R tidak refleksif sebab pada simpul a , c , dan d tidak ada busur *loop*
- (ii) R tidak setangkup sebab dari simpul a ke d ada busur berarah, sedangkan dari d ke a tidak ada. Begitu juga dari d ke c , dari c ke b , dan dari a ke b .
- (iii) R tidak menghantar sebab ada busur dari b ke d dan busur dari d ke c tetapi tidak ada busur dari b ke c
- (iv) R tidak tolak-setangkup sebab ada busur dari b ke d begitu juga sebaliknya dari d ke b , padahal $b \neq d$.

Relasi S :

- (i) S tidak refleksif sebab pada simpul a dan d tidak ada sisi *loop*
- (ii) S tidak setangkup sebab dari simpul b ke a ada sisi berarah, sedangkan dari a ke b tidak ada. Begitu juga dari b ke c dan dari d ke c .
- (iii) S tidak menghantar sebab ada sisi dari c ke a dan sisi dari a ke d tetapi tidak ada sisi dari c ke d .
- (iv) S tidak tolak-setangkup sebab ada busur dari a ke d begitu juga sebaliknya dari d ke a , padahal $a \neq d$. Begitu juga untuk busur (a, c) dan (c, a) .

3.8 Relasi Kesetaraan

Contoh 3.29 memperlihatkan bahwa sebuah relasi dapat memiliki beberapa sifat sekaligus. Jika sebuah relasi mempunyai sifat setangkup, refleksif, dan menghantar sekaligus, maka relasi tersebut dinamakan relasi kesetaraan (*equivalence relation*).

DEFINISI 3.10. Relasi R pada himpunan A disebut **relasi kesetaraan (equivalence relation)** jika ia refleksif, setangkup dan menghantar.

Secara intuitif, di dalam relasi kesetaraan, dua benda berhubungan jika keduanya memiliki beberapa sifat yang sama atau memenuhi beberapa persyaratan yang sama [LIU85].

Dua elemen yang dihubungkan dengan relasi kesetaraan dinamakan **setara (equivalent)**. Berdasarkan sifat yang dimilikinya, kesetaraan tersebut dijelaskan sebagai berikut: Karena relasi bersifat setangkup, maka dua elemen yang dihubungkan relasi adalah setara. Karena relasi bersifat refleksif, maka setiap elemen setara dengan dirinya sendiri. Karena relasi bersifat menghantar, maka jika a dan b setara dan b dan c setara, maka a dan c setara.

Sebagai contoh, misalkan R adalah relasi pada himpunan mahasiswa sedemikian sehingga $(a, b) \in R$ jika a satu angkatan dengan b . Karena setiap mahasiswa seangkatan dengan dirinya sendiri, maka R jelas refleksif. Perhatikan, jika a seangkatan dengan b , maka b pasti seangkatan dengan a . Jadi, R setangkup. Selanjutnya, jika a seangkatan dengan b dan b seangkatan dengan c , maka pastilah a seangkatan dengan c . Jelas, R bersifat menghantar. Dengan demikian, R adalah relasi kesetaraan.

3.9 Relasi Pengurutan Parsial

Relasi sering digunakan untuk mengurutkan elemen-elemen di dalam himpunan. Misalnya, elemen-elemen di dalam himpunan bilangan bulat terurut oleh relasi \leq (lebih besar atau sama dengan). Karena elemen-elemen tersebut terurut berdasarkan relasi \leq , maka jika diberikan sebuah bilangan bulat, kita dapat menentukan bilangan bulat berikutnya (*successor*) atau bilangan bulat sebelumnya (*predecessor*). Definisi keterurutan pada relasi dinyatakan oleh definisi berikut:

DEFINISI 3.11. Relasi R pada himpunan S dikatakan **relasi pengurutan parsial (partial ordering relation)** jika ia refleksif, tolak-setangkup, dan menghantar. Himpunan S bersama-sama dengan relasi R disebut **himpunan terurut secara parsial (partially ordered set, atau poset)**, dan dilambangkan dengan (S, R) .

Relasi \geq pada himpunan bilangan bulat adalah relasi pengurutan parsial. Hal ini dijelaskan sebagai berikut: Karena $a \geq a$ untuk setiap bilangan bulat a , relasi \geq refleksif. Relasi \geq tolak-setangkup, karena jika $a \geq b$ dan $b \geq a$, maka $a = b$. Relasi \geq menghantar, karena jika $a \geq b$ dan $b \geq c$ maka $a \geq c$.

Relasi “habis membagi” pada himpunan bilangan bulat juga adalah relasi pengurutan parsial. Hal ini dijelaskan sebagai berikut: Karena setiap bilangan bulat habis membagi dirinya sendiri, relasi “habis membagi” bersifat refleksif. Relasi “habis membagi” bersifat tolak-setangkup, karena a habis membagi b berarti b tidak habis membagi a , kecuali jika $a = b$. Terakhir, relasi ini mengantar karena jika a habis membagi b , dan b habis membagi c , maka a habis membagi c . Tinjau kembali Contoh 3.21, Contoh 3.24, dan Contoh 3.27 yang menjelaskan sifat pada relasi “habis membagi” ini.

Secara intuitif, di dalam relasi pengurutan parsial, istilah pengurutan berarti dua buah benda saling berhubungan jika salah satunya lebih kecil (lebih besar) daripada, atau lebih rendah (lebih tinggi) daripada lainnya menurut sifat atau kriteria tertentu. Istilah pengurutan itu sendiri memang menyatakan bahwa benda-benda di dalam himpunan tersebut dirutkan berdasarkan sifat atau kriteria tersebut. Akan tetapi, ada juga kemungkinan dua buah benda di dalam himpunan tidak berhubungan dalam suatu relasi pengurutan parsial. Dalam hal demikian, kita tidak dapat membandingkan keduanya sehingga tidak dapat diidentifikasi mana yang lebih besar atau lebih kecil. Itulah alasan digunakan istilah pengurutan parsial atau pengurutan tak-lengkap [LIU85].

3.10 Klosur Relasi

Misalkan R adalah relasi yang tidak refleksif. Kita dapat membuat relasi baru yang mengandung R sedemikian sehingga relasi baru tersebut menjadi refleksif. Relasi baru tersebut haruslah relasi terkecil yang mengandung R . Sebagai contoh, relasi $R = \{(1, 1), (1, 3), (2, 3), (3, 2)\}$ pada himpunan $A = \{1, 2, 3\}$ tidak refleksif. Bagaimana membuat relasi refleksif yang sesedikit mungkin dan mengandung R ? Untuk melakukan hal ini, kita hanya perlu menambahkan $(2, 2)$ dan $(3, 3)$ ke dalam R karena dua elemen relasi ini yang belum terdapat di dalam R . Relasi baru yang terbentuk, dilambangkan dengan S , mengandung R , yaitu

$$S = \{(1, 1), (1, 3), (2, 2), (2, 3), (3, 2), (3, 3)\}$$

Sekarang, S bersifat refleksif. Relasi S disebut **klosur refleksif** (*reflexive closure*) dari R . Sembarang relasi lain yang mengandung R harus juga memuat $(2, 2)$ dan $(3, 3)$. Catat juga bahwa S adalah himpunan bagian dari sembarang relasi refleksif lain yang memuat R .

Contoh lain, misalkan $R = \{(1, 3), (1, 2), (2, 1), (3, 2), (3, 3)\}$ pada himpunan $A = \{1, 2, 3\}$. Jelas, R tidak setangkup. Bagaimana membuat relasi setangkup yang sesedikit mungkin dan mengandung R ? Untuk melakukan hal ini, kita hanya perlu menambahkan $(3, 1)$ dan $(2, 3)$ ke dalam R karena dua elemen relasi ini yang

belum terdapat di dalam S agar S menjadi setangkup. Relasi baru yang terbentuk mengandung R yaitu

$$S = \{(1, 3), (3, 1), (1, 2), (2, 1), (3, 2), (2, 3), (3, 3)\}$$

Sekarang, S bersifat setangkup. Relasi S disebut **klosur setangkup (symmetric closure)** dari R .

Secara umum, misalkan R adalah relasi pada himpunan A . R dapat memiliki atau tidak memiliki sifat P , seperti refleksif, setangkup, atau menghantar. Jika terdapat relasi S dengan sifat P yang mengandung R sedemikian sehingga S adalah himpunan bagian dari setiap relasi dengan sifat P yang mengandung R , maka S disebut **klosur (closure)** atau tutupan dari R [ROS03].

Ada tiga jenis klosursi, yaitu klosur refleksif (*reflexive closure*), klosur setangkup (*symmetric closure*), dan klosur menghantar (*transitive closure*). Cara-cara membentuk ketiga klosur tersebut dijelaskan di bawah ini.

Klosur Refleksif

Misalkan R adalah sebuah relasi pada himpunan A . Klosur refleksif dari R adalah $R \cup \Delta$, yang dalam hal ini $\Delta = \{(a, a) \mid a \in A\}$. Pada contoh kita di atas, $R = \{(1, 1), (1, 3), (2, 3), (3, 2)\}$ adalah relasi pada himpunan $A = \{1, 2, 3\}$, maka $\Delta = \{(1, 1), (2, 2), (3, 3)\}$, sehingga klosur refleksif dari R adalah

$$\begin{aligned}R \cup \Delta &= \{(1, 1), (1, 3), (2, 3), (3, 2)\} \cup \{(1, 1), (2, 2), (3, 3)\} \\&= \{(1, 1), (1, 3), (2, 2), (2, 3), (3, 2), (3, 3)\}\end{aligned}$$

Contoh 3.31

Misalkan R adalah relasi $\{(a, b) \mid a \neq b\}$ pada himpunan bilangan bulat. Maka, klosur refleksif dari R adalah

$$R \cup \Delta = \{(a, b) \mid a \neq b\} \cup \{(a, a) \mid a \in \mathbf{Z}\} = \{(a, b) \mid a, b \in \mathbf{Z}\}$$

Klosur Setangkup

Misalkan R adalah sebuah relasi pada himpunan A . Klosur setangkup dari R adalah $R \cup R^{-1}$, yang dalam hal ini $R^{-1} = \{(b, a) \mid (a, b) \in R\}$. Pada contoh kita di atas, $R = \{(1, 3), (1, 2), (2, 1), (3, 2), (3, 3)\}$ adalah relasi pada himpunan $A = \{1, 2, 3\}$, maka $R^{-1} = \{(3, 1), (2, 1), (1, 2), (2, 3), (3, 3)\}$ sehingga klosur setangkup dari R adalah

$$\begin{aligned}R \cup R^{-1} &= \{(1, 3), (1, 2), (2, 1), (3, 2), (3, 3)\} \cup \{(3, 1), (2, 1), (1, 2), (2, 3), (3, 3)\} \\&= \{(1, 3), (3, 1), (1, 2), (2, 1), (3, 2), (2, 3), (3, 3)\}\end{aligned}$$

Contoh 3.32

Misalkan R adalah relasi $\{(a, b) \mid a \text{ habis membagi } b\}$ pada himpunan bilangan bulat. Maka, klosur setangkup dari R adalah

$$\begin{aligned}R \cup R^{-1} &= \{(a, b) \mid a \text{ habis membagi } b\} \cup \{(b, a) \mid b \text{ habis membagi } a\} \\&= \{(a, b) \mid a \text{ habis membagi } b \text{ atau } b \text{ habis membagi } a\}\end{aligned}$$

Klosur Menghantar

Pembentukan klosur menghantar lebih sulit daripada dua buah klosur sebelumnya. Sebagai contoh, misalkan $R = \{(1, 2), (1, 4), (2, 1), (3, 2)\}$ adalah relasi pada himpunan $A = \{1, 2, 3\}$. Relasi ini tidak transitif karena tidak mengandung semua pasangan (a, c) sedemikian sehingga (a, b) dan (b, c) di dalam R . Pasangan (a, c) yang tidak terdapat di dalam R adalah $(1, 1), (2, 2), (2, 4)$, dan $(3, 1)$. Penambahan semua pasangan ini ke dalam R sehingga menjadi

$$S = \{(1, 2), (1, 4), (2, 1), (3, 2), (1, 1), (2, 2), (2, 4), (3, 1)\}$$

tidak menghasilkan relasi yang bersifat menghantar karena, misalnya terdapat $(3, 1) \in S$ dan $(1, 4) \in S$, tetapi $(3, 4) \notin S$.

Menemukan klosur menghantar dari sebuah relasi ekivalen dengan menentukan pasangan-pasangan simpul simpul mana di dalam graf berarah yang terhubung dengan sebuah lintasan. Teorema berikut berguna untuk menemukan klosur menghantar [ROS03].

TEOREMA 3.1. Klosur menghantar dari relasi R adalah

$$R^* = \bigcup_{n=1}^{\infty} R^n$$

Ingatlah bahwa $R^n = R \circ R \circ \dots \circ R$. Bukti untuk teorema ini tidak ditulis di sini dan dapat ditemukan di dalam [ROS03].

LEMMA 3.1. Misalkan A adalah himpunan dengan n elemen, dan R adalah relasi pada himpunan A . Jika terdapat lintasan dengan panjang minimal 1 di dalam R dari a ke b , maka lintasan semacam itu panjangnya tidak melebihi n .

Dari *lemma* ini kita dapat menunjukkan bahwa klosur menghantar dari R adalah

$$R^* = R \cup R^2 \cup R^3 \cup \dots \cup R^n$$

karena terdapat lintasan di dalam R^* antara dua simpul jika dan hanya jika terdapat lintasan antara simpul-simpul ini di dalam R^i , untuk i positif dengan $i \leq n$.

Jika M_R adalah matriks yang merepresentasikan relasi R pada sebuah himpunan dengan n elemen, maka matriks klosur menghantar R^* adalah

$$M_{R^*} = M_R \vee M_R^{[2]} \vee M_R^{[3]} \vee \dots \vee M_R^{[n]}$$

Contoh 3.33

Misalkan $R = \{(1, 1), (1, 3), (2, 2), (3, 1), (3, 2)\}$ adalah relasi pada himpunan $A = \{1, 2, 3\}$. Tentukan klosur menghantar dari R .

Penyelesaian:

Matriks yang merepresentasikan relasi R adalah

$$M_R = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & 0 \end{bmatrix}$$

Maka, matriks klosur menghantar dari R adalah

$$M_{R^*} = M_R \vee M_R^{[2]} \vee M_R^{[3]}$$

Karena

$$M_R^{[2]} = M_R \cdot M_R = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} \text{ dan } M_R^{[3]} = M_R^{[2]} \cdot M_R = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

maka

$$M_{R^*} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} \vee \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} \vee \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

Dengan demikian, $R^* = \{(1, 1), (1, 2), (1, 3), (2, 2), (3, 1), (3, 2), (3, 3)\}$

Klosur menghantar mempunyai banyak aplikasi, salah satunya di bidang komunikasi data. Klosur menghantar menggambarkan bagaimana pesan dapat dikirim dari satu kota ke kota lain baik melalui hubungan komunikasi langsung atau melalui kota antara sebanyak mungkin [LIU85]. Misalkan jaringan komputer mempunyai pusat data di Jakarta, Bandung, Surabaya, Medan, Makassar, dan Kupang. Misalkan R adalah relasi yang mengandung (a, b) jika terdapat saluran telepon dari kota a ke kota b . Gambar 3.4 memperlihatkan graf berarah yang merepresentasikan relasi R ini. Karena tidak semua *link* langsung dari satu kota ke kota lain, maka pengiriman data dari Jakarta ke Surabaya tidak dapat

dilakukan secara langsung. Relasi R tidak menghantar karena ia tidak mengandung semua pasangan pusat data yang dapat dihubungkan (baik *link* langsung atau tidak langsung). Klosur menghantar adalah relasi yang paling minimal yang berisi semua pasangan pusat data yang mempunyai link langsung atau tidak langsung dan mengandung R .



Gambar 3.4 Graf berarah yang merepresentasikan relasi saluran telepon antar pusat data.

3.11 Relasi n -ary

Relasi biner hanya menghubungkan antara dua buah himpunan. Relasi yang lebih umum menghubungkan lebih dari dua buah himpunan. Relasi tersebut dinamakan relasi n -ary (baca: ener). Jika $n = 2$, maka relasinya dinamakan relasi biner ($bi = 2$). Relasi n -ary mempunyai terapan penting di dalam basisdata.

DEFINISI 3.12 Misalkan A_1, A_2, \dots, A_n adalah himpunan. Relasi n -ary R pada himpunan-himpunan tersebut adalah himpunan bagian dari $A_1 \times A_2 \times \dots \times A_n$, atau dengan notasi $R \subseteq A_1 \times A_2 \times \dots \times A_n$. Himpunan A_1, A_2, \dots, A_n disebut daerah asal (*domain*) relasi dan n disebut *derajat*.

Contoh 3.34

Misalkan

$$NIM = \{13598011, 13598014, 13598015, 13598019, 13598021, 13598025\}$$

$$Nama = \{\text{Amir, Santi, Irwan, Ahmad, Cecep, Hamdan}\}$$

$$MatKul = \{\text{Matematika Diskrit, Algoritma, Struktur Data, Arsitektur Komputer}\}$$

$$Nilai = \{A, B, C, D, E\}$$

berturut-turut adalah himpunan Nomor Induk Mahasiswa, himpunan nama-nama mahasiswa, himpunan nama-nama mata kuliah, dan himpunan nilai mata kuliah. Relasi MHS yang terdiri dari 5-tupel ($NIM, Nama, MatKul, Nilai$) merepresentasikan hubungan antara nomor induk mahasiswa, namanya, mata kuliah yang diambilnya, dan nilai mata kuliah,

$$MHS \subseteq NIM \times Nama \times MatKul \times Nilai$$

Satu contoh relasi yang bernama *MHS* adalah

MHS = {(13598011, Amir, Matematika Diskrit, A), (13598011, Amir, Arsitektur Komputer, B),
(13598014, Santi, Arsitektur Komputer, D), (13598015, Irwan, Algoritma, C),
(13598015, Irwan, Struktur Data C), (13598015, Irwan, Arsitektur Komputer, B),
(13598019, Ahmad, Algoritma, E), (13598021, Cecep, Algoritma, A),
(13598021, Cecep, Arsitektur Komputer, B), (13598025, Hamdan, Matematika Diskrit, B),
(13598025, Hamdan, Algoritma, A, B), (13598025, Hamdan, Struktur Data, C),
(13598025, Hamdan, Ars. Komputer, B)}
}

Relasi *MHS* di atas juga dapat ditulis dalam bentuk Tabel 3.4.

Tabel 3.4

<i>NIM</i>	<i>Nama</i>	<i>MatKul</i>	<i>Nilai</i>
13598011	Amir	Matematika Diskrit	A
13598011	Amir	Arsitektur Komputer	B
13598014	Santi	Algoritma	D
13598015	Irwan	Algoritma	C
13598015	Irwan	Struktur Data	C
13598015	Irwan	Arsitektur Komputer	B
13598019	Ahmad	Algoritma	E
13598021	Cecep	Algoritma	B
13598021	Cecep	Arsitektur Komputer	B
13598025	Hamdan	Matematika Diskrit	B
13598025	Hamdan	Algoritma	A
13598025	Hamdan	Struktur Data	C
13598025	Hamdan	Arsitektur Komputer	B

Basisdata (*database*) adalah kumpulan tabel. Salah satu model basisdata adalah **model basisdata relasional** (*relational database*). Model basisdata ini didasarkan pada konsep relasi *n-ary*. Pada basisdata relasional, satu tabel menyatakan satu relasi. Setiap kolom pada tabel disebut **atribut**. Daerah asal dari atribut adalah himpunan tempat semua anggota atribut tersebut berada. Setiap tabel pada basisdata diimplementasikan secara fisik sebagai sebuah *file*. Satu baris data pada tabel menyatakan sebuah *record*, dan setiap atribut menyatakan sebuah *field*. Dengan kata lain, secara fisik basisdata adalah kumpulan *file*, sedangkan *file* adalah kumpulan *record*, setiap *record* terdiri atas sejumlah *field*.

Teori basisdata didasarkan pada konsep relasi *n-ary*, karena setiap tabel dapat mengandung lebih dari dua buah atribut (kolom). Meskipun secara fisik sebuah tabel dapat hanya mengandung sebuah atribut, namun secara teori hal ini bertentangan dengan konsep relasi *n-ary*.

Atribut khusus pada tabel yang mengidentifikasi secara unik elemen relasi disebut **kunci primer** (*primary key*). Atribut kunci membedakan suatu baris tabel dengan baris tabel lainnya. Pada Contoh 3.34 di atas, *NIM* merupakan atribut kunci primer, karena setiap mahasiswa dibedakan dari *NIM*-nya (tidak ada dua mahasiswa yang mempunyai *NIM* sama). Atribut *Nama* tidak dapat menjadi kunci primer karena nama yang sama mungkin saja dimiliki oleh beberapa orang.

Operasi yang dilakukan terhadap basisdata dilakukan dengan perintah pertanyaan yang disebut *query*. Satu contoh *query* misalnya,

“Tampilkan semua mahasiswa yang mengambil mata kuliah Matematika Diskrit”
“Tampilkan daftar nilai mahasiswa dengan NIM = 13598015”

“Tampilkan daftar mahasiswa yang terdiri atas NIM dan mata kuliah yang diambil”

Pada hakekatnya, *query* terhadap basisdata relasional dapat dinyatakan secara abstrak dengan operasi pada relasi *n-ary*. Ada beberapa operasi yang dapat digunakan, diantaranya adalah seleksi, proyeksi, dan *join*.

Seleksi

Operasi seleksi memilih baris tertentu dari suatu tabel yang memenuhi persyaratan tertentu.

Operator: σ

Contoh 3.35

Misalkan untuk relasi *MHS* kita ingin menampilkan daftar mahasiswa yang mengambil mata kuliah Matematika Diskrit. Operasi seleksinya adalah

$$\sigma_{Matkul="Matematika Diskrit"}(MHS)$$

yang menghasilkan tupel (13598011, Amir, Matematika Diskrit, A) dan (13598025, Hamdan, Matematika Diskrit, B) ■

Proyeksi

Operasi proyeksi memilih kolom tertentu dari suatu tabel. Jika ada beberapa baris yang sama nilainya, maka hanya diambil satu kali.

Operator: π

Contoh 3.36

Operasi proyeksi

$$\pi_{Nama, MatKul, Nilai}(MHS)$$

menghasilkan Tabel 3.5. Sedangkan operasi proyeksi

$$\pi_{NIM, Nama}(MHS)$$

menghasilkan Tabel 3.6. ■

Tabel 3.5

Nama	MatKul	Nilai
Amir	Matematika Diskrit	A
Amir	Arsitektur Komputer	B
Santi	Algoritma	D
Irwan	Algoritma	C
Irwan	Struktur Data	C
Irwan	Arsitektur Komputer	B
Ahmad	Algoritma	E
Cecep	Algoritma	B
Cecep	Arsitektur Komputer	B
Hamdan	Matematika Diskrit	B
Hamdan	Algoritma	A
Hamdan	Struktur Data	C
Hamdan	Arsitektur Komputer	B

Tabel 3.6

NIM	Nama
13598011	Amir
13598014	Santi
13598015	Irwan
13598019	Ahmad
13598021	Cecep
13598025	Hamdan

Join

Operasi *join* menggabungkan dua buah tabel menjadi satu bila kedua tabel mempunyai atribut yang sama. Sebagai contoh, suatu tabel mengandung *NIM*, *Nama*, *JenisKelamin* (*JK*), dan tabel lain mengandung *NIM*, *Nama*, *MatKul*, *Nilai*. Gabungan keduanya menghasilkan tabel baru yang mengandung atribut *NIM*, *Nama*, *JenisKelamin*, *MatKul*, dan *Nilai*.

Operator: τ

Contoh 3.37

Misalkan relasi *MHS1* dinyatakan dengan Tabel 3.7 dan relasi *MHS2* dinyatakan dengan Tabel 3.8. Operasi *join*

$$\tau_{NIM, Nama}(MHS1, MHS2)$$

menghasilkan Tabel 3.9. ■

Tabel 3.7

NIM	Nama	JK
13598001	Hananto	L
13598002	Guntur	L
13598004	Heidi	W
13598006	Harman	L
13598007	Karim	L

Tabel 3.8

NIM	Nama	MatKul	Nilai
13598001	Hananto	Algoritma	A
13598001	Hananto	Basisdata	B
13598004	Heidi	Kalkulus I	B
13598006	Harman	Teori Bahasa	C
13598006	Harman	Agama	A
13598009	Junaidi	Statistik	B
13598010	Farizka	Otomata	C

Tabel 3.9

NIM	Nama	JK	MatKul	Nilai
13598001	Hananto	L	Algoritma	A
13598001	Hananto	L	Basisdata	B
13598004	Heidi	W	Kalkulus I	B
13598006	Harman	L	Teori Bahasa	C
13598006	Harman	L	Agama	A

SQL

Bahasa khusus untuk *query* di dalam basisdata disebut *SQL (Structured Query Language)*. Bahasa ini dirancang sedemikian sehingga dapat merealisasikan *query-query* abstrak yang sudah dijelaskan di atas. Misalnya,

```
SELECT NIM, Nama, MatKul, Nilai
FROM MHS
WHERE MatKul = 'Matematika Diskrit'
```

adalah bahasa *SQL* yang bersesuaian untuk *query* abstrak

```
 $\sigma_{Matkul='Matematika Diskrit'} (MHS)$ 
```

yang menghasilkan tupel (13598011, Amir, Matematika Diskrit, A) dan (13598025, Hamdan, Matematika Diskrit, B).

3.12 Fungsi

Berapa lama waktu yang dibutuhkan komputer untuk mengeksekusi sebuah program? Jawabannya bergantung pada ukuran masukan yang diberikan. Kalau program kita mengalikan matriks yang berukuran 10×10 tentu kebutuhan waktunya berbeda jika matriksnya berukuran 100×100 . Ini berarti ada hubungan antara ukuran masukan dengan kebutuhan waktu program. Dengan kata lain, kebutuhan waktu sebuah program adalah *fungsi* dari ukuran masukan.

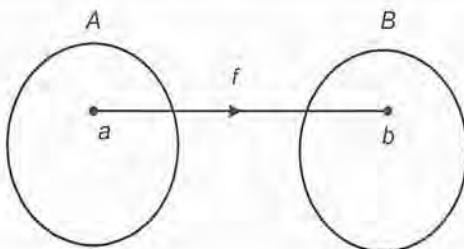
Konsep fungsi sangat penting di dalam matematika diskrit. Fungsi sering dipakai untuk mentransformasikan elemen di sebuah himpunan dengan elemen di himpunan lain. Definisi fungsi yang kita gunakan di dalam buku ini adalah:

DEFINISI 3.13. Misalkan A dan B himpunan. Relasi biner f dari A ke B merupakan suatu fungsi jika setiap elemen di dalam A dihubungkan dengan tepat satu elemen di dalam B . Jika f adalah fungsi dari A ke B kita menuliskan

$$f: A \rightarrow B$$

yang artinya f memetakan A ke B .

Nama lain untuk fungsi adalah **pemetaan** atau **transformasi**. Kita menuliskan $f(a) = b$ jika elemen a di dalam A dihubungkan dengan elemen b di dalam B . Himpunan A disebut **daerah asal (domain)** dari f dan himpunan B disebut **daerah hasil (codomain)** dari f . Jika $f(a) = b$, maka b dinamakan **bayangan (image)** dari a dan a dinamakan **pra-bayangan (pre-image)** dari b . Himpunan yang berisi semua nilai pemetaan f disebut **jelajah (range)** dari f . Perhatikan bahwa jelajah dari f adalah himpunan bagian (mungkin *proper subset*) dari B . Gambar 3.5 merepresentasikan fungsi dari A ke B .



Gambar 3.5 Fungsi f memetakan A ke B

Fungsi adalah relasi yang khusus. Kekhususan ini tercakup pada dua hal penting [DOE85]:

1. Tiap elemen di dalam himpunan A , yang merupakan daerah asal f , harus digunakan oleh prosedur atau kaidah yang mendefinisikan f .
2. Frasa "dihubungkan dengan tepat satu elemen di dalam B " berarti bahwa jika $(a, b) \in f$ dan $(a, c) \in f$, maka $b = c$.

Fungsi dapat dispesifikasi dalam berbagai bentuk, diantaranya:

1. Himpunan pasangan terurut.

Inginlah bahwa fungsi adalah relasi, sedangkan relasi biasanya dinyatakan sebagai himpunan pasangan terurut.

2. Formula pengisian nilai (*assignment*).

Di dalam kuliah aljabar atau kalkulus, fungsi dispesifikasikan dalam bentuk rumus pengisian nilai (*assignment*), misalnya $f(x) = 2x + 10$, $f(x) = x^2$, dan $f(x) = 1/x$. Jika himpunan daerah asal maupun daerah hasil fungsi tidak dinyatakan secara spesifik, maka diasumsikan daerah asal fungsi adalah **R** dan daerah hasilnya juga **R**. Dalam himpunan pasangan terurut kita mendefinisikan fungsi sebagai $f = \{(x, x_2) | x \in R\}$.

3. Kata-kata

Fungsi dapat dinyatakan secara eksplisit dalam rangkaian kata-kata. Misalnya “*f* adalah fungsi yang memetakan jumlah bit 1 di dalam suatu *string* biner”.

4. Kode program (*source code*)

Fungsi dispesifikasikan dalam bentuk kode program komputer. Misalnya dalam Bahasa Pascal, fungsi yang mengembalikan nilai mutlak dari sebuah bilangan bulat x , yaitu $|x|$, ditulis sebagai berikut:

```
function abs(x : integer) : integer;
begin
  if x < 0 then
    abs := -x
  else
    abs := x;
end;
```

Daerah asal dari fungsi *abs* di atas secara jelas dinyatakan himpunan bilangan bulat (**integer**), dan daerah hasilnya juga himpunan bilangan bulat.

Contoh 3.37

Relasi $f = \{(1, u), (2, v), (3, w)\}$ dari $A = \{1, 2, 3\}$ ke $B = \{u, v, w\}$ adalah fungsi dari A ke B . Di sini $f(1) = u$, $f(2) = v$, dan $f(3) = w$. Daerah asal dari f adalah A dan daerah hasil adalah B . Jelajah dari f adalah $\{u, v, w\}$, yang dalam hal ini sama dengan himpunan B .

Contoh 3.38

Relasi $f = \{(1, u), (2, u), (3, v)\}$ dari $A = \{1, 2, 3\}$ ke $B = \{u, v, w\}$ adalah fungsi dari A ke B , meskipun u merupakan bayangan dari dua elemen A . Daerah asal fungsi adalah A , daerah hasilnya adalah B , dan jelajah fungsi adalah $\{u, v\}$.

Contoh 3.39

Relasi $f = \{(1, u), (2, v), (3, w)\}$ dari $A = \{1, 2, 3, 4\}$ ke $B = \{u, v, w\}$ bukan fungsi, karena tidak semua elemen A dipetakan ke B .

Contoh 3.40

Relasi $f = \{(1, u), (1, v), (2, v), (3, w)\}$ dari $A = \{1, 2, 3\}$ ke $B = \{u, v, w\}$ bukan fungsi, karena 1 dipetakan ke dua buah elemen B , yaitu u dan v (ingat Definisi 3.13). ■

Contoh 3.41

Misalkan $f: \mathbb{Z} \rightarrow \mathbb{Z}$ didefinisikan oleh $f(x) = x^2$. Daerah asal dan daerah hasil dari f adalah himpunan bilangan bulat, dan jelajah dari f adalah himpunan bilangan bulat tidak-negatif (karena kuadrat dari sembarang bilangan bulat tidak mungkin negatif). ■

Contoh 3.42

Misalkan A adalah himpunan mahasiswa di ITB. Manakah dari pemetaan berikut yang mendefinisikan sebuah fungsi pada himpunan A ?

- (i) Setiap mahasiswa memetakan NIM (Nomor Induk Mahasiswa).
- (ii) Setiap mahasiswa memetakan nomor *handphone*-nya.
- (iii) Setiap mahasiswa memetakan dosen waliya.
- (iv) Setiap mahasiswa memetakan anaknya.

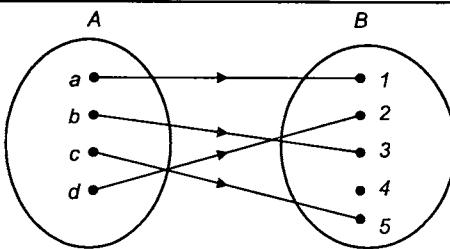
Penyelesaian:

- (i) Ya, karena setiap mahasiswa hanya mempunyai satu buah NIM.
- (ii) Tidak, karena ada mahasiswa yang mempunyai lebih dari satu nomor HP atau tidak mempunyai HP sama sekali.
- (iii) Ya, karena setiap mahasiswa hanya mempunyai 1 orang dosen wali.
- (iv) Tidak, jika ada mahasiswa yang belum menikah.

Bergantung pada bayangan, fungsi dibedakan menjadi fungsi satu-ke-satu (*one-to-one*), fungsi pada (*onto*), atau bukan salah satu dari keduanya. Kita tinjau definisi jenis setiap fungsi tersebut berikut ini.

DEFINISI 3.14. Fungsi f dikatakan **satu-ke-satu** (*one-to-one*) atau **injektif** (*injective*) jika tidak ada dua elemen himpunan A yang memiliki bayangan sama. Dengan kata lain, jika a dan b adalah anggota himpunan A , maka $f(a) \neq f(b)$ bilamana $a \neq b$. Jika $f(a) = f(b)$ maka implikasinya adalah $a = b$.

Gambar 3.6 mengilustrasikan fungsi satu-ke-satu.



Gambar 3.6 Fungsi satu-ke-satu

Contoh 3.43

Relasi $f = \{(1, w), (2, u), (3, v)\}$ dari $A = \{1, 2, 3\}$ ke $B = \{u, v, w\}$ adalah fungsi satu-ke-satu. Relasi $f = \{(1, w), (2, u), (3, v)\}$ dari $A = \{1, 2, 3\}$ ke $B = \{u, v, w\}$ juga fungsi satu-ke-satu, tetapi relasi $f = \{(1, u), (2, u), (3, v)\}$ dari $A = \{1, 2, 3\}$ ke $B = \{u, v, w\}$ bukan fungsi satu-ke-satu, karena $f(1) = f(2) = u$. ■

Contoh 3.44

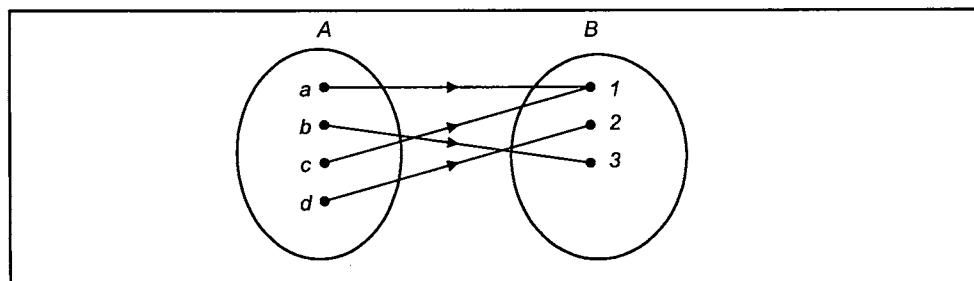
Misalkan $f : \mathbb{Z} \rightarrow \mathbb{Z}$. Tentukan apakah $f(x) = x^2 + 1$ dan $f(x) = x - 1$ merupakan fungsi satu-ke-satu?

Penyelesaian:

- (i) $f(x) = x^2 + 1$ bukan fungsi satu-ke-satu, karena untuk dua x yang bernilai mutlak sama tetapi tandanya berbeda nilai fungsinya sama, misalnya $f(2) = f(-2) = 5$ padahal $-2 \neq 2$.
- (ii) $f(x) = x - 1$ adalah fungsi satu-ke-satu karena untuk $a \neq b$, $a - 1 \neq b - 1$. Misalnya untuk $x = 2$, $f(2) = 1$ dan untuk $x = -2$, $f(-2) = -3$. ■

DEFINISI 3.14. Fungsi f dikatakan 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 . Fungsi f disebut fungsi pada himpunan B .

Gambar 3.7 mengilustrasikan fungsi pada.



Gambar 3.7 Fungsi pada

Contoh 3.45

Relasi $f = \{(1, u), (2, u), (3, v)\}$ dari $A = \{1, 2, 3\}$ ke $B = \{u, v, w\}$ bukan fungsi pada karena w tidak termasuk jelajah dari f . Relasi $f = \{(1, w), (2, u), (3, v)\}$ dari $A = \{1, 2, 3\}$ ke $B = \{u, v, w\}$ merupakan fungsi pada karena semua anggota B merupakan jelajah dari f . ■

Contoh 3.46

Misalkan $f : \mathbb{Z} \rightarrow \mathbb{Z}$. Tentukan apakah $f(x) = x^2 + 1$ dan $f(x) = x - 1$ merupakan fungsi pada?

Penyelesaian:

- (i) $f(x) = x^2 + 1$ bukan fungsi pada, karena tidak semua nilai bilangan bulat merupakan jelajah dari f . Misalnya tidak ada nilai x yang membuat nilai fungsi sama dengan 0, yaitu $x^2 + 1 = 0$ tidak dipenuhi untuk nilai x berapapun.
- (ii) $f(x) = x - 1$ adalah fungsi pada karena untuk setiap bilangan bulat y , selalu ada nilai x yang memenuhi, yaitu $y = x - 1$ akan dipenuhi untuk $x = y + 1$.

DEFINISI 3.14. Fungsi f dikatakan **berkoresponden satu-ke-satu** atau **bijeksi** (*bijection*) jika ia fungsi satu-ke-satu dan juga fungsi pada.

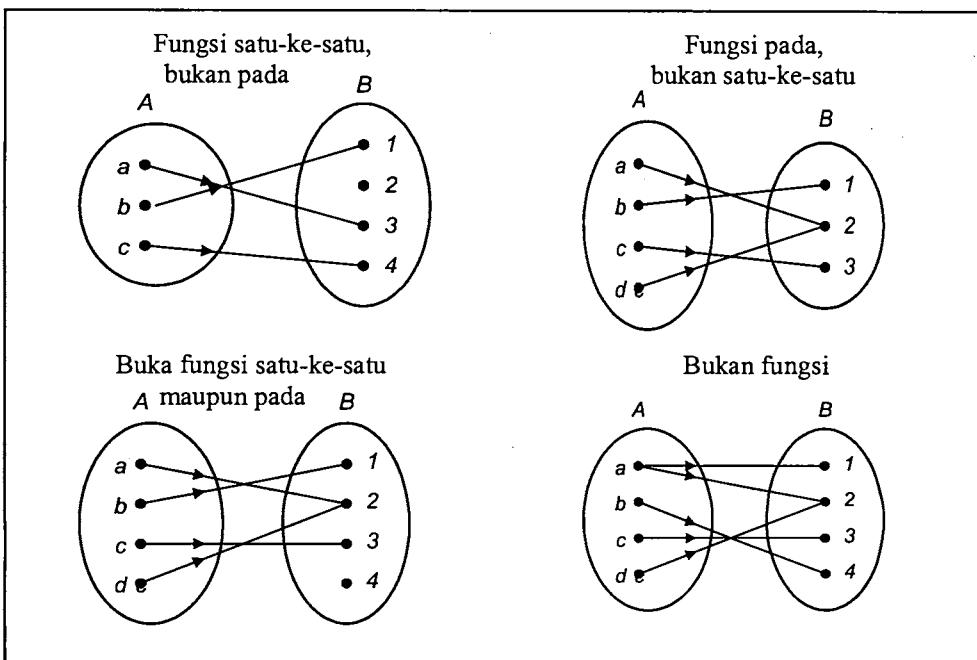
Contoh 3.47

Relasi $f = \{(1, u), (2, w), (3, v)\}$ dari $A = \{1, 2, 3\}$ ke $B = \{u, v, w\}$ adalah fungsi yang berkoresponden satu-ke-satu, karena f adalah fungsi satu-ke-satu maupun fungsi pada.

Contoh 3.48

Fungsi $f(x) = x - 1$ merupakan fungsi yang berkoresponden satu-ke-satu, karena f adalah fungsi satu-ke-satu maupun fungsi pada.

Gambar 3.8 memperlihatkan perbedaan antara fungsi satu-ke-satu tetapi bukan pada, fungsi pada tetapi bukan satu-ke-satu, bukan fungsi satu-ke-satu maupun fungsi pada, dan bukan fungsi.

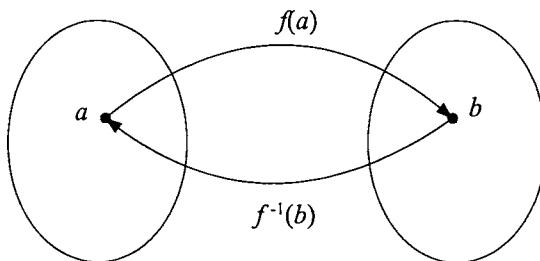


Gambar 3.8 Perbedaan empat tipe korespondensi

3.13 Fungsi Inversi

Jika f adalah fungsi berkoresponden satu-ke-satu dari A ke B , maka kita dapat menemukan **balikan** atau **inversi** (*inverse*) dari f . Fungsi inversi dari f dilambangkan dengan f^{-1} . Misalkan a adalah anggota himpunan A dan b adalah anggota himpunan B , maka $f^{-1}(b) = a$ jika $f(a) = b$. Gambar 3.9 memperlihatkan diagram panah yang menggambarkan f^{-1} sebagai inversi fungsi f .

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.



Gambar 3.9 Fungsi f^{-1} sebagai inversi fungsi f

Contoh 3.49

Relasi $f = \{(1, u), (2, w), (3, v)\}$ dari $A = \{1, 2, 3\}$ ke $B = \{u, v, w\}$ adalah fungsi yang berkoresponden satu-ke-satu. Inversi fungsi f adalah $f^{-1} = \{(u, 1), (w, 2), (v, 3)\}$. Jadi, f adalah fungsi *invertible*.

Contoh 3.50

Tentukan inversi fungsi $f(x) = x - 1$.

Penyelesaian:

Dari Contoh 3.48 kita sudah menyimpulkan bahwa $f(x) = x - 1$ adalah fungsi yang berkoresponden satu-ke-satu, jadi balikan fungsi tersebut ada. Misalkan $f(x) = y$, sehingga $y = x - 1$, maka $x = y + 1$. Jadi, inversi fungsi balikannya adalah $f^{-1}(y) = y + 1$.

Contoh 3.51

Tentukan inversi fungsi $f(x) = x^2 + 1$.

Penyelesaian:

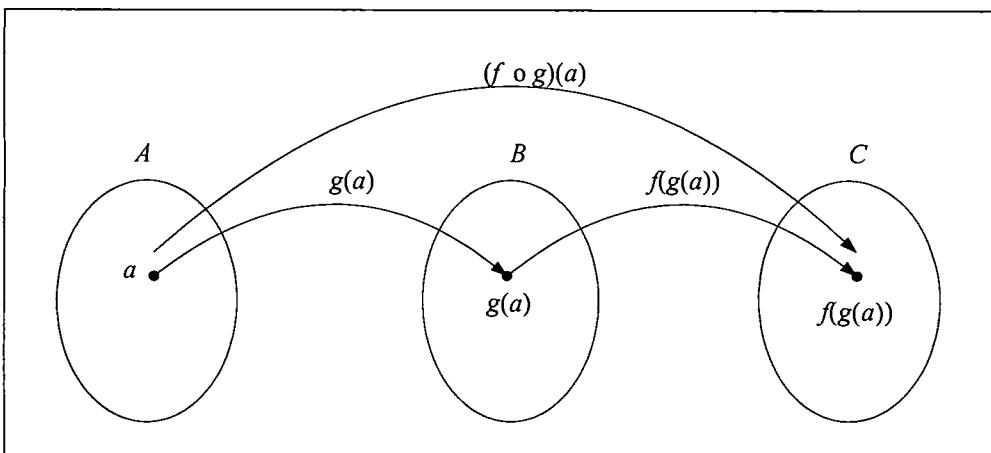
Dari Contoh 3.41 dan 3.44 kita sudah menyimpulkan bahwa $f(x) = x - 1$ bukan fungsi yang berkoresponden satu-ke-satu, sehingga fungsi inversnya tidak ada. Jadi, $f(x) = x^2 + 1$ adalah fungsi yang *not invertible*. ■

3.14 Komposisi Fungsi

Karena fungsi merupakan bentuk khusus dari relasi, kita juga dapat melakukan komposisi dari dua buah fungsi. 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 $f \circ g$, adalah fungsi dari A ke C yang didefinisikan oleh

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

Dengan kata lain, $f \circ g$ adalah fungsi yang memetakan nilai dari $g(a)$ ke f . Gambar 3.10 mengilustrasikan komposisi dua buah fungsi.



Gambar 3.10 Komposisi dua buah fungsi

Contoh 3.52

Diberikan fungsi $g = \{(1, u), (2, u), (3, v)\}$ yang memetakan $A = \{1, 2, 3\}$ ke $B = \{u, v, w\}$, dan fungsi $f = \{(u, y), (v, x), (w, z)\}$ yang memetakan $B = \{u, v, w\}$ ke $C = \{x, y, z\}$. Fungsi komposisi dari A ke C adalah

$$f \circ g = \{(1, y), (2, y), (3, x)\}$$

Contoh 3.53

Diberikan fungsi $f(x) = x - 1$ dan $g(x) = x^2 + 1$. Tentukan $f \circ g$ dan $g \circ f$.

Penyelesaian:

- (i) $(f \circ g)(x) = f(g(x)) = f(x^2 + 1) = x^2 + 1 - 1 = x^2$.
(ii) $(g \circ f)(x) = g(f(x)) = g(x - 1) = (x - 1)^2 + 1 = x^2 - 2x + 2$.
-

Contoh 3.53 ini memperlihatkan bahwa komposisi dua fungsi, f dan g , tidak komutatif, kecuali jika $f = g$.

3.15 Beberapa Fungsi Khusus

Bagian ini memberikan beberapa fungsi yang dipakai di dalam ilmu komputer, yaitu fungsi *floor*, *ceiling*, modulo, faktorial, perpangkatan, dan logaritmik.

1. Fungsi *Floor* dan *Ceiling*

Misalkan x adalah bilangan riil, berarti x berada di antara dua bilangan bulat. Fungsi *floor* dari x , dilambangkan dengan $\lfloor x \rfloor$ dan fungsi *ceiling* dari x dilambangkan dengan $\lceil x \rceil$. Definisi kedua fungsi tersebut adalah:

$\lfloor x \rfloor$ menyatakan nilai bilangan bulat terbesar yang lebih kecil atau sama dengan x
 $\lceil x \rceil$ menyatakan bilangan bulat terkecil yang lebih besar atau sama dengan x

Dengan kata lain, fungsi *floor* membulatkan x ke bawah, sedangkan fungsi *ceiling* membulatkan x ke atas.

Contoh 3.53

Beberapa contoh nilai fungsi *floor* dan *ceiling*:

$$\begin{array}{ll} \lfloor 3.5 \rfloor = 3 & \lceil 3.5 \rceil = 4 \\ \lfloor 0.5 \rfloor = 0 & \lceil 0.5 \rceil = 1 \\ \lfloor 4.8 \rfloor = 4 & \lceil 4.8 \rceil = 5 \\ \lfloor -0.5 \rfloor = -1 & \lceil -0.5 \rceil = 0 \\ \lfloor -3.5 \rfloor = -4 & \lceil -3.5 \rceil = -3 \end{array}$$

Contoh 3.54

Di dalam komputer, data dikodekan dalam untaian *byte*, satu *byte* terdiri atas 8 bit. Jika panjang data 125 bit, maka jumlah *byte* yang diperlukan untuk merepresentasikan data adalah $\lceil 125/8 \rceil = 16$ *byte*. Perhatikanlah bahwa $16 \times 8 = 128$ bit, sehingga untuk *byte* yang terakhir perlu ditambahkan 3 bit ekstra agar satu *byte* tetap 8 bit (bit ekstra yang ditambahkan untuk menggenapi 8 bit disebut *padding bits*). ■

2. Fungsi modulo

Misalkan a adalah sembarang bilangan bulat dan m adalah bilangan bulat positif. Fungsi modulo adalah fungsi dengan operator **mod**, yang dalam hal ini:

$a \text{ mod } m$ memberikan sisa pembagian bilangan bulat bila a dibagi dengan m

Secara lebih rinci, $a \text{ mod } m = r$ sedemikian sehingga $a = mq + r$, dengan $0 \leq r < m$. Lebih jauh mengenai aritmetika modulo akan dibicarakan di dalam Bab 5.

Contoh 3.55

Beberapa contoh fungsi modulo

$$\begin{aligned}25 \text{ mod } 7 &= 4 \\15 \text{ mod } 4 &= 3 \\3612 \text{ mod } 45 &= 12 \\0 \text{ mod } 5 &= 0 \\-25 \text{ mod } 7 &= 3 \quad (\text{sebab } -25 = 7 \cdot (-4) + 3)\end{aligned}$$

Contoh 3.56

Misalkan f adalah fungsi dari $X = \{0, 1, 2, 3, 4\}$ ke X yang didefinisikan oleh $f(x) = 4x \text{ mod } 5$. Tuliskan f sebagai himpunan pasangan terurut. Apakah f fungsi satu-ke-satu (*one-to-one*) atau dipetakan pada (*onto*)?

Penyelesaian:

$$\begin{aligned}x = 0 \rightarrow f(0) &= 4(0) \text{ mod } 5 = 0 \\x = 1 \rightarrow f(1) &= 4(1) \text{ mod } 5 = 4 \\x = 2 \rightarrow f(2) &= 4(2) \text{ mod } 5 = 3 \\x = 3 \rightarrow f(3) &= 4(3) \text{ mod } 5 = 2 \\x = 4 \rightarrow f(4) &= 4(4) \text{ mod } 5 = 1\end{aligned}$$

Jadi, $f = \{(0,0), (1,4), (2,3), (3,2), (4,1)\}$

Jelas fungsi f adalah fungsi satu-ke-satu karena tidak ada dua elemen di X yang mempunyai peta yang sama di himpunan hasil. Fungsi f juga fungsi dipetakan pada (*onto*) karena setiap elemen di X adalah peta dari himpunan daerah asal (yaitu X juga). Dengan kata lain, f adalah fungsi yang berkoresponden satu-ke-satu (*bijection*). ■

3. Fungsi Faktorial

Untuk sembarang bilangan bulat tidak-negatif n , faktorial dari n , dilambangkan dengan $n!$, didefinisikan sebagai

$$n! = \begin{cases} 1 & , n = 0 \\ 1 \times 2 \times \dots \times (n-1) \times n & , n > 0 \end{cases}$$

Contoh 3.57

Beberapa contoh fungsi faktorial

$$0! = 1$$

$$1! = 1$$

$$4! = 1 \cdot 2 \cdot$$

$$5! = 1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 = 120$$

■

4. Fungsi Eksponensial dan Logaritmik

Fungsi eksponensial berbentuk

$$a^n = \begin{cases} 1 & , n = 0 \\ \underbrace{a \times a \times \cdots \times a}_n & , n > 0 \end{cases}$$

Untuk kasus perpangkatan negatif,

$$a^{-n} = \frac{1}{a^n}$$

Fungsi logaritmik berbentuk

$$y = {}^a \log x \Leftrightarrow x = a^y$$

Contoh 3.58

Beberapa contoh fungsi eksponensial dan logaritmik

$$4^3 = 4 \cdot 4 \cdot 4 = 64$$

$$4^{-3} = \frac{1}{64}$$

$${}^4 \log 64 = 3 \text{ karena } 64 = 4^3$$

$$\lfloor {}^2 \log 1000 \rfloor = 9 \text{ karena } 2^9 = 512 \text{ tetapi } 2^{10} = 1024$$

■

3.16 Fungsi Rekursif

Tinjau kembali fungsi untuk menghitung faktorial dari bilangan bulat tak-negatif n yang didefinisikan sebagai berikut:

$$n! = \begin{cases} 1 & , n = 0 \\ 1 \times 2 \times \cdots \times (n-1) \times n & , n > 0 \end{cases}$$

Sebagai contoh,

$$\begin{aligned}0! &= 1 \\1! &= 1 \\2! &= 1 \times 2 \\3! &= 1 \times 2 \times 3 \\4! &= 1 \times 2 \times 3 \times 4\end{aligned}$$

Sekarang coba perhatikan bahwa faktorial dari n dapat didefinisikan dalam terminologi faktorial juga:

$$\begin{aligned}0! &= 1 \\1! &= 1 \times 0! \\2! &= 2 \times 1! \\3! &= 3 \times 2! \\4! &= 4 \times 3!\end{aligned}$$

Nyatalah, bahwa untuk $n > 0$ kita melihat bahwa

$$n! = 1 \times 2 \times \dots \times (n-1) \times n = (n-1)! \times n.$$

Dengan menggunakan notasi matematika, maka $n!$ didefinisikan dalam hubungan rekursif sebagai berikut:

$$n! = \begin{cases} 1 & , n = 0 \\ n \times (n-1)! & , n > 0 \end{cases}$$

Jika kita misalkan $f(n) = n!$, maka fungsi faktorial di atas dapat juga ditulis sebagai

$$f(n) = \begin{cases} 1 & , n = 0 \\ n \times f(n-1) & , n > 0 \end{cases}$$

Kita dapat melihat bahwa dalam proses perhitungan faktorial bilangan tak-negatif n terdapat definisi faktorial itu sendiri. Cara pendefinisian seperti itu, yang mendefinisikan sebuah objek dalam terminologi dirinya sendiri dinamakan definisi rekursif.

DEFINISI 3.15. Fungsi f dikatakan fungsi rekursif jika definisi fungsinya mengacu pada dirinya sendiri.

Nama lain dari fungsi rekursif adalah **relasi rekursif** (*recurrence relation*). Ingatlah bahwa fungsi adalah bentuk khusus dari relasi.

Fungsi rekursif disusun oleh dua bagian:

(a) *Basis*

Bagian yang berisi nilai awal yang tidak mengacu pada dirinya sendiri. Bagian ini juga sekaligus menghentikan definisi rekursif (dan memberikan sebuah nilai yang terdefinisi pada fungsi rekursif).

(b) *Rekurens*

Bagian ini mendefinisikan argumen fungsi dalam terminologi dirinya sendiri. Setiap kali fungsi mengacu pada dirinya sendiri, argumen dari fungsi harus lebih dekat ke nilai awal (basis).

Tinjau kembali perhitungan $n!$ secara rekursif. Dengan mengingat kembali definisi rekursif dari faktorial:

(a) basis:

$$n! = 1 \quad , \text{ jika } n = 0$$

(b) rekurens:

$$n! = n \times (n - 1)! \quad , \text{ jika } n > 0$$

maka $5!$ dihitung dengan langkah berikut:

$$(1) 5! = 5 \times 4! \quad (\text{rekurens})$$

$$(2) \quad 4! = 4 \times 3!$$

$$(3) \quad 3! = 3 \times 2!$$

$$(4) \quad 2! = 2 \times 1!$$

$$(5) \quad 1! = 1 \times 0!$$

$$(6) \quad 0! = 1$$

Pada baris (6) kita memperoleh nilai yang terdefinisi secara langsung dan bukan faktorial dari bilangan lainnya. Dengan melakukan runut-balik (*backtrack*) dari baris (6) ke baris (1), kita mendapatkan nilai pada setiap baris untuk menghitung hasil pada baris sebelumnya:

$$(6') \quad 0! = 1$$

$$(5') \quad 1! = 1 \times 0! = 1 \times 1 = 1$$

$$(4') \quad 2! = 2 \times 1! = 2 \times 1 = 2$$

$$(3') \quad 3! = 3 \times 2! = 3 \times 2 = 6$$

$$(2') \quad 4! = 4 \times 3! = 4 \times 6 = 24$$

$$(1') \quad 5! = 5 \times 4! = 5 \times 24 = 120$$

Jadi, $5! = 120$.

Contoh 3.59

Di bawah ini adalah contoh-contoh fungsi rekursif lainnya:

$$1. \quad F(x) = \begin{cases} 0 & , x = 0 \\ 2F(x-1) + x^2 & , x \neq 0 \end{cases}$$

2. Fungsi Chebysev

$$T(n, x) = \begin{cases} 1 & , n = 0 \\ x & , n = 1 \\ 2xT(n-1, x) - T(n-2, x) & , n > 1 \end{cases}$$

Fungsi Chebysev di atas mempunyai dua buah basis, yaitu jika $n = 0$ dan $n = 1$.

3. Fungsi Fibonacci:

$$f(n) = \begin{cases} 0 & , n = 0 \\ 1 & , n = 1 \\ f(n-1) + f(n-2) & , n > 1 \end{cases}$$

Contoh 3.59

Nyatakan perpangkatan a^n (a bulat dan $n > 0$) sebagai fungsi rekursif.

Penyelesaian:

(i) Nyatakan a^n dalam argumen rekursif

$$\begin{aligned} a^n &= a \times a \times a \times \dots \times a && (\text{sebanyak } n \text{ kali}) \\ &= a \times a^{n-1} && (\text{rekurens}) \end{aligned}$$

(ii) Tentukan kasus eksplisit yang tidak memerlukan pemanggilan rekursif lagi (basis)

$$a^n = 1 \quad \text{jika } n = 0 \quad (\text{basis})$$

Jadi, fungsi rekursif untuk perpangkatan adalah:

$$a^n = \begin{cases} 1 & , n = 0 \\ a \times a^{n-1} & , n > 0 \end{cases}$$

3.17 Ragam Soal dan Penyelesaian

Contoh 3.60

Misalkan \perp adalah relasi tegak lurus dari himpunan garis-garis di bidang Euclidean. Apakah relasi \perp setangkup? Refleksif? Tolak-setangkup? Menghantar?

Penyelesaian:

$(a, b) \in \perp$ jika dan hanya jika garis $a \perp$ garis b . Jelas, jika garis a tegak lurus dengan garis b , maka garis b juga tegak lurus dengan garis a . Berarti, $(b, a) \in \perp$, yang menunjukkan bahwa relasi \perp bersifat setangkup.

\perp tidak refleksif karena $(a, a) \notin \perp$ untuk setiap garis di dalam himpunan garis, artinya setiap garis tidak tegak lurus dengan dirinya sendiri.

Kita juga dapat memperlihatkan bahwa \perp tidak tolak-setangkup dan tidak mengantar (tunjukkan!). ■

Contoh 3.61

Misalkan relasi R pada himpunan bilangan bulat positif sedemikian sehingga $R = \{(a, b) \mid a < b\}$. Apa klosur setangkup dari R ?

Penyelesaian:

Klosur setangkup dari relasi R adalah

$$R \cup R^{-1} = \{(a, b) \mid a < b\} \cup \{(b, a) \mid b < a\} = \{(a, b) \mid a \neq b\}$$
 ■

Contoh 3.62

Misalkan relasi R pada himpunan $A = \{1, 2, 3\}$ direpresentasikan dengan menggunakan matriks berikut:

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

Apakah relasi R merupakan relasi kesetaraan?

Penyelesaian:

Relasi kesetaraan harus memiliki sifat setangkup, refleksif, dan transitif. Relasi R setangkup dilihat dari matriksnya yang setangkup terhadap diagonal. Relasi R mengantar, ini dapat dilihat dari penyajian relasi dengan himpunan pasangan terurut: $\{(1, 1), (1, 3), (3, 1), (3, 3)\}$; dapat dilihat bahwa selalu ada $(a, c) \in R$ jika $(a, b) \in R$ dan $(b, c) \in R$. Relasi R tidak refleksif karena $(2, 2) \notin R$. Relasi R tidak refleksif karena $(2, 2) \notin R$. Karena R tidak refleksif, maka R bukan relasi kesetaraan. ■

Contoh 3.63

Fungsi manakah dari yang berikut ini yang mempunyai balikan atau inversi?

(a) $f(x) = 2x + 1$ (b) $f(x) = x^4 + 1$ (c) $f(x) = x^3$

Penyelesaian:

Syarat fungsi mempunyai inversi adalah fungsi tersebut berkoresponden satu-ke-satu atau bijeksi. Fungsi f dikatakan berkoresponden satu-ke-satu jika ia fungsi satu-ke-satu dan juga fungsi pada. Fungsi yang berkoresponden satu-ke-satu adalah fungsi a dan c. Fungsi b tidak berkoresponden satu-ke-satu karena ia tidak satu-ke-satu, karena untuk dua nilai x yang bernilai mutlak sama tetapi tandanya berbeda nilai fungsinya sama, misalnya $f(1) = f(-1) = 2$ padahal $1 \neq -1$. Jadi, fungsi yang mempunyai balikan adalah fungsi a dan c saja. ■

Contoh 3.64

Misalkan n menyatakan bilangan bulat positif. Misalkan sebuah fungsi f didefinisikan secara rekursif sebagai berikut:

$$f(n) = \begin{cases} 0 & , n = 1 \\ f(\lfloor n/2 \rfloor + 1) & , n > 1 \end{cases}$$

- (a) Tentukan $f(25)$. (b) Fungsi apakah f tersebut?

Penyelesaian:

(a)
$$\begin{aligned} f(25) &= f(12) + 1 \\ &= [f(6) + 1] + 1 = f(6) + 2 \\ &= [f(3) + 1] + 2 = f(3) + 3 \\ &= [f(1) + 1] + 3 = f(1) + 4 \\ &= 0 + 4 = 4 \end{aligned}$$

- (b) f adalah bilangan bulat terbesar sedemikian sehingga $2^f \leq n$. Dengan demikian,
$$f(n) = \lfloor \log_2 n \rfloor$$
 ■
-

Soal Latihan

1. Tuliskan pasangan terurut pada relasi R dari $A = \{0, 1, 2, 3, 4\}$ ke $B = \{0, 1, 2, 3\}$ yang dalam hal ini pasangan terurut $(a, b) \in R$ jika dan hanya jika $a > b$.
2. Tuliskan anggota dari relasi R pada $\{1, 2, 3, 4\}$ yang didefinisikan oleh $(x, y) \in R$ jika $x^2 \geq y$.
3. Nyatakan relasi $R = \{(1, 2), (2, 1), (3, 3), (1, 1), (2, 2)\}$ pada $X = \{1, 2, 3\}$ dalam bentuk tabel, matriks, dan graf berarah.
4. Untuk tiap relasi pada $\{1, 2, 3, 4\}$ berikut, tentukan apakah ia refleksif, setangkup, tak-setangkup, dan menghantar.
 - (a) $\{(2, 2), (2, 3), (2, 4), (3, 2), (3, 3), (3, 4)\}$
 - (b) $\{(2, 4), (4, 2)\}$
 - (c) $\{(1, 1), (2, 2), (3, 3), (4, 4)\}$
 - (d) $\{(1, 3), (1, 4), (2, 3), (2, 4), (3, 1), (3, 4)\}$
5. Tentukan apakah relasi R pada himpunan orang bersifat refleksif, setangkup, tak-setangkup, dan/atau menghantar, yang dalam hal ini $(a, b) \in R$ jika dan hanya jika
 - (a) a lebih tinggi daripada b
 - (b) a dan b lahir pada hari yang sama
 - (c) a mempunyai nama pertama yang sama dengan b
6. Misalkan R adalah relasi $\{(1, 2), (1, 3), (2, 3), (2, 4), (3, 1)\}$ dan S adalah relasi $\{(2, 1), (3, 1), (3, 2), (4, 2), (4, 2)\}$. Tentukan $S \circ R$ dan $R \circ S$.
7. Misalkan $R = \{(1, 2), (2, 3), (3, 4)\}$ dan $S = \{(1, 1), (1, 2), (2, 1), (2, 2), (2, 3), (3, 1), (3, 2), (3, 4)\}$ adalah relasi dari $\{1, 2, 3\}$ ke $\{1, 2, 3, 4\}$. Tentukan
 - (a) $R \cup S$
 - (b) $R \cap S$
 - (c) $R - S$
 - (d) $S - R$
 - (e) $R \oplus S$
8. Misalkan R adalah relasi pada himpunan orang yang terdiri dari pasangan (a, b) yang dalam hal ini a adalah ayah dari b . Misalkan S adalah relasi pada himpunan orang yang terdiri dari pasangan (a, b) yang dalam hal ini a dan b adalah saudara kandung. Nyatakan $R \circ S$.

9. Nyatakan pasangan terurut dari relasi pada $\{1, 2, 3\}$ yang berkoresponden dengan matriks berikut:

$$(a) \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \quad (b) \begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

10. Gambarkan graf berarah dari relasi yang dinyatakan oleh matriks pada soal nomor 9.

11. Misalkan bahwa relasi R dan S pada himpunan A dinyatakan oleh matriks

$$R = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \text{ dan } S = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

Tentukan matriks yang menyatakan

- (a) $R \cup S$ (b) $R \cap S$ (c) $R \circ S$

12. Misalkan $A = \{1, 2, 3, 4\}$, dan $R = \{(1, 1), (2, 3), (4, 4), (2, 1)\}$ adalah relasi pada himpunan A .

- (a) Dari keempat sifat ini: refleksif, menghantar, setangkup, dan anti-setangkup, sifat apa yang dimiliki oleh relasi R ? Jelaskan alasannya.
 (b) Nyatakan hasil operasi R^2 sebagai himpunan pasangan terurut.

13. Sebuah relasi R yang didefinisikan pada sebuah himpunan yang beranggotan 4 buah elemen disajikan dalam matriks M sebagai berikut:

$$M_R = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

Tentukan apakah relasi tersebut refleksif/tidak refleksif, setangkup/tidak setangkup, menghantar/tidak menghantar, tolak-setangkup/tidak tolak-setangkup.

14. Tinjau matriks relasi pada soal nomor 12 di atas. Tentukan matriks yang merepresentasikan:

- (a) R^{-1} (b) \bar{R} (c) R^2 (d) R^3

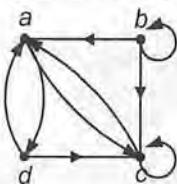
15. Sebuah relasi R yang didefinisikan pada sebuah himpunan yang beranggotakan 4 buah elemen disajikan dalam matriks M sebagai berikut:

$$M_R = \begin{bmatrix} a & 1 & 0 & 1 \\ b & c & 1 & 1 \\ d & e & f & 0 \\ g & h & i & j \end{bmatrix}$$

Tentukan nilai $a, b, c, d, e, f, g, h, i$, dan j agar relasi tersebut bersifat:

- (a) refleksif
- (b) setangkup
- (c) tolak-setangkup

16. Diketahui relasi S yang didefinisikan pada himpunan $A = \{a, b, c, d\}$. Relasi direpresentasikan dalam graf berarah berikut ini:



- (a) Jelaskan alasan mengapa relasi S tidak bersifat menghantar. Tambahkan busur tambahan yang dimaksud sehingga S bersifat menghantar.
 - (b) Jika didefinisikan bahwa $S^n = S \circ S \circ \dots \circ S$ (sebanyak n kali), tentukan matriks dan graf berarah yang merepresentasikan S^2 (graf berarah S yang digunakan adalah graf pada gambar soal)
16. Misalkan $R = \{(0, 1), (1, 1), (1, 2), (2, 0), (2, 2)\}$ adalah relasi pada himpunan $\{0, 1, 2, 3\}$. Temukan klosur refleksif dan klosur setangkup dari R .
17. Misalkan relasi R pada himpunan A dinyatakan dengan matriks M_R . Tunjukkan bahwa matriks yang merepresentasikan klosur setangkup adalah $M_R \vee M_R^T$.
18. Temukan klosur menghantar dari relasi $\{(2, 1), (2, 3), (3, 1), (3, 4), (4, 1), (4, 3)\}$ pada himpunan $\{1, 2, 3, 4\}$.
19. Misalkan relasi R refleksif. Tunjukkan bahwa R^* refleksif. Dengan cara yang sama, misalkan relasi R setangkup. Tunjukkan bahwa R^* setangkup.

20. Manakah relasi pada $\{1, 2, 3\}$ berikut yang merupakan relasi kesetaraan?
- (a) $\{(0, 0), (1, 1), (2, 2), (3, 3)\}$
 - (b) $\{(0, 0), (0, 2), (2, 0), (2, 2), (2, 3), (3, 2), (3, 3)\}$
 - (c) $\{(0, 0), (1, 1), (1, 3), (2, 2), (2, 3), (3, 1), (3, 2), (3, 3)\}$
 - (d) $\{(0, 0), (0, 1), (0, 2), (1, 0), (1, 1), (1, 2), (2, 0), (2, 2), (3, 3)\}$
21. Manakah relasi pada himpunan orang berikut ini yang merupakan relasi kesetaraan?
- (a) $\{(a, b) \mid a \text{ and } b \text{ berumur sama}\}$
 - (b) $\{(a, b) \mid a \text{ and } b \text{ berbicara dengan bahasa yang sama}\}$
 - (c) $\{(a, b) \mid a \text{ and } b \text{ pernah bertemu}\}$
 - (d) $\{(a, b) \mid a \text{ and } b \text{ mempunyai orangtua yang sama}\}$
22. Misalkan R adalah relasi pada himpunan pasangan terurut dari bilangan bulat positif sedemikian sehingga $((a, b), (c, d)) \in R$ jika dan hanya jika $ad = bc$. Tunjukkan bahwa R adalah relasi kesetaraan.
23. Misalkan R adalah relasi pada himpunan *URL* (alamat *Web*) sedemikians sehingga xRy jika dan hanya jika halaman *Web* pada x sama dengan halaman *Web* pada y . Tunjukkan bahwa R adalah relasi kesetaraan.
24. Manakah dari berikut ini yang merupakan *poset*?
- (a) $(\mathbf{Z}, =)$
 - (b) (\mathbf{Z}, \neq)
 - (c) $(\mathbf{Z}, <)$
25. Manakah relasi yang disajikan dengan matriks berikut yang merupakan relasi pengurutan parsial?
- (a) $\begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
- (b) $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$
- (c) $\begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{bmatrix}$
26. Mengapa persamaan berikut bukan merupakan fungsi dari \mathbf{R} ke \mathbf{R} ?
- (a) $f(x) = 1/x$
 - (b) $f(x) = \sqrt{x}$
 - (c) $f(x) = \pm\sqrt{x^2 + 1}$
27. Tentukan fungsi mana yang merupakan fungsi satu-ke-satu dari \mathbf{Z} ke \mathbf{Z} :
- (a) $f(n) = n + 2$
 - (b) $f(n) = n^3$
 - (c) $f(n) = \lceil n/2 \rceil$

28. Tentukan apakah setiap fungsi berikut satu-ke-satu?
- Setiap orang di bumi memetakan jumlah usianya.
 - Setiap negara di dunia memetakan letak garis lintang dan garis bujur ibukotanya.
 - Setiap buku yang ditulis oleh pengarangnya memetakan nama pengarangnya.
 - Setiap negara di dunia yang mempunyai seorang presiden memetakan nama presidennya.
29. Misalkan $g = \{(1, b), (2, c), (3, a), (4, b)\}$ adalah fungsi dari $A = \{1, 2, 3, 4\}$ ke $B = \{a, b, c, d\}$ dan $f = \{(a, x), (b, y), (c, w), (d, z)\}$ adalah fungsi dari B ke $C = \{w, x, y, z\}$.
- tuliskan $f \circ g$ sebagai himpunan pasangan terurut.
 - apakah $f \circ g$ merupakan fungsi *injektif*, *surjektif*, atau *bijektif*?
30. Jika diberikan $g = \{(1, b), (2, c), (3, a)\}$ adalah fungsi dari $A = \{1, 2, 3\}$ ke $B = \{a, b, c, d\}$ dan $f = \{(a, x), (b, x), (c, z), (d, w)\}$ adalah fungsi dari B ke $C = \{w, x, y, z\}$, tuliskan $f \circ g$ sebagai himpunan pasangan terurut.
31. Berapa banyak *byte* yang dibutuhkan untuk mengkodekan data yang panjangnya 1001 bit?
32. Di dalam suatu jaringan komputer, data dikirim dalam bentuk blok-blok bit. Setiap blok panjangnya 128 *byte*. Jika data dikirim melalui media transmisi dengan kecepatan 500 *kbytes per second*, maka selama 1 menit berapa banyak blok data dapat dikirim? (Catatan: 1 kilobit = 1000 bit).
33. Misalkan f adalah fungsi dari $X = \{0, 1, 2, 3, 4\}$ ke X yang didefinisikan oleh $f(x) = 3x \bmod 5$. Tuliskan f sebagai himpunan pasangan terurut. Apakah f satu-ke-satu atau pada?
34. Nyatakan $a \times b$ sebagai fungsi rekursif.
35. Fungsi Ackermann adalah fungsi rekursif dengan dua buah peubah bilangan bulat yang didefinisikan sebagai berikut:
- Jika $m = 0$ maka $A(m, n) = n + 1$
 - Jika $m \neq 0$ tetapi $n = 0$ maka $A(m, n) = A(m - 1, 1)$
 - Jika $m \neq 0$ dan $n \neq 0$ maka $A(m, n) = A(m - 1, A(m, n - 1))$

Tentukan nilai $A(1, 3)$.

BAB 4

Induksi Matematika

Kebijaksanaan hanya ditemukan di dalam kebenaran.
(Goethe)

Di dalam matematika, sebuah proposisi atau pernyataan tidak hanya sekadar ditulis. Kita juga harus mengerti apa yang menyebabkan proposisi tersebut benar, yaitu bukti (*proof*). Di dalam Bab 1 kita sudah membicarakan metode pembuktian untuk argumen, dan di dalam Bab 3 kita membicarakan pembuktian proposisi yang menyangkut himpunan. Di dalam Bab 4 ini kita memfokuskan pembuktian proposisi yang hanya menyangkut bilangan bulat, misalnya pembuktian pernyataan “Jumlah n buah bilangan bulat positif pertama adalah $n(n + 1)/2$ ”. Metode pembuktian untuk proposisi perihal bilangan bulat adalah **induksi matematika**.

Induksi matematik merupakan teknik pembuktian yang baku di dalam matematika. Melalui induksi matematik kita dapat mengurangi langkah-langkah pembuktian bahwa semua bilangan bulat termasuk ke dalam suatu himpunan kebenaran dengan hanya sejumlah langkah terbatas.

Menurut sejarahnya, induksi matematika berawal pada akhir abad ke-19. Dua orang matematikawan yang mempelopori perkembangan induksi matematika adalah R. Dedekind dan G. Peano [DOE85]. Dedekind mengembangkan sekumpulan aksioma yang mengambarkan bilangan bulat positif. Peano memperbaiki aksioma

tesebut dan memberikannya interpretasi logis. Keseluruhan aksioma tersebut dinamakan *Postulat Peano*.

Bab 4 ini berisi prinsip-prinsip induksi matematika, mulai dari induksi sederhana, perampatannya, sampai pada bahasan bentuk induksi secara umum.

4.1 Proposisi Perihal Bilangan Bulat

Proposisi yang menyangkut perihal bilangan bulat cukup banyak dijumpai di dalam matematika diskrit maupun di dalam ilmu komputer. Proposisi tersebut mengaitkan suatu masalah yang dihubungkan dengan bilangan bulat. Untuk memberikan ilustrasi mengenai proposisi seperti apa yang dimaksudkan, marilah tinjau dua contoh proposisi sederhana sebagai berikut.

Di dalam matematika, banyak teorema yang menyatakan bahwa $p(n)$ benar untuk semua bilangan bulat positif n , yang dalam hal ini $p(n)$ disebut juga fungsi proposisi. Contoh pertama, misalkan $p(n)$ adalah proposisi yang menyatakan: "Jumlah bilangan bulat positif dari 1 sampai n adalah $n(n + 1)/2$ ". Buktikan bahwa $p(n)$ benar!

Kalau kita coba dengan beberapa nilai n , memang timbul dugaan bahwa $p(n)$ benar. Misalnya untuk $n = 5$, $p(5)$ adalah: Jumlah bilangan bulat positif dari 1 sampai 5 adalah $5(5 + 1)/2$. Terlihat bahwa

$$1 + 2 + 3 + 4 + 5 = 15 = 5(6)/2$$

Untuk nilai-nilai n yang lain kita akan dapatkan kesimpulan serupa. Sayangnya, instansiasi seperti $p(5)$ tidak dapat berlaku sebagai bukti bahwa $p(n)$ benar untuk seluruh n . Kita memang sudah menunjukkan bahwa $n = 5$ berada di dalam himpunan kebenaran $p(n)$. Tetapi, kita tahu bahwa 5 bukanlah satu-satunya bilangan bulat positif. Karena bilangan bulat positif tidak terhingga banyaknya, kita tentu tidak mungkin mencoba sekuruhnya untuk membuktikan $p(n)$ benar. Jadi, kita tidak dapat menggunakan pendekatan semacam ini untuk membuktikan kebenaran pernyataan perihal bilangan bulat.

Contoh kedua, kita ingin menemukan rumus jumlah dari n buah bilangan ganjil positif yang pertama. Misalnya untuk $n = 1, 2, 3, 4, 5$, kita mengamati jumlah n bilangan ganjil positif pertama adalah

$$n = 1 \Rightarrow 1 = 1$$

$$n = 2 \Rightarrow 1 + 3 = 4$$

$$n = 3 \Rightarrow 1 + 3 + 5 = 9$$

$$n = 4 \Rightarrow 1 + 3 + 5 + 7 = 16$$

$$n = 5 \Rightarrow 1 + 3 + 5 + 7 + 9 = 25$$

Dari nilai-nilai penjumlahan itu kita meduga bahwa jumlah n buah bilangan ganjil positif pertama adalah n^2 . Kita perlu membuktikan bahwa perkiraan kita tersebut benar jika memang itu faktanya. Bagaimana cara membuktikannya dengan induksi matematik?

Contoh-contoh proposisi perihal bilangan bulat yang lainnya misalnya:

1. Setiap bilangan bulat positif n ($n \geq 2$) dapat dinyatakan sebagai perkalian dari (satu atau lebih) bilangan prima.
2. Untuk semua $n \geq 1$, $n^3 + 2n$ adalah kelipatan 3.
3. Untuk membayar biaya pos sebesar n sen dolar ($n \geq 8$) selalu dapat digunakan hanya perangko 3 sen dan 5 sen dolar.
4. Di dalam sebuah pesta, setiap tamu berjabat tangan dengan tamu lainnya hanya sekali. Jika ada n orang tamu maka jumlah jabat tangan yang terjadi adalah $n(n - 1)/2$.
5. Banyaknya himpunan bagian yang dapat dibentuk dari sebuah himpunan yang beranggotakan n elemen adalah 2^n .

Proposisi-proposisi semacam di ataslah yang dapat dibuktikan dengan induksi matematika. Mari kita pahami cara pembuktian dengan induksi matematika, dimulai dengan prinsip induksi sederhana terlebih dahulu, seperti yang dijelaskan di dalam upabab berikut.

4.2 Prinsip Induksi Sederhana

Prinsip induksi sederhana berbunyi sebagai berikut:

Misalkan $p(n)$ adalah proposisi perihal bilangan bulat positif dan kita ingin membuktikan bahwa $p(n)$ benar untuk semua bilangan bulat positif n . Untuk membuktikan proposisi ini, kita hanya perlu menunjukkan bahwa:

1. $p(1)$ benar, dan
2. jika $p(n)$ benar, maka $p(n + 1)$ juga benar untuk setiap $n \geq 1$.

sehingga $p(n)$ benar untuk semua bilangan bulat positif n .

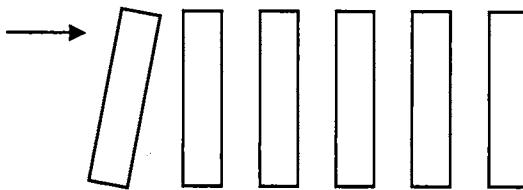
Langkah 1 dinamakan **basis induksi**, sedangkan langkah 2 dinamakan **langkah induksi**. Langkah induksi berisi asumsi (andaian) yang menyatakan bahwa $p(n)$ benar. Asumsi tersebut dinamakan **hipotesis induksi**. Bila kita sudah menunjukkan kedua langkah tersebut benar maka kita sudah membuktikan bahwa $p(n)$ benar untuk semua bilangan bulat positif n .

Basis induksi digunakan untuk memperlihatkan bahwa pernyataan tersebut benar bila n diganti dengan 1, yang merupakan bilangan bulat positif terkecil. Kemudian kita harus memperlihatkan bahwa implikasi $p(n) \rightarrow p(n + 1)$ benar untuk setiap

bilangan bulat positif. Untuk membuktikan implikasi tersebut benar untuk setiap bilangan bulat positif n , kita perlu menunjukkan bahwa $p(n+1)$ tidak mungkin salah bila $p(n)$ benar. Hal ini diselesaikan dengan cara memperlihatkan bahwa berdasarkan hipotesis $p(n)$ benar maka $p(n+1)$ juga harus benar.

Perhatikan bahwa dalam induksi matematik kita tidak mengasumsikan bahwa $p(n)$ benar untuk *semua* bilangan bulat positif. Kita hanya memperlihatkan bahwa jika diasumsikan $p(n)$ benar, maka $p(n + 1)$ juga benar untuk setiap n positif [ROS03].

Fakta bahwa langkah 1 dan langkah 2 bersama-sama memperlihatkan $p(n)$ benar untuk semua bilangan bulat positif adalah jelas secara intuitif. Dari langkah 1, kita mengetahui bahwa $p(1)$ benar. Dari langkah (2) kita mengetahui bahwa jika $p(1)$ benar maka $p(2)$ juga benar. Tetapi, $p(1)$ sudah ditunjukkan benar dan di sini $p(2)$ juga harus benar. Dari langkah (2) kita juga mengetahui bahwa jika $p(2)$ benar maka $p(3)$ juga benar. Karena kita sudah menunjukkan bahwa $p(2)$ benar, maka $p(3)$ juga benar, dan seterusnya. Secara intuitif kita melihat bahwa langkah 1 dan langkah 2 bersama-sama memperlihatkan bahwa $p(1), p(2), \dots, p(n)$ semuanya benar. Pembuktian dengan induksi matematik mirip dapat kita ilustrasikan dengan fenomena yang dikenal dengan **efek domino**. Sejumlah batu domino diletakkan berdiri dengan jarak ruang yang sama satu sama lain (lihat Gambar 4.1). Untuk merebahkan semua batu domino, kita hanya perlu mendorong domino 1 ke kanan. Jika domino 1 di dorong ke kanan, ia akan mendorong domino 2, domino 2 mendorong domino 3, begitu seterusnya sehingga semua batu domino rebah ke kanan.



Gambar 4.1 Efek domino

Contoh-contoh berikut memperlihatkan pembuktian proposisi perihal bilangan bulat dengan menggunakan prinsip induksi sederhana.

Contoh 4.1

Tunjukkan bahwa untuk $n \geq 1$, $1 + 2 + 3 + \dots + n = n(n + 1)/2$ melalui induksi matematika.

Penyelesaian:

Andaikan bahwa $p(n)$ menyatakan proposisi bahwa untuk $n \geq 1$, jumlah n bilangan bulat positif pertama adalah $n(n + 1)/2$, yaitu $1 + 2 + 3 + \dots + n = n(n + 1)/2$. Kita harus membuktikan kebenaran proposisi ini dengan dua langkah induksi sebagai berikut:

- (i) *Basis induksi:* $p(1)$ benar, karena untuk $n = 1$ kita peroleh

$$\begin{aligned}1 &= 1(1 + 1)/2 \\&= 1(2)/2 \\&= 2/2 \\&= 1\end{aligned}$$

- (ii) *Langkah induksi:* Misalkan $p(n)$ benar, yaitu mengasumsikan bahwa

$$1 + 2 + 3 + \dots + n = n(n + 1)/2$$

adalah benar (hipotesis induksi). Kita harus memperlihatkan bahwa $p(n + 1)$ juga benar, yaitu

$$1 + 2 + 3 + \dots + n + (n + 1) = (n + 1)[(n + 1) + 1]/2$$

Untuk membuktikan ini, tunjukkan bahwa

$$\begin{aligned}1 + 2 + 3 + \dots + n + (n + 1) &= (1 + 2 + 3 + \dots + n) + (n + 1) \\&= [n(n + 1)/2] + (n + 1) \\&= [(n^2 + n)/2] + (n + 1) \\&= [(n^2 + n)/2] + [(2n + 2)/2] \\&= (n^2 + 3n + 2)/2 \\&= (n + 1)(n + 2)/2 \\&= (n + 1)[(n + 1) + 1]/2\end{aligned}$$

Karena langkah (i) dan (ii) telah dibuktikan benar, maka untuk semua bilangan bulat positif n , terbukti bahwa untuk semua $n \geq 1$, $1 + 2 + 3 + \dots + n = n(n + 1)/2$. ■

Contoh 4.2

Gunakan induksi matematik untuk membuktikan bahwa jumlah n buah bilangan ganjil positif pertama adalah n^2 .

Penyelesaian:

Misakan $p(n)$ adalah proposisi yang menyatakan bahwa jumlah n buah bilangan ganjil positif pertama adalah n^2 .

- (i) *Basis induksi:* $p(1)$ benar, karena jumlah satu buah bilangan ganjil positif pertama adalah $1^2 = 1$.

(ii) *Langkah induksi:* Misalkan $p(n)$ benar, yaitu asumsikan bahwa

$$1 + 3 + 5 + \dots + (2n - 1) = n^2$$

adalah benar (hipotesis induksi) [catatlah bahwa bilangan ganjil positif ke- n adalah $(2n - 1)$].

Kita harus memperlihatkan bahwa $p(n + 1)$ juga benar, yaitu

$$1 + 3 + 5 + \dots + (2n - 1) + (2n + 1) = (n + 1)^2$$

Hal ini dapat kita tunjukkan sebagai berikut:

$$\begin{aligned} 1 + 3 + 5 + \dots + (2n - 1) + (2n + 1) &= [1 + 3 + 5 + \dots + (2n - 1)] + (2n + 1) \\ &= n^2 + (2n + 1) \\ &= n^2 + 2n + 1 \\ &= (n + 1)^2 \end{aligned}$$

Karena langkah basis dan langkah induksi keduanya telah diperlihatkan benar, maka jumlah n buah bilangan ganjil positif pertama adalah n^2 . ■

Contoh 4.3

Untuk semua $n \geq 1$, buktikan dengan induksi matematik bahwa $n^3 + 2n$ adalah kelipatan 3.

Penyelesaian:

Misalkan $p(n)$ adalah proposisi yang menyatakan bahwa untuk semua $n \geq 1$, $n^3 + 2n$ adalah kelipatan 3.

(i) *Basis induksi:* $p(1)$ benar, karena untuk $n = 1$, $1^3 + 2(1) = 3$ adalah kelipatan 3.

(ii) *Langkah induksi:* Misalkan $p(n)$ benar, yaitu proposisi

$$n^3 + 2n \text{ adalah kelipatan 3}$$

diasumsikan benar (hipotesis induksi). Kita harus memperlihatkan bahwa $p(n + 1)$ juga benar, yaitu

$$(n + 1)^3 + 2(n + 1) \text{ adalah kelipatan 3}$$

Hal ini dapat kita tunjukkan sebagai berikut:

$$\begin{aligned} (n + 1)^3 + 2(n + 1) &= (n^3 + 3n^2 + 3n + 1) + (2n + 2) \\ &= (n^3 + 2n) + 3n^2 + 3n + 3 \\ &= (n^3 + 2n) + 3(n^2 + n + 1) \end{aligned}$$

Karena $(n^3 + 2n)$ adalah kelipatan 3 (dari hipotesis induksi) dan $3(n^2 + n + 1)$ juga kelipatan 3, maka $(n^3 + 2n) + 3(n^2 + n + 1)$ adalah jumlah dua buah bilangan kelipatan 3; karena itu $(n^3 + 2n) + 3(n^2 + n + 1)$ juga kelipatan tiga. Jadi, untuk $n \geq 1$, $n^3 + 2n$ adalah kelipatan 3.

Karena langkah (i) dan (ii) sudah diperlihatkan benar, maka terbukti bahwa untuk semua $n \geq 1$, $n^3 + 2n$ adalah kelipatan 3. ■

Contoh 4.4

Cuplikan algoritma di bawah ini menghitung hasil kali dua buah bilangan bulat a (≥ 0) dan b tanpa menggunakan langsung operasi perkalian, yaitu dengan cara menjumlahkan b sebanyak a kali. Hasilnya adalah ab .

```
i ← 0
j ← 0
while i ≠ a do      (**)
    j ← j + b
    i ← i + 1
endwhile
{ i = a, j = ab }
```

Buktikan bahwa setiap kali eksekusi mencapai awal kalang *while-do* (ditandai dengan **), kita menemukan bahwa $j = i \cdot b$.

Penyelesaian:

Tabel berikut mengenumerasi nilai i dan j setiap kali eksekusi algoritma mencapai awal kalang *while-do*.

Tiap kali (n) eksekusi mencapai awal kalang <i>while-do</i>	Nilai i	Nilai j
1	0	0
2	1	$1 \cdot b$
3	2	$2 \cdot b$
4	3	$3 \cdot b$
...
$a + 1$	a	$a \cdot b$

Dari tabel di atas kita menarik kesimpulan bahwa setiap kali eksekusi algoritma mencapai awal kalang *while-do*, nilai $j = i \cdot b$. Untuk membuktikannya, kita menggunakan induksi matematik sebagai berikut: Misalkan $p(n)$ adalah proposisi bahwa setiap kali (n) eksekusi algoritma mencapai awal kalang *while-do*, nilai $j_n = i_n \cdot b$, yang dalam hal ini nilai i dan j pada eksekusi ke- n dinyatakan sebagai i_n dan j_n .

- Basis induksi:* $p(1)$ benar, karena pertama kali ($n = 1$) eksekusi mencapai awal kalang *while-do*, $i = 0$ dan $j = 0$, dan bahwa nilai $j_1 = i_1 \cdot b = 0 \cdot b = 0$ adalah benar.
- Langkah induksi:* Misalkan $p(n)$ benar, yaitu asumsikan $j_n = i_n \cdot b$ saat eksekusi mencapai awal kalang *while-do*. Kita harus menunjukkan bahwa $p(n+1)$ benar, yaitu saat eksekusi mencapai awal kalang *while-do* kali untuk yang ke- $(n+1)$ kalinya, maka $j_{n+1} = i_{n+1} \cdot b$ juga benar.

Kita dapat melihat bahwa nilai i yang baru bertambah sebesar 1 dari nilai i yang lama, dan nilai j yang baru bertambah sebesar b dari nilai j yang lama. Jadi,

$$i_{n+1} = i_n + 1$$

dan

$$\begin{aligned} j_{n+1} &= j_n + b \\ &= (i_n \cdot b) + b \quad (\text{dari hipotesis induksi}) \\ &= (i_n + 1) \cdot b \\ &= i_{n+1} \cdot b \end{aligned}$$

Karena langkah 1 dan 2 keduanya sudah diperlihatkan benar, maka terbukti setiap kali eksekusi algoritma mencapai awal kalang *while-do*, nilai $j = i \cdot b$. ■

4.3 Prinsip Induksi yang Dirampatkan

Kadang-kadang kita ingin membuktikan bahwa pernyataan $p(n)$ benar untuk semua bilangan bulat $\geq n_0$, jadi tidak hanya bilangan bulat yang dimulai dari 1 saja. Prinsip induksi sederhana dapat dirampatkan (*generalized*) untuk menunjukkan hal ini sebagai berikut:

Misalkan $p(n)$ adalah pernyataan perihal bilangan bulat dan kita ingin membuktikan bahwa $p(n)$ benar untuk semua bilangan bulat $n \geq n_0$. Untuk membuktikan ini, kita hanya perlu menunjukkan bahwa:

1. $p(n_0)$ benar, dan
2. jika $p(n)$ benar maka $p(n+1)$ benar untuk setiap $n \geq n_0$,

sehingga $p(n)$ benar untuk semua bilangan bulat $n \geq n_0$.

Contoh 4.5

Untuk semua bilangan bulat tidak-negatif n , buktikan dengan induksi matematik bahwa $2^0 + 2^1 + 2^2 + \dots + 2^n = 2^{n+1} - 1$

Penyelesaian:

Misalkan $p(n)$ adalah proposisi bahwa untuk semua bilangan bulat tidak-negatif n , $2^0 + 2^1 + 2^2 + \dots + 2^n = 2^{n+1} - 1$

- (i) *Basis induksi:* $p(0)$ benar, karena untuk $n = 0$ (bilangan bulat tidak negatif pertama), kita peroleh:

$$\begin{aligned} 2^0 &= 1 = 2^{0+1} - 1 \\ &= 2^1 - 1 \\ &= 2 - 1 \\ &= 1 \end{aligned}$$

- (ii) *Langkah induksi:* Misalkan $p(n)$ benar, yaitu proposisi

$$2^0 + 2^1 + 2^2 + \dots + 2^n = 2^{n+1} - 1$$

diasumsikan benar (hipotesis induksi). Kita harus menunjukkan bahwa $p(n + 1)$ juga benar, yaitu

$$2^0 + 2^1 + 2^2 + \dots + 2^n + 2^{n+1} = 2^{(n+1)+1} - 1$$

Hal ini kita tunjukkan sebagai berikut:

$$\begin{aligned} 2^0 + 2^1 + 2^2 + \dots + 2^n + 2^{n+1} &= (2^0 + 2^1 + 2^2 + \dots + 2^n) + 2^{n+1} \\ &= (2^{n+1} - 1) + 2^{n+1} \text{ (dari hipotesis induksi)} \\ &= (2^{n+1} + 2^{n+1}) - 1 \\ &= (2 \cdot 2^{n+1}) - 1 \\ &= 2^{n+2} - 1 \\ &= 2^{(n+1)+1} - 1 \end{aligned}$$

Karena langkah 1 dan 2 keduanya telah diperlihatkan benar, maka untuk semua bilangan bulat tidak-negatif n , terbukti bahwa $2^0 + 2^1 + 2^2 + \dots + 2^n = 2^{n+1} - 1$ ■

Contoh 4.6

Jika A_1, A_2, \dots, A_n masing-masing adalah himpunan, buktikan dengan induksi matematik hukum De Morgan rampatan berikut:

$$\overline{A_1 \cap A_2 \cap \dots \cap A_n} = \overline{A_1} \cup \overline{A_2} \cup \dots \cup \overline{A_n}$$

Penyelesaian:

Misalkan $p(n)$ adalah proposisi bahwa jika A_1, A_2, \dots, A_n masing-masing adalah himpunan, maka berlaku hukum De Morgan rampatan berikut:

$$\overline{A_1 \cap A_2 \cap \dots \cap A_n} = \overline{A_1} \cup \overline{A_2} \cup \dots \cup \overline{A_n}$$

- (i) *Basis induksi:* $p(2)$ benar, karena untuk $n = 2$,

$$\overline{A_1 \cap A_2} = \overline{A_1} \cup \overline{A_2}$$

sesuai dengan hukum De Morgan untuk 2 buah himpunan.

- (ii) *Langkah induksi:* Misalkan $p(n)$ benar, yaitu asumsikan

$$\overline{A_1 \cap A_2 \cap \dots \cap A_n} = \overline{A_1} \cup \overline{A_2} \cup \dots \cup \overline{A_n}$$

adalah benar (hipotesis induksi). Kita harus memperlihatkan bahwa untuk $p(n + 1)$ benar, yaitu

$$\overline{A_1 \cap A_2 \cap \dots \cap A_n \cap A_{n+1}} = \overline{A_1} \cup \overline{A_2} \cup \dots \cup \overline{A_n} \cup \overline{A_{n+1}}$$

Hal ini ditunjukkan sebagai berikut:

$$\begin{aligned}\overline{A_1 \cap A_2 \cap \cdots \cap A_n \cap A_{n+1}} &= \overline{(A_1 \cap A_2 \cap \cdots \cap A_n) \cap A_{n+1}} \\&= (A_1 \cap A_2 \cap \cdots \cap A_n) \cup \overline{A_{n+1}} \quad (\text{Hukum De Morgan}) \\&= \overline{(A_1 \cup A_2 \cup \cdots \cup A_n)} \cup \overline{A_{n+1}} \quad (\text{dari hipotesis induksi}) \\&= \overline{A_1} \cup \overline{A_2} \cup \cdots \cup \overline{A_n} \cup \overline{A_{n+1}} \quad (\text{Hukum Asosiatif})\end{aligned}$$

Karena langkah (i) dan (ii) sudah diperlihatkan benar, maka hukum De Morgan rampatan berikut:

$$\overline{A_1 \cap A_2 \cap \cdots \cap A_n} = \overline{A_1} \cup \overline{A_2} \cup \cdots \cup \overline{A_n}$$

sudah dibuktikan benar. ■

Contoh 4.7

Buktikan dengan induksi matematik bahwa $3^n < n!$ untuk n bilangan bulat positif yang lebih besar dari 6.

Penyelesaian:

Misalkan $p(n)$ adalah proposisi bahwa $3^n < n!$ untuk n bilangan bulat positif yang lebih besar dari 6.

- (i) *Basis induksi:* $p(7)$ benar, karena $3^7 < 7!$ sebab $3^7 = 2187$ dan $7! = 5040$
- (ii) *Langkah induksi:* Misalkan bahwa $p(n)$ benar, yaitu asumsikan bahwa $3^n < n!$ adalah benar. Kita harus menunjukkan bahwa $p(n + 1)$ juga benar, yaitu $3^{n+1} < (n+1)!$. Hal ini ditunjukkan sebagai berikut:

$$\begin{aligned}3^{n+1} &< (n+1)! \\3 \cdot 3^n &< (n+1) \cdot n! \\3^n \cdot 3/(n+1) &< n!\end{aligned}$$

Menurut hipotesis induksi, $3^n < n!$, sedangkan untuk $n > 6$, nilai $3/(n+1) < 1$, sehingga $3/(n+1)$ akan memperkecil nilai di ruas kiri persamaan. Efek nettonya, $3^n \cdot 3/(n+1) < n!$ jelas benar.

Karena langkah (i) dan (ii) sudah ditunjukkan benar, maka terbukti bahwa $3^n < n!$ untuk n bilangan bulat positif lebih besar dari 6. ■

Contoh 4.8

Buktikan dengan induksi matematik bahwa pada sebuah himpunan beranggotakan n elemen, banyaknya himpunan bagian yang dapat dibentuk dari himpunan tersebut adalah 2^n .

Penyelesaian:

Misalkan $p(n)$ adalah proposisi bahwa sebuah himpunan beranggotakan n elemen, banyaknya himpunan bagian yang dapat dibentuk dari himpunan tersebut adalah 2^n .

- (i) *Basis induksi:* $p(0)$ benar, karena untuk $n = 0$ (himpunan kosong) himpunan kosong hanya mempunyai mempunyai $2^0 = 1$ himpunan bagian, yaitu himpunan kosong itu sendiri.
- (ii) *Langkah induksi:* Andaikan bahwa $p(n)$ adalah benar, yaitu asumsikan “Banyaknya himpunan bagian dari suatu himpunan yang beranggotakan n elemen adalah 2^n ” adalah benar. Kita harus menunjukkan bahwa $p(n + 1)$ benar, yaitu jumlah himpunan bagian dari himpunan yang beranggotakan $n + 1$ elemen adalah 2^{n+1} . Hal ini ditunjukkan sebagai berikut. Misalkan elemen ke- $n+1$ adalah a . Tinjau masing-masing dari 2^n buah himpunan bagian yang sudah terbentuk. Untuk setiap himpunan bagian, buatlah himpunan baru yang anggotanya adalah seluruh anggota himpunan bagian tersebut ditambah dengan dengan tambahan satu elemen a . Karena ada 2^n buah himpunan bagian semula, maka juga akan terdapat 2^n himpunan bagian tambahan. Jumlah himpunan bagian seluruhnya adalah $2^n + 2^n = 2 \cdot 2^n = 2^{n+1}$.

Karena langkah (i) dan (ii) sudah diperlihatkan benar, maka terbukti banyaknya himpunan bagian yang dapat dibentuk dari sebuah himpunan beranggotakan n elemen adalah 2^n . ■

Contoh 4.9

Buktikan pernyataan “Untuk membayar biaya pos sebesar n sen ($n \geq 8$) selalu dapat digunakan hanya perangko 3 sen dan perangko 5 sen” benar.

Penyelesaian:

Misalkan $p(n)$ adalah proposisi bahwa untuk membayar biaya pos sebesar n sen ($n \geq 8$) selalu dapat digunakan hanya perangko 3 sen dan perangko 5 sen.

- (i) *Basis induksi:* $p(8)$ benar, karena untuk membayar biaya pos 8 sen dapat digunakan 1 buah perangko 3 sen dan 1 buah perangko 5 sen saja.
- (ii) *Langkah induksi:* Andaikan bahwa $p(n)$ benar, yaitu asumsikan bahwa untuk membayar biaya pos sebesar n sen dapat digunakan perangko 3 sen dan 5 sen (hipotesis induksi). Kita harus menunjukkan bahwa $p(n + 1)$ benar, yaitu untuk membayar biaya pos sebesar $n + 1$ sen juga dapat menggunakan perangko 3 sen dan perangko 5 sen. Ada dua kemungkinan yang perlu diperiksa. Kemungkinan pertama, misalkan kita membayar biaya pos senilai n sen dengan sedikitnya satu perangko 5 sen. Dengan mengganti satu buah perangko 5 sen dengan dua buah perangko 3 sen, akan diperoleh susunan perangko senilai $n + 1$ sen. Kemungkinan kedua, jika tidak ada perangko 5 sen yang digunakan, biaya pos senilai n sen menggunakan perangko 3 sen semuanya. Karena $n \geq 8$, setidaknya harus digunakan tiga buah perangko 3 sen. Dengan mengganti tiga buah perangko 3 sen dengan 2 buah perangko 5 sen, akan dihasilkan nilai perangko $n + 1$ sen.

Karena langkah (i) dan (ii) sudah ditunjukkan benar, maka pernyataan “Untuk membayar biaya pos sebesar n sen ($n \geq 8$) selalu dapat digunakan hanya perangko 3 sen dan perangko 5 sen” terbukti benar. ■

Contoh 4.10

Sebuah ATM (Anjungan Tunai Mandiri) hanya menyediakan pecahan uang Rp 20.000,- dan Rp 50.000,-. Kelipatan uang berapakah yang dapat dikeluarkan oleh ATM tersebut? Buktikan jawaban anda dengan induksi matematik.

Penyelesaian:

Dengan pecahan uang Rp 20.000,-, ATM dapat mengeluarkan uang untuk penarikan Rp 20.000, Rp 40.000,-, Rp 60.000,-, ..., sedangkan dengan pecahan uang Rp 50.000,-, ATM dapat mengeluarkan uang untuk penarikan Rp 50.000, Rp 100.000,-, Dari kedua kombinasi pecahan uang tersebut kita dapat menyimpulkan bahwa ATM dapat mengeluarkan uang kelipatan Rp 10.000,- atau dengan kata lain mengeluarkan uang senilai $10.000n$ rupiah untuk $n \geq 4$ (*catatan:* perhatikanlah bahwa kita tidak dapat menggunakan basis $n = 2$ sebab ia tidak dapat digunakan pada langkah induksi).

Misalkan $p(n)$ adalah proposisi bahwa ATM dapat mengeluarkan uang senilai $10.000n$ rupiah untuk $n \geq 4$ dengan pecahan Rp 20.000,- dan Rp 50.000,-. Kita akan membuktikan $p(n)$ dengan induksi matematik.

- (i) *Basis induksi:* $p(4)$ benar, karena ATM dapat mengeluarkan uang senilai Rp 40.000 dengan 2 buah pecahan Rp 20.000,-.
- (ii) *Langkah induksi:* Andaikan $p(n)$ benar, yaitu asumsikan bahwa ATM dapat mengeluarkan uang senilai $10.000n$ rupiah dengan pecahan Rp 20.000,- dan Rp 50.000,-. Kita harus menunjukkan bahwa $p(n + 1)$ benar, yaitu ATM juga dapat mengeluarkan uang senilai $10.000(n + 1)$ rupiah dengan menggunakan pecahan Rp 20.000,- dan Rp 50.000,-. Ada dua kemungkinan yang harus kita tinjau:
 - 1) Jika untuk uang senilai $10.000n$ rupiah ATM menggunakan minimal 1 buah pecahan Rp 50.000,-, maka dengan mengganti 1 pecahan Rp 50.000,- dengan 3 buah pecahan Rp 20.000, maka ATM selalu dapat mengeluarkan uang senilai $10.000(n + 1)$ rupiah.
 - 2) Jika untuk uang senilai $10.000n$ rupiah ATM menggunakan pecahan Rp 20.000,-, maka paling sedikit digunakan 2 buah pecahan Rp 20.000,- (sebab $n \geq 4$). Dengan mengganti 2 buah pecahan Rp 20.000,- dengan 1 buah pecahan Rp 50.000,-, maka ATM selalu dapat mengeluarkan uang senilai $10.000(n + 1)$ rupiah.

Karena langkah (i) dan (ii) sudah diperlihatkan benar, maka terbukti bahwa ATM dapat mengeluarkan uang senilai $10.000n$ rupiah untuk $n \geq 4$ dengan pecahan Rp 20.000,- dan Rp 50.000,-. ■

4.4 Prinsip Induksi Kuat

Kadang-kadang versi induksi yang lebih kuat diperlukan untuk membuktikan pernyataan mengenai bilangan bulat. Versi induksi yang lebih kuat adalah sebagai berikut:

Misalkan $p(n)$ adalah pernyataan perihal bilangan bulat dan kita ingin membuktikan bahwa $p(n)$ benar untuk semua bilangan bulat $n \geq n_0$. Untuk membuktikan ini, kita hanya perlu menunjukkan bahwa:

1. $p(n_0)$ benar, dan
2. jika $p(n_0)$, $p(n_0+1)$, ..., $p(n)$ benar, maka $p(n+1)$ juga benar untuk setiap bilangan bulat $n \geq n_0$,

sehingga $p(n)$ benar untuk semua bilangan bulat $n \geq n_0$.

Catatlah bahwa versi induksi yang lebih kuat ini mirip dengan induksi sederhana, kecuali bahwa pada langkah 2 kita mengambil hipotesis induksi yang lebih kuat bahwa semua pernyataan $p(1)$, $p(2)$, ..., $p(n)$ adalah benar daripada hipotesis yang menyatakan bahwa $p(n)$ benar (pada induksi sederhana). Prinsip induksi kuat memungkinkan kita mencapai kesimpulan yang sama meskipun memberlakukan andaian yang lebih banyak.

Dua contoh berikut memperlihatkan penggunaan prinsip induksi kuat.

Contoh 4.11

Bilangan bulat positif disebut prima jika dan hanya jika bilangan bulat tersebut habis dibagi dengan 1 dan dirinya sendiri. Kita ingin membuktikan bahwa setiap bilangan bulat positif n ($n \geq 2$) dapat dinyatakan sebagai perkalian dari (satu atau lebih) bilangan prima. Buktikan dengan prinsip induksi kuat.

Penyelesaian:

Mislakan $p(n)$ adalah proposisi bahwa setiap bilangan bulat positif n ($n \geq 2$) dapat dinyatakan sebagai perkalian dari (satu atau lebih) bilangan prima.

- (i) *Basis induksi:* $p(2)$ benar, karena 2 sendiri adalah bilangan prima dan di sini 2 dapat dinyatakan sebagai perkalian dari satu buah bilangan prima, yaitu dirinya sendiri.
- (ii) *Langkah induksi.* Misalkan $p(n)$ benar, yaitu asumsikan bahwa bilangan $2, 3, \dots, n$ dapat dinyatakan sebagai perkalian (satu atau lebih) bilangan prima (hipotesis induksi). Kita perlu menunjukkan bahwa $p(n + 1)$ benar, yaitu $n + 1$ juga dapat dinyatakan sebagai perkalian bilangan prima. Hal ini ditunjukkan sebagai berikut: jika $n + 1$ sendiri bilangan prima, maka jelas ia dapat dinyatakan sebagai perkalian satu atau lebih bilangan prima. Jika $n + 1$ bukan bilangan prima, maka terdapat bilangan bulat positif a yang membagi habis $n + 1$ tanpa sisa. Dengan kata lain,

$$(n + 1)/a = b \text{ atau } (n + 1) = ab$$

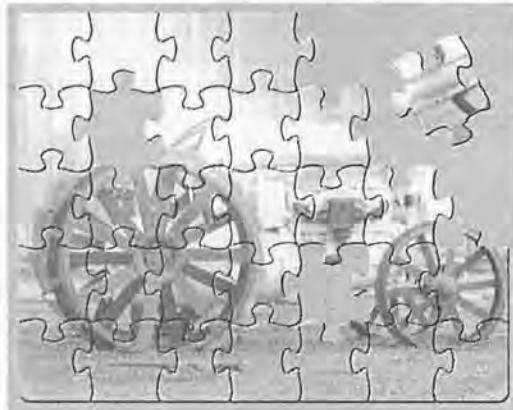
yang dalam hal ini, $2 \leq a \leq b \leq n$. Menurut hipotesis induksi, a dan b dapat dinyatakan sebagai perkalian satu atau lebih bilangan prima. Ini berarti, $n + 1$ jelas dapat dinyatakan sebagai perkalian bilangan prima, karena $n + 1 = ab$.

Karena langkah (i) dan (ii) sudah ditunjukkan benar, maka terbukti bahwa setiap bilangan bulat positif n ($n \geq 2$) dapat dinyatakan sebagai perkalian dari (satu atau lebih) bilangan prima.

Catatlah bahwa pernyataan di atas lebih tepat dibuktikan dengan prinsip induksi kuat daripada dengan prinsip induksi sederhana. Kita tahu bahwa a dan b keduanya $\leq n$, karena itu, untuk dapat menerapkan hipotesis induksi terhadap keduanya, kita perlu mengetahui bahwa tiap bilangan bulat positif $2, 3, \dots, n$ dapat dinyatakan sebagai perkalian bilangan prima. Mengandaikan bahwa n dapat dinyatakan sebagai perkalian bilangan prima saja tidaklah cukup.

Contoh 4.12

[LIU85] Teka-teki susun potongan gambar (*jigsaw puzzle*) terdiri dari sejumlah potongan (bagian) gambar (lihat Gambar 4.2). Dua atau lebih potongan dapat disatukan untuk membentuk potongan yang lebih besar. Lebih tepatnya, kita gunakan istilah blok bagi satu potongan gambar. Blok-blok dengan batas yang cocok dapat disatukan membentuk blok yang lain yang lebih besar. Akhirnya, jika semua potongan telah disatukan menjadi satu buah blok, teka-teki susun gambar itu dikatakan telah dipecahkan. Menggabungkan dua buah blok dengan batas yang cocok dihitung sebagai satu langkah. Gunakan prinsip induksi kuat untuk membuktikan bahwa untuk suatu teka-teki susun gambar dengan n potongan, selalu diperlukan $n - 1$ langkah untuk memecahkan teki-teki itu.



Gambar 4.2 Teka-teki susun potongan gambar

Penyelesaian:

Misalkan $p(n)$ adalah proposisi bahwa suatu teka-teki susun gambar dengan n potongan, selalu diperlukan $n - 1$ langkah untuk memecahkan teki-teki itu.

- (i) *Basis induksi:* $p(1)$ benar, karena untuk teka-teki susun gambar dengan satu potongan, tidak diperlukan langkah apa-apa untuk memecahkan teka-teki itu.
- (ii) *Langkah induksi:* Misalkan $p(n)$ benar, yaitu asumsikan bahwa untuk teka-teki dengan n potongan ($n = 1, 2, 3, \dots, k$) diperlukan sejumlah $n - 1$ langkah untuk memecahkan teka-teki itu (hipotesis induksi). Kita harus membuktikan bahwa $p(n + 1)$

benar, yaitu untuk $n + 1$ potongan gambar diperlukan n langkah. Hal ini ditunjukkan sebagai berikut:

Bagilah $n + 1$ potongan menjadi dua buah blok-satu dengan n_1 potongan dan satu lagi dengan n_2 potongan, dan $n_1 + n_2 = n + 1$. Untuk langkah terakhir yang memecahkan teka-teki ini, dua buah blok disatukan sehingga membentuk satu blok besar. Menurut hipotesis induksi, diperlukan $n_1 - 1$ langkah untuk menyatukan blok yang satu dan $n_2 - 1$ langkah untuk menyatukan blok yang lain. Digabungkan dengan langkah terakhir yang menyatukan kedua blok tersebut, maka banyaknya langkah adalah

$$(n_1 - 1) + (n_2 - 1) + 1 \text{ langkah terakhir} = (n_1 + n_2) - 2 + 1 = n + 1 - 1 = n.$$

Karena langkah (i) dan (ii) sudah diperlihatkan benar maka terbukti bahwa suatu teka-teki susun gambar dengan n potongan, selalu diperlukan $n - 1$ langkah untuk memecahkan teka-teki itu. ■

4.5 Bentuk Induksi Secara Umum

Adalah mungkin membuat bentuk umum metode induksi sehingga ia dapat diterapkan tidak hanya untuk pembuktian proposisi yang menyangkut himpunan bilangan bulat positif, tetapi juga pembuktian yang menyangkut himpunan obyek yang lebih umum. Syaratnya, himpunan obyek tersebut harus mempunyai keterurutan dan mempunyai elemen terkecil.

DEFINISI 4.1. Relasi biner " $<$ " pada himpunan X dikatakan terurut dengan baik (atau himpunan X dikatakan terurut dengan baik dengan " $<$ ") bila memiliki properti berikut:

- (i) Diberikan $x, y, z \in X$, jika $x < y$ dan $y < z$, maka $x < z$.
- (ii) Diberikan $x, y \in X$. Salah satu dari kemungkinan ini benar: $x < y$ atau $y < x$ atau $x = y$.
- (iii) Jika A adalah himpunan bagian tidak kosong dari X , terdapat elemen $x \in A$ sedemikian sehingga $x \leq y$ untuk semua $y \in A$. Dengan kata lain, setiap himpunan bagian tidak kosong dari X mengandung "elemen terkecil".

Himpunan bilangan riil tak-negatif tidak terurut dengan baik oleh relasi " $<$ ". Himpunan ini mempunyai properti (i) dan (ii) tetapi tidak (iii). Sebagai contoh, himpunan semua bilangan riil yang lebih besar dari 1, yaitu $\{x \mid x \text{ adalah bilangan riil dan } x > 1\}$, tidak mengandung elemen terkecil.

Himpunan pasangan terurut bilangan bulat tidak negatif terurut dengan baik oleh relasi " $<$ ", dengan kata lain " $<$ " didefinisikan oleh $(n_1, n_2) < (n_3, n_4)$ jika dan hanya jika $(n_1 < n_3)$ atau $(n_1 = n_3 \text{ dan } n_2 < n_4)$. Properti (i), (ii), dan (iii) dimiliki oleh himpunan ini.

Bentuk induksi secara umum dapat dituliskan sebagai berikut:

Misalkan X terurut dengan baik oleh " $<$ ", dan $p(x)$ adalah pernyataan perihal elemen x dari X . Kita ingin membuktikan bahwa $p(x)$ benar untuk semua $x \in X$. Untuk membuktikan ini, kita hanya perlu menunjukkan bahwa:

1. $p(x_0)$ benar, yang dalam hal ini x_0 adalah elemen terkecil di dalam X , dan
 2. jika $p(y)$ benar untuk $y < x$, maka $p(x)$ juga benar untuk setiap $x > x_0$ di dalam X ,
- sehingga $p(x)$ benar untuk semua $x \in X$.
-

Contoh 4.13

Tinjau barisan bilangan yang didefinisikan sebagai berikut:

$$S_{m,n} = \begin{cases} 0 & \text{jika } m=0 \text{ dan } n=0 \\ S_{m-1,n} + 1 & \text{jika } n=0 \\ S_{m,n-1} + 1 & \text{jika } n \neq 0 \end{cases}$$

Sebagai contoh,

$$\begin{array}{ll} S_{0,0} = 0 & S_{1,0} = S_{0,0} + 1 = 0 + 1 = 1 \\ S_{0,1} = S_{0,0} + 1 = 1 & S_{1,1} = S_{1,0} + 1 = 1 + 1 = 2 \\ S_{2,0} = S_{1,0} + 1 = 2 & S_{2,1} = S_{2,0} + 1 = 3, \dots \end{array}$$

Buktikanlah dengan induksi matematik bahwa untuk pasangan tidak negatif m dan n , $S_{m,n} = m + n$.

Penyelesaian:

- (i) *Basis induksi:* Karena $(0, 0)$ adalah elemen terkecil di dalam X , maka $S_{0,0} = 0 + 0 = 0$. Ini benar dari definisi $S_{0,0}$.
- (ii) *Langkah induksi.* Buktikan untuk semua $(m, n) > (0, 0)$ di dalam X bahwa jika $S_{m',n'} = m' + n'$ benar untuk semua $(m', n') < (m, n)$ maka $S_{m,n} = m + n$ juga benar. Andaikan bahwa $S_{m',n'} = m' + n'$ benar untuk semua $(m', n') < (m, n)$. Ini adalah hipotesis induksi. Kita perlu menunjukkan bahwa $S_{m,n} = m + n$, baik untuk $n = 0$ atau $n \neq 0$.

Kasus 1: Jika $n = 0$, maka dari definisi $S_{m,n} = S_{m-1,n} + 1$. Karena $(m-1, n) < (m, n)$, maka dari hipotesis induksi,

$$S_{m-1,n} = (m-1) + n \text{ sehingga } S_{m,n} = S_{m-1,n} + 1 = (m-1) + n + 1 = m + n.$$

Kasus 2: Jika $n \neq 0$, maka dari definisi $S_{m,n} = S_{m,n-1} + 1$. Karena $(m, n-1) < (m, n)$, maka dari hipotesis induksi,

$$S_{m,n-1} = m + (n-1) \text{ sehingga } S_{m,n} = S_{m,n-1} + 1 = m + (n-1) + 1 = m + n.$$

Karena langkah (1) dan (ii) sudah diperlihatkan benar, maka terbukti bahwa untuk pasangan tidak negatif m dan n , $S_{m,n} = m + n$. ■

4.6 Ragam Soal dan Penyelesaian

Contoh 4.14

Temukan rumus untuk menghitung $1/2 + 1/4 + 1/8 + \dots + 1/2^n$ dengan memeriksa nilai-nilai ekspresi untuk n yang kecil, lalu gunakan induksi matematik untuk membuktikan rumus itu.

Penyelesaian:

Rumus ditentukan secara empirik dengan mencoba menghitung deret untuk n yang kecil,

$$n = 1 \rightarrow 1/2 = (2^1 - 1)/2^1$$

$$n = 2 \rightarrow 1/2 + 1/4 = 3/4 = (2^2 - 1)/2^2$$

$$n = 3 \rightarrow 1/2 + 1/4 + 1/8 = 9/8 = (2^3 - 1)/2^3$$

...

$$n = k \rightarrow 1/2 + 1/4 + 1/8 + \dots + 1/2^k = (2^k - 1)/2^k$$

Jadi, dapat disimpulkan (untuk sementara) bahwa

$$1/2 + 1/4 + 1/8 + \dots + 1/2^n = (2^n - 1)/2^n, n \geq 1$$

Misalkan $p(n)$ adalah proposisi bahwa untuk $n \geq 1$, $1/2 + 1/4 + 1/8 + \dots + 1/2^n = (2^n - 1)/2^n$. Kita akan buktikan kebenaran $p(n)$ dengan induksi matematik:

(i) *Basis induksi:* $p(1)$ benar, karena

$$1/2 = (2^1 - 1)/2^1 = 1/2$$

(ii) *Langkah induksi:* Andaikan bahwa $p(n)$ benar, yaitu asumsikan bahwa jumlah deret $1/2 + 1/4 + 1/8 + \dots + 1/2^n = (2^n - 1)/2^n$ (hipotesis induksi). Kita harus memperlihatkan bahwa $p(n+1)$ benar, yaitu untuk $n+1$ jumlah deret tersebut adalah

$$1/2 + 1/4 + 1/8 + \dots + 1/2^n + 1/2^{n+1} = (2^{n+1} - 1)/2^{n+1}$$

Hal ini ditunjukkan sebagai berikut:

$$\begin{aligned} 1/2 + 1/4 + 1/8 + \dots + 1/2^n + 1/2^{n+1} &= (1/2 + 1/4 + 1/8 + \dots + 1/2^n) + 1/2^{n+1} \\ &= (2^n - 1)/2^n + 1/2^{n+1} \\ &= 2(2^n - 1)/2 \cdot 2^n + 1/2^{n+1} \\ &= 2(2^n - 1)/2^{n+1} + 1/2^{n+1} \\ &= (2^{n+1} - 2 + 1)/2^{n+1} \\ &= (2^{n+1} - 1)/2^{n+1} \end{aligned}$$

Karena langkah (i) dan (ii) sudah diperlihatkan benar, maka terbukti bahwa

$$1/2 + 1/4 + 1/8 + \dots + 1/2^n = (2^n - 1)/2^n$$

Contoh 4.15

Buktikan dengan induksi matematik bahwa $n^5 - n$ habis dibagi 5 untuk n bilangan bulat positif.

Penyelesaian:

Andaikan bahwa $p(n)$ adalah proposisi bahwa $n^5 - n$ habis dibagi 5 untuk n bilangan bulat positif.

- (i) *Basis induksi:* $p(1)$ benar, karena $1^5 - 1 = 0$ habis dibagi 5.
- (ii) *Langkah induksi:* Andaikan bahwa $p(n)$ benar, yaitu asumsikan bahwa $n^5 - n$ habis dibagi 5 untuk $n > 0$ (hipotesis induksi). Kita harus memperlihatkan bahwa $p(n + 1)$ benar, yaitu $(n+1)^5 - (n+1)$ juga habis dibagi 5. Ini ditunjukkan sebagai berikut:

$$\begin{aligned}(n+1)^5 - (n+1) &= n^5 + 5n^4 + 10n^3 + 10n^2 + 5n + 1 - n - 1 \\&= n^5 - n + 5n^4 + 10n^3 + 10n^2 + 5n \\&= (n^5 - n) + 5(n^4 + 2n^3 + 5n^2 + n)\end{aligned}$$

Menurut hipotesis induksi, $(n^5 - n)$ habis dibagi 5 dan $5(n^4 + 2n^3 + 5n^2 + n)$ jelas juga habis dibagi 5, sehingga $(n^5 - n) + 5(n^4 + 2n^3 + 5n^2 + n)$ habis dibagi 5. Dengan demikian, terbukti $(n+1)^5 - (n+1)$ habis dibagi 5.

Karena langkah (i) dan (ii) sudah diperlihatkan benar, maka terbukti bahwa $n^5 - n$ habis dibagi 5 untuk n bilangan bulat positif. ■

Contoh 4.16

Untuk biaya pos berapa saja yang dapat menggunakan perangko senilai 5 sen dan 6 sen? Buktikan jawaban anda dengan induksi matematik.

Penyelesaian:

Kombinasi biaya pos dengan perangko 5 sen dan 6 sen dapat ditulis sebagai $5m + 6n$, dengan m dan n adalah bilangan bulat. Dengan mencoba kombinasi nilai m dan n mulai dari 0, 1, 2, 3, 4, ..., maka diperoleh biaya pos yang dapat dibayar mulai 20 sen, 21 sen, 22 sen dan seterusnya.

Akan kita buktikan dengan induksi matematika bahwa untuk biaya pos sebesar $n \geq 20$ sen selalu dapat menggunakan perangko 5 sen dan 6 sen.

Misalkan $p(n)$ adalah proposisi bahwa untuk biaya pos sebesar $n \geq 20$ sen selalu dapat menggunakan perangko 5 sen dan 6 sen.

- (i) *Basis induksi:* $p(20)$ benar, karena untuk biaya pos sebesar 20 sen, kita dapat menggunakan 4 perangko 5 sen saja.
- (ii) *Langkah induksi.* Andaikan $p(n)$ benar, yaitu asumsikan biaya pos sebesar $n \geq 20$ sen selalu dapat menggunakan perangko 5 sen dan 6 sen (hipotesis induksi). Kita harus menunjukkan bahwa $p(n + 1)$ benar, yaitu biaya pos sebesar $n + 1$ sen juga dapat

menggunakan perangko 5 sen dan 6 sen saja. Ada dua kemungkinan yang harus kita tinjau:

- 1) Jika untuk membayar biaya pos n sen digunakan perangko 5 sen saja, maka paling sedikit digunakan 4 buah perangko 5 sen (sebab $n \geq 20$), maka dengan mengganti sebuah perangko 5 sen dengan 6 sen selalu dapat dibayar biaya pos sebesar $n + 1$ sen.
- 2) Jika untuk membayar biaya pos n sen digunakan perangko 6 sen, maka paling sedikit digunakan 4 buah perangko 6 sen (sebab $n \geq 20$). Dengan mengganti 4 buah perangko 6 sen dengan 5 buah, diperoleh susunan perangko senilai $n + 1$ sen.

Karena langkah (i) dan (ii) sudah diperlihatkan benar, maka terbukti bahwa biaya pos sebesar $n \geq 20$ sen selalu dapat menggunakan perangko 5 sen dan 6 sen. ■

Contoh 4.17

Di dalam sebuah pesta, setiap tamu berjabat tangan dengan tamu lainnya hanya sekali saja. Buktikan dengan induksi matematik bahwa jika ada n orang tamu maka jumlah jabat tangan yang terjadi adalah $n(n - 1)/2$.

Penyelesaian:

Nilai n minimal 1. Misalkan $p(n)$ adalah proposisi bahwa jika ada n orang tamu maka jumlah jabat tangan yang terjadi adalah $n(n - 1)/2$.

- (i) *Basis Induksi:* $p(1)$ benar, karena untuk $n = 1$ orang tamu, tidak ada jabat tangan yang terjadi, atau $1(1 - 1)/2 = 0$ kali.
- (ii) *Langkah induksi:* Misalkan $p(n)$ benar, yaitu asumsikan jumlah jabat tangan yang terjadi sebanyak $n(n - 1)/2$ (hipotesis induksi). Kita harus menunjukkan bahwa $p(n + 1)$ benar, yaitu jumlah jabat tangan yang terjadi di antara $n + 1$ orang tamu adalah $(n+1)((n + 1) - 1)/2$ atau $n(n+1)/2$.

Hal ini dapat ditunjukkan sebagai berikut:

Untuk $n + 1$ orang, jumlah jabat tangan yang terjadi haruslah berupa jumlah jabat tangan n orang tamu ditambah jabat tangan yang dilakukan tamu ke- $(n + 1)$. Menurut hipotesis induksi, untuk n orang tamu, jumlah jabat tangan yang terjadi adalah $n(n - 1)/2$. Tamu yang ke- $(n + 1)$ ini akan berjabat tangan sebanyak n kali dengan n orang tamu lainnya (masing-masing sekali) sehingga jumlah jabat tangan keseluruhan adalah

$$\begin{aligned} n(n - 1)/2 + n &= n(n - 1)/2 + 2n/2 \\ &= (n^2 - n + 2n)/2 \\ &= (n^2 + n)/2 \\ &= n(n + 1)/2 \end{aligned}$$

Karena langkah basis dan langkah induksi keduanya telah ditunjukkan benar, maka terbukti bahwa jika ada n orang tamu, maka jumlah jabat tangan yang terjadi adalah $n(n - 1)/2$. ■

Contoh 4.18

Gunakan induksi matematik untuk membuktikan bahwa suatu himpunan dengan n elemen ($n \geq 2$) mempunyai $n(n - 1)/2$ himpunan bagian yang mengandung tepat 2 elemen.

Penyelesaian:

Misalkan $p(n)$ adalah proposisi bahwa suatu himpunan dengan n elemen ($n \geq 2$) mempunyai $n(n - 1)/2$ himpunan bagian yang mengandung tepat 2 elemen.

- (i) *Basis induksi:* $p(2)$ benar, karena suatu himpunan dengan anggota 2 elemen memiliki $2(2 - 1)/2 = 1$ himpunan yang mengandung tepat 2 elemen.
- (ii) *Langkah induksi:* Misalkan $p(n)$ benar, yaitu suatu himpunan dengan n elemen mempunyai $n(n - 1)/2$ himpunan bagian yang mengandung tepat 2 elemen (hipotesis induksi). Kita harus menunjukkan bahwa $p(n + 1)$ benar, yaitu suatu himpunan dengan $n + 1$ elemen mempunyai $(n + 1)((n + 1) - 1)/2$ himpunan bagian yang mengandung tepat 2 elemen. Hal ini ditunjukkan sebagai berikut: Misalkan S adalah himpunan beranggotakan n elemen. Misalkan ditambahkan elemen ke- $(n + 1)$, yaitu a , ke dalam S menjadi himpunan baru, T , yang dalam hal ini $T = S \cup \{a\}$. Himpunan bagian 2-elemen dari S tidak mengandung a , dan menurut hipotesis induksi jumlahnya adalah $n(n - 1)/2$. Himpunan bagian dari T dibentuk dengan cara berikut: untuk setiap himpunan bagian dari S , bentuklah himpunan baru yang anggotanya adalah seluruh anggota himpunan bagian tersebut ditambah dengan satu elemen a . Tinjau hanya himpunan bagian 1-elemen dari S yang banyaknya n buah. Dengan penambahan satu elemen a , maka anggota himpunan bagian tersebut menjadi 2 buah elemen. Dengan demikian, jumlah himpunan bagian 2-elemen dari T adalah himpunan bagian 2-elemen yang tidak mengandung a dan himpunan bagian 2-elemen yang mengandung a . Jumlah seluruh himpunan bagian 2-elemen dari T adalah

$$\begin{aligned} n(n - 1)/2 + n &= n(n - 1)/2 + 2n/2 \\ &= (n^2 - n)/2 + 2n/2 \\ &= (n^2 - n + 2n)/2 \\ &= (n^2 + n)/2 \\ &= (n + 1)n/2 \\ &= (n + 1)((n + 1) - 1)/2 \end{aligned}$$

Karena langkah basis dan langkah induksi keduanya telah ditunjukkan benar, maka terbukti bahwa suatu himpunan dengan n elemen ($n \geq 2$) mempunyai $n(n - 1)/2$ himpunan bagian yang mengandung tepat 2 elemen. ■

Contoh 4.19

Tunjukkan apa yang salah dari pembuktian di bawah ini yang menyimpulkan bahwa semua kuda berwarna sama?

Misalkan $p(n)$ adalah pernyataan bahwa semua kuda di dalam sebuah himpunan berwarna sama.

- (i) *Basis induksi:* jika kuda di dalam himpunan hanya seekor, jelaslah $p(1)$ benar.

- (ii) *Langkah induksi:* Misalkan $p(n)$ benar, yaitu asumsikan bahwa semua kuda di dalam himpunan n ekor kuda berwarna sama. Tinjau untuk himpunan dengan $n + 1$ kuda; nomori kuda-kuda tersebut dengan $1, 2, 3, \dots, n, n+1$. Tinjau dua himpunan, yaitu n ekor kuda yang pertama $(1, 2, \dots, n)$ harus berwarna sama, dan n ekor kuda yang terakhir $(2, 3, \dots, n, n+1)$ juga harus berwarna sama. Karena himpunan n kuda pertama dan himpunan n kuda terakhir beririsan, maka semua $n+1$ kuda harus berwarna sama. Ini membuktikan bahwa $P(n+1)$ benar.

Penyelesaian:

langkah induksi tidak benar jika $n + 1 = 2$, sebab dua himpunan (yang masing-masing beranggotakan $n = 1$ elemen) tidak beririsan. ■

Soal Latihan

1. Buktikan melalui induksi matematik bahawa

(a) $1(2) + 2(3) + \dots + n(n+1) = \frac{n(n+1)(n+2)}{3}$ untuk semua $n \geq 1$.

(b) $\frac{1}{1(2)} + \frac{1}{2(3)} + \frac{1}{3(4)} + \dots + \frac{1}{n(n+1)} = \frac{n}{n+1}$ untuk semua $n \geq 1$.

(c) $1^2 + 3^2 + 5^2 + \dots + (2n-1)^2 = \frac{n(2n-1)(2n+1)}{3}$ untuk semua $n \geq 1$.

(d) $1 + a + a^2 + \dots + a^n = \frac{1-a^{n+1}}{1-a}$ untuk semua $n \geq 0$ dan $a \neq 1$.

(e) $3 + 3 \cdot 5 + 3 \cdot 5^2 + \dots + 3 \cdot 5^n = 3(5^{n+1} - 1)/4$ untuk semua $n \geq 0$.

2. Buktikan melalui induksi matematik bahawa $n^4 - 4n^2$ habis dibagi 3 untuk semua bilangan bulat $n \geq 2$.

3. Carilah dan kemudian buktikan melalui induksi matematik suatu rumus umum berdasarkan pengamatan berikut:

$$1^3 = 1$$

$$2^3 = 3 + 5$$

$$3^3 = 7 + 9 + 11$$

$$4^3 = 13 + 15 + 17 + 19$$

4. Buktikan dengan induksi matematik bahawa jumlah pangkat tiga dari tiga buah bilangan bulat positif berurutan selalu habis dibagi 9.

5. Ketika n pasangan tamu tiba di pesta, mereka disambut oleh tuan dan nyonya rumah di pintu. Setelah saling berjabat tangan, tuan rumah bertanya kepada para tamu maupun istrinya untuk mengatakan berapa kali mereka masing-masing telah berjabat tangan. Ia memperoleh $2n + 1$ jawaban yang berbeda. Jika tidak seorang pun berjabat tangan dengan istri atau suaminya sendiri, berapa kalikah nyonya rumah telah berjabat tangan? Buktikan jawaban Anda dengan induksi matematik.

6. Buktikan bahwa surat pos yang menggunakan perangko 24 sen atau lebih dapat hanya menggunakan perangko 5 sen atau 7 sen.

7. Untuk biaya pos berapa saja dapat menggunakan perangko 5 sen dan perangko 6 sen? Buktikan jawaban anda dengan menggunakan induksi matematik!

8. Sebuah kios penukaran uang hanya mempunyai pecahan uang senilai Rp2000 dan Rp5000. Untuk uang senilai berapa saja yang dapat ditukar dengan kedua pecahan tersebut? Buktikan jawaban anda dengan induksi matematik.
9. Suatu *string* biner panjangnya n bit. Jumlah *string* biner yang mempunyai bit 1 sejumlah genap adalah 2^{n-1} . Buktikan pernyataan tersebut untuk $n \geq 1$.
10. Misalkan bahwa $A = \begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix}$. Buktikan dengan induksi matematik bahwa
- $$A^n = \begin{bmatrix} a^n & 0 \\ 0 & b^n \end{bmatrix}$$
11. Perlihatkan bahwa $[(p_1 \rightarrow p_2) \wedge (p_2 \rightarrow p_3) \wedge \dots \wedge (p_{n-1} \rightarrow p_n)] \rightarrow [(p_1 \wedge p_2 \wedge \dots \wedge p_{n-1}) \rightarrow p_n]$ adalah tautologi bilamana p_1, p_2, \dots, p_n adalah proposisi.
12. (Kalkulus diferensial) Gunakan induksi matematik untuk membuktikan bahwa turunan $f(x) = x^n$ sama dengan nx^{n-1} .
13. Gunakan induksi matematik untuk menunjukkan bahwa n garis lurus pada sebuah bidang membagi bidang tersebut menjadi $(n^2 + n + 2)/2$ daerah. Asumsikan bahwa tidak ada dua garis lurus yang sejajar dan tidak ada tiga garis lurus yang berpotongan pada satu titik persekutuan.
14. Buktiakan dengan induksi matematik bahwa jika A, B_1, B_2, \dots, B_n adalah himpunan, $n \geq 2$, maka
- $$A \cap (B_1 \cup B_2 \cup \dots \cup B_n) = (A \cap B_1) \cup (A \cap B_2) \cup \dots \cup (A \cap B_n)$$

15. Gunakan induksi matematik untuk membuktikan bahwa suatu himpunan dengan n buah elemen ($n \geq 3$) mempunyai $n(n - 1)(n - 2)/6$ himpunan bagian yang mengandung tepat 3 buah elemen.
16. Temukan kesalahan dalam pembuktian berikut. Kita ingin membuktikan bahwa semua marmut di dalam kandang berwarna sama. Misalkan $p(n)$ adalah pernyataan bahwa semua marmut di dalam kandang yang berkapasitas n ekor berwarna sama.
- (i) *Basis induksi.* Untuk $n = 1$ jelas benar karena hanya ada satu ekor marmut pastilah warnanya sama.
- (ii) *Langkah induksi.* Andaikan pernyataan tersebut benar untuk kandang yang berisi n ekor marmut. Kita harus memperlihatkan bahwa pernyataan tersebut juga benar untuk kandang yang berisi $n+1$ ekor. Misalkan kita gambar koleksi $n+1$ ekor marmut dan diberi nomor 1, 2, ..., $n+1$ sebagai berikut:

$$\begin{array}{ccccccccc} \bullet & \bullet & \dots & \bullet & \bullet \\ 1 & 2 & & n & n+1 \end{array}$$

Jika kita pindahkan marmut ke- $(n+1)$ dari kandang, maka kandang berisi n ekor marmut dengan susunan seperti berikut:

$$\begin{array}{ccccccc} & \bullet & \bullet & \dots & \bullet & & \\ 1 & & 2 & & n & & \end{array}$$

Menurut hipotesis induksi, n ekor marmut di dalam kandang tersebut berwarna sama. Jika yang kita pindahkan adalah marmut pertama, maka kandang juga berisi n ekor marmut dengan susunan seperti berikut:

$$\begin{array}{ccccccc} & \bullet & \dots & \bullet & \bullet & & \\ 2 & & n & & n+1 & & \end{array}$$

Menurut hipotesis induksi, n ekor marmut di dalam kandang tersebut juga berwarna sama. Hal ini mengimplikasikan bahwa semua $n+1$ marmut berwarna sama, karena kita tahu bahwa marmut

$$\begin{array}{ccccccc} & \bullet & \bullet & \dots & \bullet & & \\ 1 & & 2 & & n & & \end{array}$$

berwarna sama dan marmut ke- $(n+1)$ juga berwarna sama seperti marmut ke- n (faktanya, ia tidak hanya sama sewarna dengan marmut ke- n : ia juga sewarna dengan marmut 2, 3, ..., n). Jadi marmut $n+1$ berwarna sama dengan marmut yang lain.

17. Apa yang salah dari pembuktian dengan induksi matematik untuk proposisi di bawah ini, yang secara praduga menunjukkan bahwa dua bilangan bulat positif sembarang adalah sama?

Proposisi:

Jika a dan b masing-masing adalah bilangan bulat positif dan $n = \max\{a, b\}$ maka $a = b$.

(i) *Basis induksi.* Jika a dan b masing-masing bilangan bulat positif dan $1 = \max\{a, b\}$, kita mempunyai $a = b = 1$, adalah benar.

(ii) *Langkah induksi.* Andaikan pernyataan bahwa jika a dan b masing-masing bilangan bulat positif dan $n = \max\{a, b\}$ maka $a = b$, adalah benar (hipotesis induksi). Kita harus menunjukkan bahwa pernyataan tersebut juga benar untuk $n + 1$. Andaikan bahwa a dan b keduanya bilangan bulat positif dan $n + 1 = \max\{a, b\}$ sehingga $n = \max\{a - 1, b - 1\}$. Menurut hipotesis induksi, $a - 1 = b - 1$. Jadi $a = b$.

Karena kita sudah membuktikan basis induksi dan langkah induksi benar, maka menurut prinsip induksi matematik, dua bilangan bulat sembarang adalah sama!

18. Temukan kesalahan dalam pembuktian berikut. Kita ingin membuktikan bahwa $a^n = 1$ untuk semua bilangan bulat tak-negatif n bilamana a adalah bilangan riil tidak-nol. Kita akan membuktikan ini dengan prinsip induksi kuat.

(i) *Basis induksi.* Untuk $n = 0$, jelas $a^0 = 1$ adalah benar sesuai definisi a^0 .

(ii) *Langkah induksi.* Misalkan pernyataan tersebut benar untuk $0, 1, 2, \dots, n$, yaitu $a^0 = 1, a^1 = 1, a^2 = 1, \dots, a^n = 1$. Kita ingin memperlihatkan bahwa $a^{(n+1)} = 1$. Untuk menunjukkan hal ini, maka

$$\begin{aligned} a^{n+1} &= \frac{a^n \cdot a^n}{a^{n-1}} \\ &= \frac{1 \times 1}{1} \quad (\text{dari hipotesis induksi}) \\ &= 1 \end{aligned}$$

19. Tinjau runtunan nilai yang didefinisikan sebagai berikut:

$$S_{1,1} = 5$$

Untuk semua pasang bilangan bulat positif (m, n) , kecuali, $(1,1)$ didefinisikan

$$S_{m,n} = \begin{cases} S_{m-1,n} + 2 & \text{jika } n = 1 \\ S_{m,n-1} + 2 & \text{jika } n \neq 1 \end{cases}$$

Buktikan dengan induksi matematika bahwa untuk semua pasangan bilangan bulat positif (m, n) ,

$$S_{m,n} = 2(m+n) + 1$$

Esensi dari matematika adalah kebebasannya.
(George Cantor)

BAB 5

Algoritma dan Bilangan Bulat

Bulat air karena pembuluh, bulat kata karena mufakat.
(Pepatah adat Minangkabau)

Bab 5 ini akan membahas dua pokok bahasan: **algoritma** dan **bilangan bulat** (*integer*). Kedua pokok bahasan ini disatukan di dalam satu bab karena dalam pokok bahasan bilangan bulat kita akan mulai menggunakan terminologi algoritma untuk mendeskripsikan langkah-langkah penyelesaian masalah dengan suatu metode.

Algoritma menjadi penting karena ia merupakan jantung ilmu komputer. Banyak bahasan di dalam cabang-cabang ilmu komputer yang diacu dengan terminologi algoritma. Di dalam bab ini akan dijelaskan definisi algoritma, notasi yang digunakan, dan beberapa contoh algoritma dasar.

Pokok bahasan selanjutnya adalah bilangan bulat. Bilangan bulat atau *integer* memainkan peranan yang penting di dalam matematika diskrit. Sebagian besar (kalau tidak dapat dikatakan seluruhnya) pokok bahasan dalam matematika diskrit melibatkan bilangan bulat. Bahkan, cabang matematika yang bernama **teori bilangan** (*number theory*) mengkaji secara khusus bilangan bulat dan sifat-

sifatnya. Bilangan bulat sendiri merupakan objek diskrit, ini berlawanan dengan bilangan riil yang merupakan objek yang bersifat malar (*continue*).

5.1 Algoritma

Sebuah masalah dipecahkan dengan mendeskripsikan langkah-langkah penyelesaiannya. Misalnya masalah pengurutan (*sorting*) berikut: diberikan sejumlah bilangan bulat, tuliskan semua bilangan bulat tersebut dalam urutan yang menaik. Metode untuk pengurutan data diskrit sudah banyak ditemukan orang. Metode pengurutan sering digambarkan dalam sejumlah langkah terbatas yang mengarah pada solusi permasalahan. Urutan langkah-langkah penyelesaian masalah ini dinamakan algoritma.

DEFINISI 5.1 Algoritma adalah urutan logis langkah-langkah penyelesaian masalah yang disusun secara sistematis.

Ditinjau dari asal usul kata, kata algoritma sendiri mempunyai sejarah yang aneh. Kata ini tidak muncul di dalam kamus Webster sampai akhir tahun 1957. Orang hanya menemukan kata *algorism* yang berarti proses menghitung dengan angka Arab [KNU73]. Anda dikatakan *algorist* jika anda menggunakan angka Arab. Para ahli bahasa berusaha menemukan asal kata *algorism* ini namun hasilnya kurang memuaskan. Akhirnya para ahli sejarah matematika menemukan asal mula kata tersebut. Kata *algorism* berasal dari nama penulis buku Arab yang terkenal, yaitu Abu Ja'far Muhammad ibnu Musa al-Khuwarizmi (al-Khuwarizmi dibaca orang Barat menjadi *algorism*). Al-Khuwarizmi menulis buku yang berjudul *Kitab al jabar wal-muqabala*, yang artinya "Buku pemugaran dan pengurangan" (*The book of restoration and reduction*). Dari judul buku itu kita juga memperoleh akar kata "aljabar" (*algebra*). Perubahan dari kata *algorism* menjadi *algorithm* muncul karena kata *algorism* sering dikelirukan dengan *arithmetric*, sehingga akhiran *-sm* berubah menjadi *-thm*. Karena perhitungan dengan angka Arab sudah menjadi hal yang sudah biasa/lumrah, maka lambat laun kata *algorithm* berangsurngansur dipakai sebagai metode perhitungan (komputasi) secara umum, sehingga kehilangan makna aslinya [PAR95]. Dalam bahasa Indonesia, kata *algorithm* diserap menjadi algoritma.

Meskipun algoritma sering dikaitkan dengan ilmu komputer, namun sesungguhnya dalam kehidupan sehari-haripun banyak terdapat proses yang digambarkan dalam suatu algoritma. Cara-cara membuat kue atau masakan, misalnya dinyatakan dalam suatu resep. Misalnya resep membuat masakan Rendang Padang adalah contoh sebuah algoritma:

1. Potong daging sapi menjadi potongan-potongan dadu atau sesuai selera.
2. Haluskan bumbu berupa bawang merah, bawang putih, cabe merah, kunyit, laos, dan jahe.

- Masukkan seluruh bumbu tadi ke dalam santan. Tambahkan dua buah daun jeruk, satu lembar daun kunyit, dan sebatang serai.
- Masak santan di atas api sedang. Aduk terus hingga santan mendidih.
- Masukkan daging sapi, dan kecilkan api. Sekali-sekali santan diaduk agar tidak pecah.
- Jika sudah timbul minyak dan santan sudah kering, matikan api. Rendang siap dihidangkan.

Contoh-contoh algoritma yang lain dalam kehidupan sehari-hari misalnya pola pakaian, panduan praktikum, papan not balok, dan pengisian voucher ditunjukkan pada Tabel 5.1.

Tabel 5.1 Contoh-contoh Algoritma dalam Kehidupan Sehari-hari.

Proses	Algoritma	Contoh Langkah dalam Algoritma
1. Membuat kue	resep kue	Masukkan telur ke dalam wajan, kocok sampai mengembang
2. Membuat pakaian	pola pakaian	gunting kain dari pinggir kiri bawah ke arah kanan sejauh 5 cm
3. Praktikum reaksi kimia	panduan praktikum	campurkan 10 ml H ₂ SO ₄ dengan 15 ml NaOH
4. Merakit mobil	panduan merakit	sambungkan komponen A dengan komponen B
5. Kegiatan sehari-hari	jadwal harian	pukul 15.00 : tidur siang, pukul 16.00 : membuat PR
6. Memainkan musik	papan not balok	not balok
7. Mengisi voucher kartu prabayar telepon genggam (HP)	panduan pengisian	tekan nomor 888 masukkan nomor voucher 14 digit

5.2 Notasi Untuk Algoritma

Algoritma dapat dituliskan dalam berbagai notasi, misalnya dalam notasi kalimat-kalimat deksriptif seperti contoh resep masakan Rendang Padang di atas. Dengan notasi bergaya kalimat ini, deskripsi setiap langkah dijelaskan dengan bahasa sehari-hari secara gamblang. Setiap langkah biasanya diawali dengan kata kerja seperti ‘baca’, ‘hitung’, ‘masukkan’, ‘bagi’, ‘ganti’, dan sebagainya, sedangkan pernyataan bersyarat dinyatakan dengan ‘jika ... maka ...’.

Sebagai contoh pertama, kita akan menuliskan algoritma untuk mencari elemen terbesar (*maximum*) dari sebuah himpunan yang beranggotakan n buah bilangan bulat. Bilangan-bilangan bulat tersebut dinyatakan sebagai a_1, a_2, \dots, a_n . Elemen terbesar akan disimpan di dalam peubah (*variable*) yang bernama *maks*.

Algoritma Cari Elemen Terbesar

1. Asumsikan a_1 sebagai elemen terbesar sementara. Simpan a_1 ke dalam *maks*.
2. Bandingkan *maks* dengan elemen a_2 . Jika a_2 lebih besar dari *maks*, maka nilai *maks* diganti dengan a_2 .
3. Ulangi langkah 2 untuk elemen-elemen berikutnya (a_3, a_4, \dots, a_n).
4. Berhenti jika tidak ada lagi elemen yang dibandingkan. Dalam hal ini, *maks* berisi nilai dari elemen terbesar.

Notasi algoritma dengan kalimat deskriptif bagus untuk algoritma yang pendek, namun untuk masalah yang algoritmanya besar, notasi ini tidak mangkus. Selain itu, notasi kalimat deskriptif kadang-kadang dianggap kurang bisa menjelaskan sebuah algoritma.

Selain dengan notasi dekriptif, algoritma juga dapat digambarkan dalam notasi bahasa komputer (lebih tepatnya bahasa pemrograman), misalnya dalam bahasa Pascal atau Bahasa C. Dalam Bahasa Pascal, algoritma mencari elemen terbesar ditulis seperti pada Algoritma 5.1.

```
procedure CariElemenTerbesar(a : array_integer; n : integer;
                             var maks : integer);
{ Mencari elemen terbesar di dalam array a[1..n]. Elemen terbesar akan
  disimpan di dalam maks. array_integer adalah tipe array yang sudah
  didefinisikan di dalam program utama dengan pendeklarasian berikut:
    const Nmaks = 1000; { ukuran maksimum array }
    type array_integer = array[1..Nmaks] of integer ;
}
var
  i : integer;
begin
  maks := a[1];
  for i := 2 to n do
    if a[i] > maks then
      maks := a[i];
end;
```

Algoritma 5.1 Program Pascal untuk mencari elemen terbesar.

Sayangnya, setiap bahasa komputer memiliki aturan sintaks yang rumit yang membuat algoritma tersebut menjadi lebih sulit dipahami. Padahal, sebuah algoritma pada hakikatnya berisi abstraksi dari model penyelesaian masalah, sehingga algoritma seharusnya dibebaskan dari hal-hal teknis yang tidak perlu (misalnya tanda titik koma pada akhir setiap pernyataan, format masukan dan keluaran, dan lain-lain). Hal ini diperumit oleh kenyataan bahwa saat ini terdapat puluhan bahasa komputer, setiap bahasa tentu mempunyai aturan sintaks yang berbeda-beda.

Para ilmuwan komputer lebih menyukai menuliskan algoritma dalam notasi yang lebih praktis, yaitu notasi *pseudo-code*. *Pseudo-code* (*pseudo* artinya semu atau tidak sebenarnya) adalah notasi yang menyerupai notasi bahasa pemrograman tingkat tinggi, khususnya Bahasa *Pascal* dan *C*. Hasil pengamatan memperlihatkan bahwa bahasa pemrograman umumnya mempunyai notasi yang hampir mirip untuk beberapa instruksi, seperti notasi *if-then-else*, *while-do*, *repeat-until*, *read*, *write*, dan sebagainya. Berdasarkan pengamatan tersebut, maka beberapa penulis buku algoritma, termasuk penulis buku ini, mendefinisikan notasi algoritma yang disebut *pseudo-code* itu. Tidak seperti bahasa pemrograman yang direpotkan dengan tanda titik koma (*semicolon*), indeks, format keluaran, kata-kata khusus, dan sebagainya, sembarang versi *pseudo-code* dapat diterima asalkan perintahnya tidak membingungkan pembaca. Keuntungan menggunakan notasi *pseudo-code* adalah kemudahan mengkonversinya—lebih tepat disebut mentranslasi—ke notasi bahasa pemrograman, karena terdapat korespondensi antara setiap *pseudocode* dengan notasi bahasa pemrograman. Korespondensi ini dapat diwujudkan dengan tabel translasi dari notasi algoritmik ke notasi bahasa pemrograman apa pun.

Pseudo-code yang digunakan di dalam buku ini diadopsi dari Bahasa Pascal, namun tidak benar-benar mematuhi semua sintaks Pascal. Dengan menggunakan notasi *pseudo-code*, algoritma mencari elemen terbesar dituliskan dengan notasi *pseudo-code* seperti ditunjukkan pada Algoritma 5.2.

```
procedure CariElemenTerbesar(input a1, a2, ..., an : integer,
                               output maks : integer)
{ Mencari elemen terbesar di antara elemen a1, a2, ..., an. Elemen
  terbesar akan disimpan di dalam maks.
  Masukan: a1, a2, ..., an
  Keluaran: maks
}
Deklarasi
  i : integer

Algoritma:
  maks ← a1
  for i ← 2 to n do
    if ai > maks then
      maks ← ai
    endif
  endfor
```

Algoritma 5.2 Algoritma mencari elemen terbesar dalam notasi *pseudo-code*.

Kata-kata yang digarisbawahi menyatakan kata kunci (*keywords*) yang nantinya berpadanan dengan kata kunci pada bahasa komputer yang dipilih untuk mentranslasikan algoritma tersebut. Kalimat yang diapit dengan pasangan kurung kurawal ({ dan }) menyatakan komentar. Komentar berguna untuk lebih memperjelas instruksi yang dituliskan. Peubah lokal yang digunakan di dalam algoritma dituliskan

pada bagian **Deklarasi**, sedangkan langkah-langkah penyelesaian masalah dinyatakan di bagian **Algoritma**. Data masukan untuk algoritma dituliskan setelah kata input, sedangkan keluaran algoritma dituliskan sesudah kata output. Jika suatu peubah berfungsi sebagai masukan dan keluaran, maka peubah tersebut dituliskan seudah kata input/output. Karakter “ \leftarrow ” menyatakan bahwa nilai di sebelah kanannya diisikan ke dalam peubah di sebelah kirinya.

5.3 Beberapa Contoh Algoritma

Di bawah ini disajikan beberapa contoh algoritma. Algoritma pertama mempertukarkan nilai dari dua buah peubah. Algoritma kedua mencari elemen tertentu di antara sekumpulan nilai bilangan bulat. Sedangkan algoritma ketiga adalah algoritma untuk mengurutkan sekumpulan bilangan bulat.

Algoritma mempertukarkan nilai dari dua buah peubah.

Diberikan dua buah peubah, x dan y . Nilai x dan y dipertukarkan sehingga x akan berisi nilai y , sedangkan y akan berisi nilai x yang lama. Misalnya,

sebelum pertukaran, $x = 10$, $y = 8$,
setelah pertukaran, $x = 8$ dan $y = 10$.

Dalam operasi pertukaran ini, kita membutuhkan sebuah peubah bantu, $temp$, sebagai tempat menampung sementara nilai dari salah satu peubah (x atau y). Algoritma pertukaran nilai tersebut ditunjukkan oleh Algoritma 5.3.

```
procedure Tukar(input/output x, y : integer)
{ Mempertukarkan nilai x dan y
  Masukan: x dan y
  Keluaran: x dan y
}
Deklarasi
  temp : integer

Algoritma:
  temp  $\leftarrow$  x
  x  $\leftarrow$  y
  y  $\leftarrow$  temp
```

Algoritma 5.3 Algoritma mempertukarkan nilai dari dua buah peubah.

Algoritma mencari nilai tertentu di dalam himpunan elemen.

Diberikan n buah bilangan bulat yang dinyatakan sebagai a_1, a_2, \dots, a_n . Carilah apakah x terdapat di dalam himpunan bilangan bulat tersebut. Jika x ditemukan,

maka lokasi (indeks) elemen yang bernilai x disimpan di dalam peubah idx . Jika x tidak terdapat di dalam himpunan tersebut, maka idx diisi dengan nilai 0.

Algoritma pencarian yang paling sederhana adalah algoritma **pencarian beruntun** (*sequential search* atau *linear search*). Setiap elemen di dalam himpunan dibandingkan dengan x , mulai dari elemen pertama, a_1 , sampai elemen yang bernilai sama dengan x ditemukan atau sampai semua elemen sudah habis diperiksa. Jika $a_k = x$, maka k adalah lokasi tempat x berada (idx diisi dengan k). Jika x tidak ditemukan (semua elemen sudah habis diperiksa), maka 0 diisikan ke dalam idx . Algoritma pencarian yang termaksud ditunjukkan oleh Algoritma 5.4.

```
procedure PencarianBeruntun(input a1, a2, ..., an : integer, x : integer,
output idx : integer)
{ Mencari x di dalam elemen a1, a2, ..., an. Lokasi (indeks elemen) tempat
x ditemukan diisi ke dalam idx. Jika x tidak ditemukan, maka idx diisi
dengan 0.
Masukan: a1, a2, ..., an
Keluaran: idx
}
Deklarasi
i : integer

Algoritma:
i ← 1
while (i < n) and (ai ≠ x) do
    i ← i + 1
endwhile
{ i = n or ai = x }

if ai = x then { x ditemukan }
    idx ← i
else
    idx ← 0 { x tidak ditemukan }
endif
```

Algoritma 5.4 Algoritma pencarian beruntun.

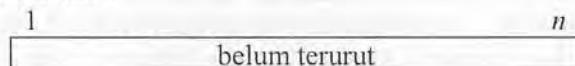
Algoritma pengurutan

Diberikan n buah bilangan bulat yang dinyatakan sebagai a_1, a_2, \dots, a_n . Urutkan semua bilangan bulat tersebut sedemikian sehingga $a_1 \leq a_2 \leq \dots \leq a_n$.

Pengurutan adalah persoalan yang kaya dengan solusi algoritma, karena saat ini terdapat puluhan algoritma pengurutan. Salah satu algoritma pengurutan adalah algoritma **pengurutan seleksi** (*selection sort*). Gagasan dasarnya adalah mencari elemen terbesar (atau terkecil), lalu menempatkan elemen terbesar (atau terkecil) itu pada awal atau akhir susunan (Gambar 5.1). Dua aktivitas ini disebut satu kali *pass*. Selanjutnya, elemen terujung tersebut “diisolasi” dan tidak disertakan pada proses selanjutnya. Untuk *pass* yang lainnya (seluruhnya ada $n - 1$ kali *pass*),

aktivitas yang sama diulang untuk elemen lain yang tersisa, yaitu memilih elemen terbesar (atau terkecil) berikutnya dan mempertukarkannya dengan elemen terujung dari bagian sisa.

Sebelum :



Sesudah:



Gambar 5.1 Bagian elemen yang terurut dan belum terurut pada metode Pengurutan Seleksi.

Algoritma pengurutan seleksi dituliskan di dalam Algoritma 5.5. Algoritma tersebut menghasilkan susunan yang menaik (*ascending order*). Perhatikan bahwa Algoritma 5.5 menggunakan Algoritma 5.2 (mencari elemen terbesar, tetapi yang dicatat hanya indeks elemen terbesar) dan Algoritma 5.3 (pertukaran dua buah nilai).

```
procedure PengurutanPilih(input n : integer,
                        input/output a1, a2, ..., an : integer,
                        )
{ Mengurutkan a1, a2, ..., an sehingga tersusun menaik dengan metode
  pengurutan maksimum.
  Masukan: a1, a2, ..., an
  Keluaran: a1, a2, ..., an (terurut menaik)
}
Deklarasi
  i      : integer      { pencacah jumlah pengulangan/pass }
  j      : integer      { pencacah untuk mencari nilai maksimum }
  imaks : integer      { indeks yang berisi nilai maksimum sementara }
  temp  : integer      { peubah bantu untuk pertukaran }

Algoritma:
  for i ← n downto 2 do    { jumlah pass sebanyak n - 1 }
    { cari elemen terbesar di antara a1, a2, ..., ai }
    imaks ← 1          { elemen pertama diasumsikan sebagai elemen
                         terbesar sementara }
    for j ← 2 to i do
      if aj > aimaks then
        imaks ← j
      endif
    endfor
```

```

(pertukarkan  $a_{\text{imaks}}$  dengan  $a_i$ )
temp  $\leftarrow a_i$ 
 $a_i \leftarrow a_{\text{imaks}}$ 
 $a_{\text{imaks}} \leftarrow \text{temp}$ 

endfor

```

Algoritma 5.5 Algoritma pengurutan pilih

5.4 Bilangan Bulat

Bilangan bulat adalah bilangan yang tidak mempunyai pecahan desimal, misalnya 8, 21, 8765, -34, 0, dan sebagainya (berlawanan dengan bilangan bulat adalah bilangan riil yang mempunyai titik desimal, seperti 8.0, 34.25, 0.02, dan sebagainya).

Pada upabab-upabab berikut, kita akan mempelajari aritmetika bilangan bulat yang berhubungan dengan sifat pembagian. Jika diberikan dua buah bilangan bulat a dan b , apakah “ a habis membagi b ”? (kebalikan dari pertanyaan tersebut adalah: apakah “ b kelipatan a ”?

Sifat pembagian pada bilangan bulat melahirkan konsep-konsep seperti bilangan prima dan aritmetika modulo. Satu algoritma penting yang berhubungan dengan sifat pembagian ini adalah algoritma Euclidean. Baik bilangan prima, aritmetika modulo, dan algoritma Euclidean memainkan peranan yang penting dalam bidang kriptografi, yaitu ilmu yang mempelajari kerahasiaan pesan.

5.5 Sifat Pembagian pada Bilangan Bulat

Pembahasan bilangan bulat kita mulai dari sifat pembagian (*division*). Mengapa pembagian? Karena salah satu konsep bilangan bulat yang berguna dalam aritmetika komputer adalah bilangan prima. Bilangan prima adalah bilangan yang hanya habis dibagi oleh 1 dan dirinya sendiri. Bahkan, sembarang bilangan bulat positif dapat dinyatakan sebagai hasil perkalian satu atau lebih bilangan prima.

DEFINISI 5.2. Misalkan a dan b adalah dua buah bilangan bulat dengan syarat $a \neq 0$. Kita menyatakan bahwa a **habis membagi** b (a divides b) jika terdapat bilangan bulat c sedemikian sehingga $b = ac$.

Notasi: $a | b$ jika $b = ac$, $c \in \mathbb{Z}$ dan $a \neq 0$.

(ingatlah bahwa $\mathbb{Z} =$ himpunan bilangan bulat)

Dengan kata lain, jika b dibagi dengan a , maka hasil pembagiannya berupa bilangan bulat. Kadang-kadang pernyataan “ a habis membagi b “ ditulis juga “ b **kelipatan** a ”.

Sebagai contoh, $4 \mid 12$ karena $12 \div 4 = 3$ (bilangan bulat) atau $12 = 4 \times 3$. Tetapi 4 tidak habis membagi 13 karena $13 \div 4 = 3.25$ (bukan bilangan bulat).

Secara umum, jika hasil pembagian bilangan bulat dinyatakan sebagai bilangan bulat juga, maka sembarang bilangan bulat bila dibagi dengan suatu bilangan bulat positif, maka selalu terdapat (1) hasil bagi dan (2) sisa pembagian. Misalnya, $13 \div 4$ memberikan hasil bagi 3 dan sisa 1 . Kasus khusus, jika a habis membagi b , maka sisa pembagian adalah 0 , misalnya $12 \div 4$ memberikan hasil bagi 3 dan sisa 0 . Perhatikan juga bahwa sisa hasil pembagian selalu lebih besar atau sama dengan nol tetapi lebih kecil dari pembagi. Sifat ini kita tuangkan dalam Teorema 5.1 berikut.

TEOREMA 5.1. Misalkan m dan n adalah dua buah bilangan bulat dengan syarat $n > 0$. Jika m dibagi dengan n maka terdapat dua buah bilangan bulat unik q (*quotient*) dan r (*remainder*), sedemikian sehingga

$$m = nq + r \quad (5.1)$$

dengan $0 \leq r < n$.

Teorema 5.1 sering disebut juga **teorema Euclidean** (dari nama ilmuwan Yunani yang bernama Euclid, lahir pada tahun 350 sebelum Masehi). Bilangan n disebut **pembagi** (*divisor*), m disebut **yang dibagi** (*dividend*), q disebut **hasil bagi** (*quotient*), dan r disebut **sisa** (*remainder*). Notasi yang digunakan untuk mengekspresikan hasil bagi dan sisa adalah dengan menggunakan operator *mod* dan *div* seperti berikut:

$$q = m \text{ div } n, \quad r = m \text{ mod } r$$

Contoh 5.1

1987 jika dibagi dengan 97 memberikan hasil bagi 20 dan sisa 47. Jadi, kita dapat menuliskan bahwa

$$1987 = 97 \cdot 20 + 47$$

atau diekspresikan sebagai $1987 \text{ div } 97 = 20$ dan $1987 \text{ mod } 97 = 47$. Begitu juga, jika -22 dibagi dengan 3 dapat dituliskan sebagai

$$-22 = 3(-8) + 2$$

atau diekspresikan sebagai $-22 \text{ div } 3 = -8$ dan $-22 \text{ mod } 3 = 2$.

Ingatlah bahwa sisa pembagian tidak boleh negatif, jadi kita tidak dapat menuliskan

$$-22 = 3(-7) - 1$$

karena $r = -1$ tidak memenuhi syarat $0 \leq r < n$.

Sebaliknya, jika 24 dibagi dengan 3, maka kita dapat menuliskan

$$24 = 3 \cdot 8 + 0$$

karena $r = 0$ memenuhi syarat $0 \leq r < n$. ■

5.6 Pembagi Bersama Terbesar

Dua buah bilangan bulat dapat memiliki faktor pembagi yang sama. Faktor pembagi bersama yang terpenting adalah faktor **pembagi bersama terbesar** (*greatest common divisor – gcd*)¹ atau PBB. Misalnya 45 memiliki faktor pembagi 1, 3, 5, 9, 15, dan 45 sendiri; sedangkan 36 memiliki faktor pembagi 1, 2, 3, 4, 9, 12, 18, dan 36 sendiri. Faktor pembagi bersama dari 45 dan 36 adalah 1, 3, 9, yang terbesar adalah 9 sehingga disimpulkan $\text{PBB}(45, 36) = 9$. Definisi formal dari PBB dinyatakan pada Definisi 5.3 berikut ini.

DEFINISI 5.3. Misalkan a dan b adalah dua buah bilangan bulat tidak nol. Pembagi bersama terbesar (PBB) dari a dan b adalah bilangan bulat terbesar d sedemikian sehingga $d | a$ dan $d | b$. Dalam hal ini kita nyatakan bahwa $\text{PBB}(a, b) = d$.

Sifat-sifat dari pembagi bersama terbesar dinyatakan dengan Teorema 5.2 [JOH97]:

TEOREMA 5.2. Misalkan a , b , dan c adalah bilangan bulat.

- (a) Jika c adalah PBB dari a dan b , maka $c | (a + b)$
- (b) Jika c adalah PBB dari a dan b , maka $c | (a - b)$
- (c) Jika $c | a$, maka $c | ab$

Pembuktian untuk Teorema 5.2 hanya dikerjakan untuk yang (a) saja, selebihnya diserahkan kepada pembaca sebagai latihan.

Bukti:

(a) Karena c adalah PBB dari a dan b , maka $c | a$ dan $c | b$. Karena $c | a$, maka berarti

$$a = cd_1 \tag{5.2}$$

untuk suatu bilangan bulat d_1 . Begitu juga karena $c | b$, maka berarti

$$b = cd_2 \tag{5.3}$$

¹ Di sekolah dasar dan menengah istilah “pembagi bersama terbesar” sering dinamakan “faktor persekutuan terbesar” atau FPB

untuk suatu bilangan bulat d_2 . Jumlah dari (5.2) dan (5.3) adalah

$$a + b = cd_1 + cd_2 = c(d_1 + d_2) \quad (5.4)$$

Dari (5.4) terlihat bahwa c habis membagi $a + b$. ■

Contoh 5.2

PBB dari 45 dan 36 adalah 9. Menurut Teorema 5.2,

- (a) 9 habis membagi $45 + 36 = 81$, atau $9 | (45 + 36)$
 - (b) 9 habis membagi $45 - 36 = 9$, atau $9 | (45 - 36)$
 - (c) 9 habis membagi $45 \cdot 36 = 1620$, atau $9 | (45 \cdot 36)$
-

Dari Teorema 5.1 kita ingin menunjukkan bahwa PBB dari dua buah bilangan bulat sama dengan PBB dari salah satu bilangan bulat tersebut dengan sisa hasil pembagiannya. Hal ini dinyatakan pada Teorema 5.3 berikut ini.

TEOREMA 5.3. Misalkan m dan n adalah dua buah bilangan bulat dengan syarat $n > 0$ sedemikian sehingga

$$m = nq + r, \quad 0 \leq r < n$$

maka $\text{PBB}(m, n) = \text{PBB}(n, r)$

Bukti untuk Teorema 5.3 tidak diberikan di sini dan ditinggalkan sebagai latihan buat pembaca.

Contoh 5.3

Jika 80 dibagi dengan 12 memberikan hasil 6 dan sisa 8, atau $80 = 12 \cdot 6 + 8$. Menurut Teorema 5.3,

$$\text{PBB}(80, 12) = \text{PBB}(12, 8) = 4$$

Jika 12 dibagi dengan 8 memberikan hasil 1 dan sisa 4, atau $12 = 8 \cdot 1 + 4$. Menurut Teorema 5.3,

$$\text{PBB}(12, 8) = \text{PBB}(8, 4) = 4$$

Jika 8 dibagi dengan 4 memberikan hasil 2 dan sisa 0, atau $8 = 4 \cdot 2 + 0$. Menurut Teorema 5.3,

$$\text{PBB}(8, 4) = \text{PBB}(4, 0) = 4$$

Dari runtunan perhitungan di atas, kita memperoleh bahwa

$$\text{PBB}(80, 12) = \text{PBB}(12, 8) = \text{PBB}(8, 4) = \text{PBB}(4, 0) = 4 \quad ■$$

5.7 Algoritma Euclidean

Di dalam upabab 5.6 telah diperlihatkan bahwa untuk mencari PBB dari dua buah bilangan bulat m dan n , mula-mula kita mendaftarkan semua pembagi dari masing-masing m dan n , lalu memilih pembagi persekutuan yang bernilai terbesar. Di dalam upabab 5.7 ini diberikan metode yang mangkus untuk menemukan PBB, yang dikenal dengan nama **algoritma Euclidean**. Algoritma Euclidean sudah dikenal sejak berabad-abad yang lampau. Euclid, penemu algoritma Euclidean, adalah seorang matematikawan Yunani yang menuliskan algoritmanya tersebut dalam bukunya yang terkenal, *Element*.

Algoritma Euclidean didasarkan pada aplikasi Teorema 5.3 secara berturut-turut sampai kita menemukan sisa pembagian bernilai nol, dan contoh ilustrasi yang menarik untuk hal ini sudah diperlihatkan pada Contoh 5.3. Secara formal algoritma Euclidean kita dirumuskan sebagai berikut:

Misalkan m dan n adalah bilangan bulat tak negatif dengan $m \geq n$. Misalkan $r_0 = m$ dan $r_1 = n$. Lakukan secara berturut-turut pembagian (Teorema 5.1) untuk memperoleh

$$r_0 = r_1 q_1 + r_2 \quad 0 \leq r_2 \leq r_1,$$

$$r_1 = r_2 q_2 + r_3 \quad 0 \leq r_3 \leq r_2,$$

⋮

$$r_{n-2} = r_{n-1} q_{n-1} + r_n \quad 0 \leq r_n \leq r_{n-1},$$

$$r_{n-1} = r_n q_n + 0$$

Menurut Teorema 5.3,

$$\begin{aligned} \text{PBB}(m, n) &= \text{PBB}(r_0, r_1) = \text{PBB}(r_1, r_2) = \dots = \text{PBB}(r_{n-2}, r_{n-1}) \\ &= \text{PBB}(r_{n-1}, r_n) = \text{PBB}(r_n, 0) = r_n \end{aligned}$$

Jadi, PBB dari m dan n adalah sisa terakhir yang tidak nol dari runtunan pembagian tersebut.

Diberikan dua buah bilangan bulat tak-negatif m dan n ($m \geq n$). Algoritma Euclidean berikut mencari pembagi bersama terbesar, PBB, dari kedua bilangan tersebut, yaitu bilangan bulat positif terbesar yang habis membagi m dan n .

Algoritma Euclidean

1. Jika $n = 0$ maka
 m adalah PBB(m, n);
stop.
tetapi jika $n \neq 0$,
lanjutkan ke langkah 2.
2. Bagilah m dengan n dan misalkan r adalah sisanya.
3. Ganti nilai m dengan nilai n dan nilai n dengan nilai r , lalu ulang kembali ke langkah 1.

Catatan: jika $m \leq n$, maka pertukarkan terlebih dahulu nilai m dan n .

Contoh 5.4

Dengan mengambil contoh yang sama dengan Contoh 5.3, maka PBB dari 80 dan 12 dicari dengan algoritma Euclidean sebagai berikut:

$$m = 80, n = 12 \text{ dan dipenuhi syarat } m \geq n$$

Karena $m = 12 \neq 0$, maka langkah instruksi 2 dikerjakan: 80 dibagi 12 memberikan hasil 6 dan sisa $r = 8$,

$$80 = 6 \cdot 12 + 8$$

Kerjakan langkah instruksi 3:

$$m = 12, n = 8$$

Kembali ke langkah instruksi 1, karena $n = 8 \neq 0$, maka langkah instruksi 2 dikerjakan: 12 dibagi 8 memberikan hasil 1 dan sisa $r = 4$,

$$12 = 1 \cdot 8 + 4$$

Kerjakan langkah instruksi 3:

$$m = 8, n = 4$$

Kembali ke langkah instruksi 1, karena $b = 4 \neq 0$, maka langkah instruksi 2 dikerjakan: 8 dibagi 4 memberikan hasil 2 dan sisa $r = 0$,

$$8 = 2 \cdot 4 + 0$$

Kerjakan langkah instruksi 3:

$$m = 4, n = 0$$

Kembali ke langkah instruksi 1, karena $b = 0$, maka PBB dari 80 dan 12 adalah nilai m terakhir, yaitu 4. Jadi $\text{PBB}(80, 12) = 4$.

Secara ringkas proses perhitungan dengan algoritma Euclidean di atas dinyatakan dalam runtunan pembagian berikut ini:

$$\begin{array}{l} 80 = 6 \cdot 12 + 8 \\ \downarrow \quad \downarrow \\ 12 = 1 \cdot 8 + 4 \\ \downarrow \quad \downarrow \\ 8 = 2 \cdot 4 + 0 \end{array}$$

Sisa pembagian terakhir sebelum 0 adalah 4, maka $\text{PBB}(80, 12) = 4$. ■

Dalam notasi *pseudo-code*, algoritma Euclidean kita tulis seperti pada Algoritma 5.6. Algoritma menerima masukan m dan n , dan menghasilkan keluaran $\text{PBB}(m, n)$.

```
procedure Euclidean(input m, n : integer, output PBB : integer)
{ Mencari  $\text{PBB}(m, n)$  dengan syarat  $m$  dan  $n$  bilangan tak-negatif dan  $m \geq n$ 
Masukan:  $m$  dan  $n$  dengan syarat  $m \geq n$  dan  $m, n \geq 0$ 
Keluaran:  $\text{PBB}(m, n)$ 
}

Deklarasi
r : integer

Algoritma:
while n ≠ 0 do
    r ← m mod n
    m ← n
    n ← r
endwhile
(  $n = 0$ , maka  $\text{PBB}(m, n) = m$  )
PBB ← m
```

Algoritma 5.6 Algoritma Euclidean

TEOREMA 5.4. Misalkan a dan b adalah dua buah bilangan bulat positif, maka terdapat bilangan bulat m dan n sedemikian sehingga $\text{PBB}(a, b) = ma + nb$.

Teorema 5.4 menyatakan bahwa PBB dua buah bilangan bulat a dan b dapat dinyatakan sebagai **kombinasi lanjar** (*linear combination*) dengan m dan n sebagai

koefisien-koefisennya. Misalnya $\text{PBB}(80, 12) = 4$, dan $4 = (-1) \cdot 80 + 7 \cdot 12$. Di sini $m = -1$ dan $n = 7$.

Metode untuk menemukan kombinasi lanjar dari dua buah bilangan sama dengan PBB-nya adalah dengan melakukan pekerjaan pembagian secara mundur pada algoritma Euclidean. Perhatikan Contoh 5.5 berikut ini.

Contoh 5.5

Nyatakan $\text{PBB}(312, 70) = 2$ sebagai kombinasi lanjar dari 312 dan 70.

Penyelesaian:

Terapkan algoritma Euclidean untuk memperoleh $\text{PBB}(312, 70) = 2$:

$$\begin{array}{ll} 312 = 4 \cdot 70 + 32 & \text{(i)} \\ 70 = 2 \cdot 32 + 6 & \text{(ii)} \\ 32 = 5 \cdot 6 + 2 & \text{(iii)} \\ 6 = 3 \cdot 2 + 0 & \text{(iv)} \end{array}$$

Susun pembagian nomor (iii) menjadi

$$2 = 32 - 5 \cdot 6 \quad \text{(iv)}$$

Susun pembagian nomor (ii) menjadi

$$6 = 70 - 2 \cdot 32 \quad \text{(v)}$$

Sulihkan (v) ke dalam (iv) menjadi

$$2 = 32 - 5 \cdot (70 - 2 \cdot 32) = 1 \cdot 32 - 5 \cdot 70 + 10 \cdot 32 = 11 \cdot 32 - 5 \cdot 70 \quad \text{(vi)}$$

Susun pembagian nomor (i) menjadi

$$32 = 312 - 4 \cdot 70 \quad \text{(vii)}$$

Sulihkan (vii) ke dalam (vi) menjadi

$$2 = 11 \cdot 32 - 5 \cdot 70 = 11 \cdot (312 - 4 \cdot 70) - 5 \cdot 70 = 11 \cdot 312 - 49 \cdot 70$$

Jadi, $\text{PBB}(312, 70) = 2 = 11 \cdot 312 - 49 \cdot 70$



Relatif Prima

DEFINISI 5.4. Dua buah bilangan bulat a dan b dikatakan relatif prima (*relatively prime*) jika $\text{PBB}(a, b) = 1$.

Sebagai contoh, 20 dan 3 relatif prima sebab $\text{PBB}(20, 3) = 1$. Begitu juga 7 dan 11 relatif prima karena $\text{PBB}(7, 11) = 1$. Tetapi 20 dan 5 tidak relatif prima sebab $\text{PBB}(20, 5) = 5 \neq 1$.

Jika a dan b relatif prima, maka menurut Teorema 5.4 kita dapat menemukan bilangan bulat m dan n sedemikian sehingga

$$ma + nb = 1 \quad (5.5)$$

Contoh 5.6

Bilangan 20 dan 3 adalah relatif prima karena $\text{PBB}(20, 3) = 1$, atau dapat dituliskan

$$2 \cdot 20 + (-13) \cdot 3 = 1$$

dengan $m = 2$ dan $n = -13$. Tetapi 20 dan 5 tidak relatif prima karena $\text{PBB}(20, 5) = 5 \neq 1$ sehingga 20 dan 5 tidak dapat dinyatakan dalam $m \cdot 20 + n \cdot 5 = 1$. ■

5.8 Aritmetika Modulo

Aritmetika modulo (*modular arithmetic*) memainkan peranan yang penting dalam komputasi integer, khususnya pada aplikasi kriptografi. Operator yang digunakan pada aritmetika modulo adalah **mod**. Operator mod memberikan sisa pembagian. Misalnya 23 dibagi 5 memberikan hasil = 4 dan sisa = 3, sehingga kita tulis $23 \bmod 5 = 3$ (operator mod sudah pernah kita gunakan pada Contoh 5.3). Definisi dari operator mod dinyatakan sebagai berikut:

DEFINISI 5.5. Misalkan a adalah bilangan bulat dan m adalah bilangan bulat > 0 . Operasi $a \bmod m$ (dibaca “ a modulo m ”) memberikan sisa jika a dibagi dengan m . Dengan kata lain, $a \bmod m = r$ sedemikian sehingga $a = mq + r$, dengan $0 \leq r < m$.

Notasi: $a \bmod m = r$ sedemikian sehingga $a = mq + r$, dengan $0 \leq r < m$.

Bilangan m disebut **modulus** atau **modulo**, dan hasil aritmetika modulo m terletak di dalam himpunan $\{0, 1, 2, \dots, m - 1\}$ (mengapa?).

Contoh 5.7

Beberapa hasil operasi dengan operator modulo:

- (i) $23 \bmod 5 = 3$ (karena 23 dibagi 5 memberikan hasil (q) = 4 dan sisa (r) = 3, atau dituliskan sebagai $23 = 5 \cdot 4 + 3$)
- (ii) $27 \bmod 3 = 0 \quad (27 = 3 \cdot 9 + 0)$
- (iii) $6 \bmod 8 = 6 \quad (6 = 8 \cdot 0 + 6)$

- (iv) $0 \bmod 12 = 12$ ($0 = 12 \cdot 0 + 12$)
(v) $-41 \bmod 9 = 4$ ($-41 = 9(-5) + 4$)
(vi) $-39 \bmod 13 = 0$ ($-39 = 13(-3) + 0$)

Penjelasan untuk (v): Karena a negatif, bagi $|a|$ dengan m mendapatkan sisa r' . Maka $a \bmod m = m - r'$ bila $r' \neq 0$. Jadi $|-41| \bmod 9 = 5$, sehingga $-41 \bmod 9 = 9 - 5 = 4$. ■

Jika $a \bmod m = 0$, maka dikatakan bahwa a adalah kelipatan dari m , yaitu a habis dibagi dengan m . Misalnya pada Contoh 5.7 di atas $27 \bmod 3 = 0$, berarti 27 adalah kelipatan 3.

Kongruen

Kadang-kadang dua buah bilangan bulat, a dan b , mempunyai sisa yang sama jika dibagi dengan bilangan bulat positif m . Kita katakan bahwa a dan b **kongruen dalam modulo m** , dan dilambangkan sebagai

$$a \equiv b \pmod{m}$$

(notasi ‘≡’ dibaca ‘kongruen’)

Jika a tidak kongruen dengan b dalam modulus m , maka ditulis

$$a \not\equiv b \pmod{m}$$

Misalnya $38 \bmod 5 = 3$ dan $13 \bmod 5 = 3$, maka $38 \equiv 13 \pmod{5}$. Definisi formal dari kekongruenan dinyatakan sebagai berikut:

DEFINISI 5.6. Misalkan a dan b adalah bilangan bulat dan m adalah bilangan > 0 , maka $a \equiv b \pmod{m}$ jika m habis membagi $a - b$.

Contoh 5.8

Bilangan 38 kongruen dengan 13 modulo 5 karena 5 membagi $38 - 13 = 25$, sehingga dapat kita tulis bahwa $38 \equiv 13 \pmod{5}$. Tetapi, 41 tidak kongruen dengan 30 modulo 5 karena 5 tidak habis membagi $41 - 30 = 11$, sehingga dapat kita tulis $41 \not\equiv 30 \pmod{5}$. Dengan cara yang sama, kita dapat menunjukkan bahwa

- | | |
|--------------------------|---|
| $17 \equiv 2 \pmod{3}$ | (3 habis membagi $17 - 2 = 15 \rightarrow 15 \div 3 = 5$) |
| $-7 \equiv 15 \pmod{11}$ | (11 habis membagi $-7 - 15 = -22 \rightarrow -22 \div 11 = 2$) |
| $12 \equiv 2 \pmod{7}$ | (7 tidak habis membagi $12 - 2 = 10$) |
| $-7 \equiv 15 \pmod{3}$ | (3 tidak habis membagi $-7 - 15 = -22$) ■ |

Kekongruenan $a \equiv b \pmod{m}$ dapat pula dituliskan dalam hubungan

$$a = b + km \quad (5.6)$$

yang dalam hal ini sembarang k adalah bilangan bulat. Pembuktianya adalah sebagai berikut: menurut Definisi 5.6, $a \equiv b \pmod{m}$ jika $m | (a - b)$. Menurut Definisi 5.1, jika $m | (a - b)$, maka terdapat bilangan bulat k sedemikian sehingga $a - b = km$ atau $a = b + km$.

Contoh 5.9

Dari Contoh 5.6 di atas kita dapat menyatakan bahwa

$$38 \equiv 13 \pmod{5} \text{ dapat ditulis sebagai } 38 = 13 + 5 \cdot 5$$

$$17 \equiv 2 \pmod{3} \text{ dapat ditulis sebagai } 17 = 2 + 5 \cdot 3$$

$$-7 \equiv 15 \pmod{11} \text{ dapat ditulis sebagai } -7 = 15 + (-2)11 \quad \blacksquare$$

Berdasarkan definisi aritmetika modulo (lihat Definisi 5.5), kita dapat menuliskan $a \bmod m = r$ sebagai

$$a \equiv r \pmod{m}$$

Contoh 5.10

Beberapa hasil operasi dengan operator modulo berikut:

- (i) $23 \bmod 5 = 3$ dapat ditulis sebagai $23 \equiv 3 \pmod{5}$
 - (ii) $27 \bmod 3 = 0$ dapat ditulis sebagai $27 \equiv 0 \pmod{3}$
 - (iii) $6 \bmod 8 = 6$ dapat ditulis sebagai $6 \equiv 6 \pmod{8}$
 - (iv) $0 \bmod 12 = 0$ dapat ditulis sebagai $0 \equiv 0 \pmod{12}$
 - (v) $-41 \bmod 9 = 4$ dapat ditulis sebagai $-41 \equiv 4 \pmod{9}$
 - (vi) $-39 \bmod 13 = 0$ dapat ditulis sebagai $-39 \equiv 0 \pmod{13}$
-

Sifat-sifat pengeraian hitung pada aritmetika modulo, khususnya terhadap operasi perkalian dan penjumlahan, dinyatakan dalam Teorema 5.5 berikut:

TEOREMA 5.5. Misalkan m adalah bilangan bulat positif.

1. Jika $a \equiv b \pmod{m}$ dan c adalah sembarang bilangan bulat maka

- (i) $(a + c) \equiv (b + c) \pmod{m}$
- (ii) $ac \equiv bc \pmod{m}$
- (iii) $a^p \equiv b^p \pmod{m}$ untuk suatu bilangan bulat tak negatif p .

2. Jika $a \equiv b \pmod{m}$ dan $c \equiv d \pmod{m}$, maka

- (i) $(a + c) \equiv (b + d) \pmod{m}$
- (ii) $ac \equiv bd \pmod{m}$

Bukti Teorema 5.5 hanya diperlihatkan untuk 1(ii) dan 2(i) saja. Bukti untuk 1(i), 1(iii), dan 2(ii) diserahkan sebagai latihan bagi pembaca.

Bukti:

1(ii) $a \equiv b \pmod{m}$ berarti:

$$\begin{aligned} &\Leftrightarrow a = b + km && \text{(dari persamaan 5.6)} \\ &\Leftrightarrow a - b = km \\ &\Leftrightarrow (a - b)c = ckm && \text{(kedua ruas dikalikan dengan } c\text{)} \\ &\Leftrightarrow ac = bc + Km && \text{(dalam hal ini, } K = kc\text{)} \\ &\Leftrightarrow ac \equiv bc \pmod{m} \end{aligned} \quad \blacksquare$$

2(i) $a \equiv b \pmod{m} \Leftrightarrow a = b + k_1 m$

$$\begin{aligned} c \equiv d \pmod{m} &\Leftrightarrow c = d + k_2 m + \\ &\Leftrightarrow (a + c) = (b + d) + (k_1 + k_2)m \\ &\Leftrightarrow (a + c) = (b + d) + km \quad \text{(dalam hal ini, } k = k_1 + k_2\text{)} \\ &\Leftrightarrow (a + c) \equiv (b + d) \pmod{m} \end{aligned} \quad \blacksquare$$

Contoh 5.11

Misalkan $17 \equiv 2 \pmod{3}$ dan $10 \equiv 4 \pmod{3}$, maka menurut Teorema 5.5,

$$17 + 5 = 2 + 5 \pmod{3} \Leftrightarrow 22 = 7 \pmod{3} \quad (\text{Teorema 5.5.1(i)})$$

$$17 \cdot 5 = 5 \cdot 2 \pmod{3} \Leftrightarrow 85 = 10 \pmod{3} \quad (\text{Teorema 5.5.1(ii)})$$

$$17 + 10 = 2 + 4 \pmod{3} \Leftrightarrow 27 = 6 \pmod{3} \quad (\text{Teorema 5.5.2(i)})$$

$$17 \cdot 10 = 2 \cdot 4 \pmod{3} \Leftrightarrow 170 = 8 \pmod{3} \quad (\text{Teorema 5.5.2(ii)}) \quad \blacksquare$$

Perhatikanlah bahwa Teorema 5.5. tidak memasukkan operasi pembagian pada aritmetika modulo karena jika kedua ruas dibagi dengan bilangan bulat, maka kekongruenan tidak selalu dipenuhi. Misalnya:

- $10 \equiv 4 \pmod{3}$ dapat dibagi dengan 2 karena $10/2 = 5$ dan $4/2 = 2$, dan $5 \equiv 2 \pmod{3}$
- $14 \equiv 8 \pmod{6}$ tidak dapat dibagi dengan 2, karena $14/2 = 7$ dan $8/2 = 4$, tetapi $7 \not\equiv 4 \pmod{6}$.

Inversi Modulo (Modulo Invers)

Di dalam aritmetika bilangan riil, inversi (*inverse*) dari perkalian adalah pembagian. Misalnya inversi dari 4 adalah $1/4$, karena $4 \times 1/4 = 1$. Di dalam aritmetika modulo, masalah menghitung inversi modulo lebih rumit.

Jika a dan m relatif prima dan $m > 1$, maka kita dapat menemukan inversi dari a modulo m . Inversi dari a modulo m adalah bilangan bulat \bar{a} sedemikian sehingga

$$a\bar{a} \equiv 1 \pmod{m}$$

Pembuktian inversi modulo ini sangat mudah [ROS99]. Dari Definisi 5.4 tentang relatif prima diketahui bahwa $\text{PBB}(a, m) = 1$, dan menurut persamaan (5.5) terdapat bilangan bulat p dan q sedemikian sehingga

$$pa + qm = 1$$

yang mengimplikasikan bahwa

$$pa + qm \equiv 1 \pmod{m}$$

Karena $qm \equiv 0 \pmod{m}$, maka

$$pa \equiv 1 \pmod{m}$$

Kekongruenan yang terakhir ini berarti bahwa p adalah inversi dari a modulo m . ■

Pembuktian di atas juga menceritakan bahwa untuk mencari inversi dari a modulo m , kita harus membuat kombinasi lanjar dari a dan m sama dengan 1. Koefisien a dari kombinasi lanjar tersebut merupakan inversi dari a modulo m . Perhatikan Contoh 5.12 berikut.

Contoh 5.12

Tentukan inversi dari $4 \pmod{9}$, $17 \pmod{7}$, dan $18 \pmod{10}$.

Penyelesaian:

(a) Karena $\text{PBB}(4, 9) = 1$, maka inversi dari $4 \pmod{9}$ ada. Dari algoritma Euclidean diperoleh bahwa

$$9 = 2 \cdot 4 + 1$$

Susun persamaan di atas menjadi

$$-2 \cdot 4 + 1 \cdot 9 = 1$$

Dari persamaan terakhir ini kita peroleh -2 adalah inversi dari 4 modulo 9 . Periksalah bahwa

$$-2 \cdot 4 \equiv 1 \pmod{9} \quad (9 \text{ habis membagi } -2 \cdot 4 - 1 = -9)$$

Catatlah bahwa setiap bilangan yang kongruen dengan -2 modulo 9 juga adalah inversi dari 4 , misalnya $7, -11, 16$, dan seterusnya, karena

$$\begin{array}{ll} 7 \equiv -2 \pmod{9} & (9 \text{ habis membagi } 7 - (-2) = 9) \\ -11 \equiv -2 \pmod{9} & (9 \text{ habis membagi } -11 - (-2) = -9) \\ 16 \equiv -2 \pmod{9} & (9 \text{ habis membagi } 16 - (-2) = 18) \end{array}$$

- (b) Karena PBB($17, 7$) = 1 , maka inversi dari $17 \pmod{7}$ ada. Dari algoritma Euclidean diperoleh rangkaian pembagian berikut:

$$\begin{array}{ll} 17 = 2 \cdot 7 + 3 & (\text{i}) \\ 7 = 2 \cdot 3 + 1 & (\text{ii}) \\ 3 = 3 \cdot 1 + 0 & (\text{iii}) \quad (\text{yang mengimplikasikan bahwa PBB}(17, 7) = 1) \end{array}$$

Susun (ii) menjadi:

$$1 = 7 - 2 \cdot 3 \quad (\text{iv})$$

Susun (i) menjadi

$$3 = 17 - 2 \cdot 7 \quad (\text{v})$$

Sulihkan (v) ke dalam (iv):

$$1 = 7 - 2 \cdot (17 - 2 \cdot 7) = 1 \cdot 7 - 2 \cdot 17 + 4 \cdot 7 = 5 \cdot 7 - 2 \cdot 17$$

atau

$$-2 \cdot 17 + 5 \cdot 7 = 1$$

Dari persamaan terakhir ini kita peroleh -2 adalah inversi dari 17 modulo 7 .

$$-2 \cdot 17 \equiv 1 \pmod{7} \quad (7 \text{ habis membagi } -2 \cdot 17 - 1 = -35)$$

- (c) Karena PBB($18, 10$) = $2 \neq 1$, maka inversi dari $18 \pmod{10}$ tidak ada.

Metode lain yang cukup sederhana untuk menghitung inversi modulo adalah dengan melihat bahwa menurut persamaan (5.6) kekongruenan

$$a \bar{a} \equiv 1 \pmod{m}$$

dapat ditulis dalam hubungan

$$a \bar{a} = 1 + km$$

sehingga persoalan menemukan inversi modulo adalah ekivalen dengan menemukan \bar{a} dan k sedemikian sehingga

$$\bar{a} = \frac{1+km}{a}$$

Sebagai ilustrasi, tinjau kembali Contoh 5.12 di atas, bahwa untuk inversi dari $4 \pmod{9}$ adalah \bar{a} sedemikian sehingga

$$4\bar{a} \equiv 1 \pmod{9}$$

Inversi dari $4 \pmod{9}$ adalah

$$\bar{a} = \frac{1+9k}{4}$$

Cobakan $k = 0, 1, 2, \dots$ dan $k = -1, -2, \dots$ ke dalam persamaan yang terakhir yang menghasilkan \bar{a} sebagai bilangan bulat. Hasilnya adalah untuk $k = 3$ diperoleh $\bar{a} = 7$, untuk $k = -1$ diperoleh $\bar{a} = -2$, dan seterusnya.

Selain dengan kedua cara yang telah disebutkan di atas, ada metode lain yang banyak digunakan untuk mencari inversi modulo, yaitu **algoritma Euclidean yang diperluas** (*extended Euclidean algorithm*) [ROS03]

Kekongruenan Lanjar

Kekongruenan lanjar adalah kongruen yang berbentuk

$$ax \equiv b \pmod{m}$$

dengan m adalah bilangan bulat positif, a dan b sembarang bilangan bulat, dan x adalah peubah. Bentuk kongruen lanjar berarti menentukan nilai-nilai x yang memenuhi kekongruenan tersebut. Metode yang sederhana untuk mencari nilai-nilai x tersebut adalah dengan menggunakan persamaan (5.6). Menurut persamaan (5.6), $ax \equiv b \pmod{m}$ dapat ditulis dalam hubungan

$$ax = b + km$$

yang dapat disusun menjadi

$$x = \frac{b + km}{a}$$

dengan k adalah sembarang bilangan bulat. Cobakan nilai-nilai $k = 0, 1, 2, \dots$ dan $k = -1, -2, \dots$ ke dalam persamaan yang terakhir untuk menghasilkan x sebagai bilangan bulat.

Contoh 5.13

Tentukan solusi dari (i) $4x \equiv 3 \pmod{9}$ dan (ii) $2x \equiv 3 \pmod{4}$

Penyelesaian:

- (i) Kekongruenan $4x \equiv 3 \pmod{9}$ ekivalen dengan menemukan k dan x bilangan bulat sedemikian sehingga

$$x = \frac{3 + k \cdot 9}{4}$$

Nilai k bilangan bulat yang menghasilkan x bulat adalah untuk $k = 1$ diperoleh $x = 3$, untuk $k = 5$ diperoleh $x = 12$, untuk $k = -3$ diperoleh $x = -6$, untuk $k = -6$ diperoleh $x = -15$, dan seterusnya. Jadi, nilai-nilai x yang memenuhi $4x \equiv 3 \pmod{9}$ adalah 3, 12, ... dan -6, -15, ...

- (ii) Kekongruenan $2x \equiv 3 \pmod{4}$ ekivalen dengan menemukan k dan x bilangan bulat sedemikian sehingga

$$x = \frac{3 + k \cdot 4}{2}$$

Karena $4k$ genap dan 3 ganjil maka penjumlahannya menghasilkan ganjil, sehingga hasil penjumlahan tersebut jika dibagi dengan 2 tidak menghasilkan bilangan bulat. Dengan kata lain, tidak ada nilai-nilai x yang memenuhi $2x \equiv 3 \pmod{4}$. ■

Metode lain untuk mencari solusi kekongruenan lanjar adalah dengan menggunakan inversi modulo. Caranya serupa dengan pencarian solusi pada persamaan lanjar biasa, seperti pada

$$4x = 12$$

Untuk mencari solusi persamaan di atas, kalikan kedua ruas dengan inversi perkalian dari 4, yaitu $1/4$,

$$\begin{aligned} 1/4 \cdot 4x &= 1/4 \cdot 12 \\ 1 \cdot x &= 3 \\ x &= 3 \end{aligned}$$

Sekarang, terapkan metode seperti ini pada kekongruenan lanjar pada Contoh 5.13 di atas,

$$4x \equiv 3 \pmod{9}$$

Kalikan kedua ruas dengan inversi dari 4 ($\pmod{9}$), yang dalam hal ini sudah dihitung pada Contoh 5.12, yaitu -2 :

$$-2 \cdot 4x \equiv -2 \cdot 3 \pmod{9}$$

$$-8x \equiv -6 \pmod{9}$$

Karena $-8 \equiv 1 \pmod{9}$, maka $x \equiv -6 \pmod{9}$. Jadi, solusi dari $4x \equiv 3 \pmod{9}$ adalah bilangan bulat x sedemikian sehingga $x \equiv -6 \pmod{9}$, yaitu 3, 12, ... dan $-6, -15, \dots$.

(Perhatikan bahwa

$$3 \equiv -6 \pmod{9}, \text{ karena } 9 \text{ habis membagi } 3 - (-6) = 9$$

$$2 \equiv -6 \pmod{9}, \text{ karena } 9 \text{ habis membagi } 12 - (-6) = 18$$

$$-6 \equiv -6 \pmod{9}, \text{ karena } 9 \text{ habis membagi } -6 - (-6) = 0$$

$$-15 \equiv -6 \pmod{9}, \text{ karena } 9 \text{ habis membagi } -15 - (-6) = -9$$

)

Chinese Remainder Problem

Pada abad pertama, seorang matematikawan China yang bernama Sun Tse mengajukan pertanyaan sebagai berikut:

Tentukan sebuah bilangan bulat yang bila dibagi dengan 5 menyisakan 3, bila dibagi 7 menyisakan 5, dan bila dibagi 11 menyisakan 7.

Pertanyaan Sun Tse dapat dirumuskan kedalam sistem kongruen lanjar:

$$x \equiv 3 \pmod{5}$$

$$x \equiv 5 \pmod{7}$$

$$x \equiv 7 \pmod{11}$$

Teorema *Chinese Remainder* berikut akan digunakan untuk menemukan solusi sistem kongruen lanjar seperti di atas.

TEOREMA 5.6. (*Chinese Remainder Theorem*) Misalkan m_1, m_2, \dots, m_n adalah bilangan bulat positif sedemikian sehingga $\text{PBB}(m_i, m_j) = 1$ untuk $i \neq j$. Maka sistem kongruen lanjar

$$\begin{aligned} x &\equiv a_1 \pmod{m_1} \\ &\vdots \\ &\equiv a_k \pmod{m_k} \end{aligned}$$
 memiliki solusi unik dalam modulo $m = m_1 \cdot m_2 \cdots m_n$.

Contoh 5.14.

Tentukan solusi dari pertanyaan Sun Tse di atas.

Penyelesaian:

Menurut persamaan (5.6), kongruen pertama, $x \equiv 3 \pmod{5}$, memberikan $x = 3 + 5k_1$ untuk beberapa nilai k . Sulihkan ini ke dalam kongruen kedua menjadi $3 + 5k_1 \equiv 5 \pmod{7}$

7), dari sini kita peroleh $k_1 \equiv 6 \pmod{7}$, atau $k_1 = 6 + 7k_2$ untuk beberapa nilai k_2 . Jadi kita mendapatkan $x = 3 + 5k_1 = 3 + 5(6 + 7k_2) = 33 + 35k_2$ yang mana memenuhi dua kongruen pertama. Jika x memenuhi kongruen yang ketiga, kita mempunyai $33 + 35k_2 \equiv 7 \pmod{11}$, yang mengakibatkan $k_2 \equiv 9 \pmod{11}$ atau $k_2 = 9 + 11k_3$. Sulihkan k_2 ini ke dalam kongruen yang ketiga menghasilkan $x = 33 + 35(9 + 11k_3) \equiv 348 + 385k_3 \pmod{11}$. Dengan demikian, $x \equiv 348 \pmod{385}$ yang memenuhi ketiga kongruen tersebut. Dengan kata lain, 348 adalah solusi unik modulo 385. Catatlah bahwa $385 = 5 \cdot 7 \cdot 11$. Solusi unik ini mudah dibuktikan sebagai berikut. Solusi tersebut modulo $m = m_1 \cdot m_2 \cdot m_3 = 5 \cdot 7 \cdot 11 = 5 \cdot 77 = 11 \cdot 35$. Karena $77 \cdot 3 \equiv 1 \pmod{5}$, $55 \cdot 6 \equiv 1 \pmod{7}$, dan $35 \cdot 6 \equiv 1 \pmod{11}$, solusi unik dari sistem kongruen tersebut adalah

$$\begin{aligned}x &\equiv 3 \cdot 77 \cdot 3 + 5 \cdot 55 \cdot 6 + 7 \cdot 35 \cdot 6 \pmod{385} \\&\equiv 3813 \pmod{385} \equiv 348 \pmod{385}\end{aligned}$$

■

5.9 Bilangan Prima

Bilangan bulat positif yang mempunyai aplikasi penting dalam ilmu komputer dan matematika diskrit adalah bilangan prima. Bilangan prima adalah bilangan bulat positif yang lebih besar dari 1 yang hanya habis dibagi oleh 1 dan dirinya sendiri.

DEFINISI 5.7: Bilangan bulat positif p ($p > 1$) disebut bilangan prima jika pembaginya hanya 1 dan p .

Sebagai contoh, 23 adalah bilangan prima karena ia hanya habis dibagi oleh 1 dan 23. Karena bilangan prima harus lebih besar dari 1, maka barisan bilangan prima dimulai dari 2, yaitu 2, 3, 5, 7, 11, 13, Seluruh bilangan prima adalah bilangan ganjil, kecuali 2 yang merupakan bilangan genap.

Bilangan selain prima disebut bilangan **komposit** (*composite*). Misalnya 20 adalah bilangan komposit karena 20 dapat dibagi oleh 2, 4, 5, dan 10, selain 1 dan 20 sendiri.

Teorema penting yang menyangkut bilangan prima dinyatakan oleh teorema yang terkenal dalam teori bilangan, yaitu **teorema fundamental aritmetik**, yang bunyinya adalah seperti di bawah ini.

TEOREMA 5.7 (The Fundamental Theorem of Arithmetic). Setiap bilangan bulat positif yang lebih besar atau sama dengan 2 dapat dinyatakan sebagai perkalian satu atau lebih bilangan prima.

Teorema 5.7 menyatakan bahwa baik bilangan prima maupun bilangan komposit, keduanya dapat dinyatakan sebagai perkalian dari satu atau lebih faktor prima. Misalnya,

$9 = 3 \times 3$	(2 buah faktor prima)
$100 = 2 \times 2 \times 5 \times 5$	(4 buah faktor prima)
$13 = 13$ (atau 1×13)	(1 buah faktor prima)

Masalah lain yang juga penting adalah menguji apakah sebuah bilangan merupakan prima atau bukan. Teorema 5.6 menyatakan bahwa sembarang bilangan bulat positif habis dibagi oleh faktor-faktor primanya. Faktor prima dari sebuah bilangan selalu lebih kecil atau sama dengan akar kuadrat dari bilangan tersebut. Hal ini mudah ditunjukkan sebagai berikut: misalkan a adalah faktor prima dari n , dengan $1 < a < n$, maka a habis membagi n dengan hasil bagi b sedemikian sehingga $n = ab$. Nilai a dan b haruslah $\leq n$ agar

$$ab > \sqrt{n} . \sqrt{n} = n$$

Dengan kata lain, faktor prima dari n selalu lebih kecil atau sama dengan \sqrt{n} . Hal ini dinyatakan dengan teorema berikut:

TEOREMA 5.8. Jika n adalah bilangan komposit, maka n mempunyai faktor prima yang lebih kecil atau sama dengan \sqrt{n} .

Untuk menguji apakah n merupakan bilangan prima atau komposit, kita cukup membagi n dengan sejumlah bilangan prima, mulai dari 2, 3, ..., bilangan prima $\leq \sqrt{n}$. Jika n habis dibagi dengan salah satu dari bilangan prima tersebut, maka n adalah bilangan komposit, tetapi jika n tidak habis dibagi oleh semua bilangan prima tersebut, maka n adalah bilangan prima.

Contoh 5.15

Tunjukkan apakah (i) 171 dan (ii) 199 merupakan bilangan prima atau komposit.

Penyelesaian:

- (i) $\sqrt{171} = 13.077$. Bilangan prima yang $\leq \sqrt{171}$ adalah 2, 3, 5, 7, 11, 13. Karena 171 habis dibagi 3, maka 171 adalah bilangan komposit.
- (ii) $\sqrt{199} = 14.107$. Bilangan prima yang $\leq \sqrt{199}$ adalah 2, 3, 5, 7, 11, 13. Karena 199 tidak habis dibagi 2, 3, 5, 7, 11, dan 13, maka 199 adalah bilangan prima.

Contoh 5.16

Temukan semua faktor prima dari 1617.

Penyelesaian:

Bagilah 1617 berturut-turut dengan barisan bilangan prima, mulai dari 2, 3, 5, 7,

2 tidak habis membagi 1617

3 habis membagi 1617, yaitu $1617/3 = 539$

Selanjutnya, bagilah 539 dengan bilangan prima berturut-turut, dimulai dari 3, 5, ...

- 3 tidak habis membagi 539
- 5 tidak habis membagi 539
- 7 habis membagi 539, yaitu $539/7 = 77$

Selanjutnya, bagilah 77 dengan bilangan prima berturut-turut, dimulai dari 7, 11, ...

- 7 habis membagi 77, yaitu $77/7 = 11$

Karena 11 adalah bilangan prima, maka pencarian faktor prima dari 1617 dihentikan.
Jadi, faktor prima dari 1617 adalah 3, 7, 7, dan 11, yaitu $1617 = 3 \times 7 \times 7 \times 11$. ■

Menguji keprimaan suatu bilangan bulat n dengan cara membaginya dengan sejumlah bilangan prima yang $\leq \sqrt{n}$ tentu tidak mangkus untuk n yang besar, karena kita harus menentukan terlebih dahulu semua bilangan prima yang lebih kecil dari \sqrt{n} . Untunglah terdapat metode lain yang dapat digunakan untuk menguji keprimaan suatu bilangan bulat, yang terkenal dengan **Teorema Fermat** (kadang-kadang dinamakan juga *Fermat's Little Theorem*). Fermat adalah seorang matematikawan Perancis pada tahun 1640.

TEOREMA 5.9 (Teorema Fermat). Jika p adalah bilangan prima dan a adalah bilangan bulat yang tidak habis dibagi dengan p , yaitu $\text{PBB}(a, p) = 1$, maka

$$a^{p-1} \equiv 1 \pmod{p}$$

Contoh 5.17

Kita akan menguji apakah 17 dan 21 bilangan prima atau bukan. Di sini kita mengambil nilai $a = 2$ karena $\text{PBB}(17, 2) = 1$ dan $\text{PBB}(21, 2) = 1$. Untuk 17,

$$2^{17-1} = 65536 \equiv 1 \pmod{17}$$

karena 17 habis membagi $65536 - 1 = 65535$ (yaitu, $65535 \div 17 = 3855$). Karena 17 memenuhi teorema Fermat, maka 17 adalah bilangan prima.

Untuk 21,

$$2^{21-1} = 1048576 \not\equiv 1 \pmod{21}$$

karena 21 tidak habis membagi $1048576 - 1 = 1048575$. Karena 21 tidak memenuhi teorema Fermat, maka 21 bukan bilangan prima. ■

Sayangnya, teorema Fermat ini mengandung kekurangan sebab terdapat bilangan komposit n sedemikian sehingga $2^{n-1} \equiv 1 \pmod{n}$. Bilangan bulat seperti itu disebut bilangan **prima semu** (*pseudoprimes*) terhadap basis 2. Misalnya

komposit 341 (yaitu $341 = 11 \cdot 31$) adalah bilangan prima semu terhadap basis 2 karena menurut teorema Fermat,

$$2^{340} \equiv 1 \pmod{341}$$

Kita dapat menggunakan bilangan selain 2 sebagai basis menguji bilangan prima semu.

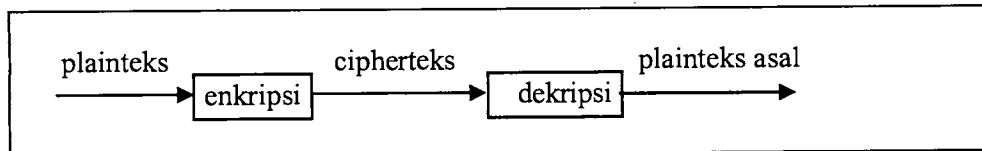
Untunglah bilangan prima semu relatif jarang terdapat [ROS99]. Untuk bilangan bulat yang lebih kecil dari 10^{10} terdapat 455.052.512 bilangan prima, tapi hanya 14.884 buah yang merupakan bilangan prima semu terhadap basis 2 [ROS03].

5.10 Kriptografi

Aritmetika modulo dan bilangan prima mempunyai banyak aplikasi dalam ilmu komputer, salah satu aplikasinya yang terpenting adalah **kriptografi**.

Kriptografi adalah ilmu sekaligus seni untuk menjaga keamanan pesan [SCH96]. Keamanan pesan diperoleh dengan menyandikannya menjadi pesan yang tidak mempunyai makna. Zaman sekarang ini kerahasiaan informasi menjadi sesuatu yang penting. Informasi yang rahasia perlu disembunyikan agar tidak diketahui oleh orang yang tidak berhak. Anda tentu tidak ingin nomor PIN kartu kredit atau kartu ATM anda diketahui orang, bukan? Atau, jika anda menulis sebuah pesan yang sifatnya rahasia, anda tidak ingin pesan tersebut dibaca oleh orang lain. Kriptografi dapat digunakan untuk menyamarkan informasi rahasia itu dari orang atau pihak yang tidak berhak membacanya.

Pesan yang dirahasiakan dinamakan **plainteks** (*plaintext*, artinya teks jelas yang dapat dimengerti), sedangkan pesan hasil penyandian disebut **cipherteks** (*ciphertext*, artinya teks tersandi). Pesan yang telah disandikan dapat dikembalikan lagi ke pesan aslinya hanya oleh orang yang berhak (orang yang berhak adalah orang yang mengetahui metode penyandian atau memiliki kunci penyandian). Proses menyandikan plainteks menjadi cipherteks disebut **enkripsi** (*encryption*) dan proses membalikkan cipherteks menjadi plainteksnya disebut **dekripsi** (*decryption*). Gambar 5.1 memperlihatkan diagram kedua proses yang dimaksud.



Gambar 5.1. Enkripsi dan Dekripsi

Sebagai contoh, sebuah pesan rahasia (plainteks) berikut:

uang disimpan di balik buku X

disandikan menjadi cipherteks dengan suatu teknik kriptografi tertentu menjadi:

j&kloP(d\$gkhtpuBn%6^k1p..t@

Cipherteks, meskipun tidak dirahasiakan, namun isinya sudah tidak jelas dan tidak dapat dimengerti maksudnya. Hanya orang yang berhak yang dapat mengeminkan pesan tersebut menjadi pesan semula.

Kriptografi digunakan untuk dua aplikasi, yaitu aplikasi pengiriman data melalui saluran komunikasi dan aplikasi penyimpanan data di dalam *disk storage*. Data ditransmisikan melalui saluran komunikasi dalam bentuk cipherteks. Di tempat penerima cipherteks dikembalikan lagi menjadi plainteks. Sedangkan untuk aplikasi penyimpanan, data di dalam media penyimpanan (seperti *hard disk*) disimpan dalam bentuk cipherteks. Untuk membacanya, hanya orang yang berhak yang dapat membalikkan cipherteks menjadi plainteks.

Contoh enkripsi dan dekripsi pada data tersimpan misalnya enkripsi pada dokumen plainteks yang bernama plain.txt yang isinya adalah:

Ketika saya berjalan-jalan di pantai,
saya menemukan banyak sekali kepiting
yang merangkak menuju laut. Mereka
adalah anak-anak kepiting yang baru
menetas dari dalam pasir. Naluri
mereka mengatakan bahwa laut adalah
tempat kehidupan mereka.

Misalkan hasil enkripsi plain.txt dengan metode kriptografi tertentu disimpan di dalam berkas cipher.txt yang isinya adalah sebagai berikut:

Ztâxzp/épêp/qtüyp{p}<yp{p}/sx/□p}âpx;
□épêp/ | t } t | äzp}/qp}êpz/étpz{x/zt□xâx
>v□ép}v/ | tüp}vzpz/ | t } äyä/ {pää= / \tütz
p□psp{pw/p}pz<p>pz/zt□xâx}v/ép}
v/qpüä□ | t } tâpé/spüx/sp{p| / □pexü= /]
p{äüx□ | ttüzp/ | t } vpâpzp}/qpwâp/ {pää
/ psp{pw□ât | □pâ/ztxwsä□p}/ | tützp=

Hasil dekripsi terhadap berkas cipher.txt menghasilkan berkas yang tepat sama dengan berkas plain.txt semula:

Ketika saya berjalan-jalan di pantai,
saya menemukan banyak sekali kepiting
yang merangkak menuju laut. Mereka
adalah anak-anak kepiting yang baru
menetas dari dalam pasir. Naluri
mereka mengatakan bahwa laut adalah
tempat kehidupan mereka.

Perdagangan elektronis (*e-commerce*) pun menggunakan kriptografi untuk menyandikan nomor PIN *customer*, karena pembayaran di dalam *e-commerce* umumnya menggunakan kartu kredit. Pengacakan (*scrambling*) siaran televisi melalui parabola, seperti pada siaran Piala Dunia 2002 di RCTI baru-baru ini, juga termasuk termasuk ke dalam aplikasi kriptografi. Di sini, penyandian dilakukan dengan mengacak susunan gambar.

Sejarah Kriptografi

Menurut sejarahnya, kriptografi sudah lama digunakan oleh tentara Sparta di Yunani pada permulaan tahun 400 SM. Mereka menggunakan alat yang disebut *scytale*. Alat ini terdiri dari sebuah pita panjang dari daun papyrus yang dililitkan pada sebatang silinder (Gambar 5.2). Pesan yang akan dikirim ditulis horizontal (baris per baris). Bila pita dilepaskan, maka huruf-huruf di dalamnya telah tersusun membentuk pesan rahasia. Untuk membaca pesan, penerima melilitkan kembali silinder yang diameternya sama dengan diameter silinder pengirim. Teknik kriptografi seperti ini dikenal dengan nama transposisi cipher, yang merupakan metode enkripsi tertua.



Gambar 5.2. *Scytale* yang digunakan oleh tentara Yunani untuk transposisi pesan

Kriptanalisis dan Kriptologi

Kriptanalisis (*cryptanalysis*) adalah ilmu dan seni untuk memecahkan cipherteks menjadi plainteks tanpa mengetahui *kunci* yang diberikan. Jadi, kriptanalisis adalah kebalikan dari kriptografi. Pelakunya disebut **kriptanalisis**. **Kriptologi** (*cryptology*) adalah studi mengenai kriptografi dan kriptanalisis (*cryptanalyst*).

Jika seorang kriptografer menggunakan enkripsi untuk merahasiakan pesan dan mendekripsikannya kembali, maka seorang kriptanalisis (*cryptanalyst*) mempelajari metode enkripsi dan cipherteks dengan tujuan menemukan plainteksnya. Baik kriptografer maupun kriptanalisis menerjemahkan cipherteks menjadi plainteks, namun kriptografer bekerja atas legitimasi pengirim atau penerima pesan, sedangkan kriptanalisis bekerja atas nama penyusup yang tidak berhak.

Notasi Matematis

Jika cipherteks dilambangkan dengan C dan plainteks dilambangkan dengan P , maka fungsi enkripsi E memetakan P ke C ,

$$E(P) = C \quad (5.7)$$

Pada proses kebalikannya, fungsi dekripsi D memetakan C ke P ,

$$D(C) = P \quad (5.8)$$

Dengan kata lain, D adalah fungsi inversi (*inverse*) dari E , atau $D = E^{-1}$. Seperti yang telah dijelaskan di dalam Bab 3, fungsi yang mempunyai inversi adalah fungsi yang berkoresponden satu-ke-satu. Sifat berkoresponden satu-ke-satu harus djamin pada fungsi enkripsi dan dekripsi, karena cipherteks harus dapat dikembalikan ke plainteks semula.

Karena proses enkripsi kemudian dekripsi mengejeminversi pesan ke pesan asal, maka kesamaan berikut harus benar,

$$D(E(P)) = P \quad (5.9)$$

Algoritma kriptografi –atau *cipher*– adalah fungsi matematika yang digunakan untuk enkripsi dan dekripsi. Kekuatan suatu algoritma kriptografi diukur dari banyaknya kerja yang dibutuhkan untuk memecahkan data cipherteks menjadi plainteksnya. Kerja ini dapat diequivalekan dengan waktu. Semakin banyak usaha yang diperlukan, yang berarti juga semakin lama waktu yang dibutuhkan, maka semakin kuat algoritma kriptografinya, yang berarti semakin aman digunakan untuk menyandikan pesan.

Jika kekuatan kriptografi ditentukan dengan menjaga kerahasiaan algoritmanya, maka algoritma kriptografinya dinamakan algoritma *restricted*. Misalkan di dalam sebuah kelompok orang mereka sepakat menyandikan setiap pesan pesan dengan algoritma yang sama. Algoritmanya adalah mempertukarkan pada setiap kata karakter pertama pdengan karakter kedua, karakter ketiga dengan karakter keempat dan seterusnya. Contohnya,

Plainteks: STRUKTUR DISKRIT

Cipherteks: TSURTKRU IDKSIRT

Untuk mendekripsikan pesan, algoritma yang sama digunakan kembali. Sayangnya, algoritma *restricted* tidak cocok lagi saat ini. Bila salah seorang dari anggota keluar dari kelompok, maka algoritma penyandian pesan harus diubah lagi karena kerahasiaannya tidak dapat lagi diandalkan.

Kriptografi modern tidak lagi mendasarkan kekuatan pada algoritmanya. Jadi algoritmanya tidak dirahasiakan dan boleh diketahui oleh umum. Kekuatan kriptografinya terletak pada kunci, yaitu berupa deretan karakter atau bilangan bulat. Kunci ini dijaga kerahasiaannya, dan hanya orang yang mengetahui kunci dapat melakukan enkripsi dan dekripsi. Kunci di sini sama fungsinya dengan sandi-lewat (*password*) pada sistem komputer, PIN pada kartu ATM atau kartu kredit. Bedanya, jika sandi-lewat atau PIN bertujuan untuk otorisasi akses, maka kunci pada kriptografi digunakan pada proses enkripsi maupun dekripsi.

Misalnya pada *Caesar cipher*, yaitu teknik kriptografi yang digunakan oleh Kaisar Romawi, Julius Caesar, untuk menyandikan pesan yang ia kirim kepada para gubernurnya. Pada *Caesar cipher*, tiap huruf disubstitusi dengan huruf ketiga berikutnya dari susunan alfabet. Dalam hal ini kuncinya adalah jumlah pergeseran huruf (yaitu 3). Susunan alfabet setelah digeser sejauh 3 huruf adalah:

Plainteks: A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
Cipherteks: D E F G H I J K L M N O P Q R S T U V W X Y Z A B C

Jadi, huruf A pada plainteks disubstitusi dengan D, huruf B disubstitusi dengan E, demikian seterusnya. Dengan mengkodekan setiap huruf alfabet dengan *integer*: ‘A’ = 0, ‘B’ = 1, ..., ‘Z’ = 25, maka secara matematis pergeseran 3 huruf alfabet ekivalen dengan melakukan operasi modulo terhadap plainteks p menjadi cipherteks c dengan persamaan

$$c = E(p) = (p + 3) \bmod 26 \quad (5.10)$$

Contoh 5.18

Pesan

AWASI ASTERIX DAN TEMANNYA OBELIX

disandikan dengan *Caesar Cipher*. Dengan mengkodekan ‘A’ = 0, ‘B’ = 1, ..., ‘Z’ = 25, maka cipherteks dihitung dengan persamaan 5.10 sebagai berikut:

$$\begin{aligned} p_1 = 'A' = 0 & \rightarrow c_1 = E(0) = (0 + 3) \bmod 26 = 3 = 'D' \\ p_2 = 'W' = 22 & \rightarrow c_2 = E(22) = (22 + 3) \bmod 26 = 25 = 'Z' \\ p_3 = 'A' = 0 & \rightarrow c_3 = E(0) = (0 + 3) \bmod 26 = 3 = 'D' \\ p_4 = 'S' = 18 & \rightarrow c_4 = E(18) = (18 + 3) \bmod 26 = 21 = 'V' \\ \text{dst...} & \end{aligned}$$

Bila keseluruhan perhitungan diselesaikan, maka diperoleh cipherteksnya adalah

DZDVL DVWHULA GDQ WHPDQQBA REHOLA

Alternatif lain, cipherteks juga dapat langsung diperoleh dengan menggunakan tabel pergeseran 3 huruf di atas, yaitu: A disubstitusidengan D, W disubstitusi dengan Z, A disubstitusi dengan D, W disubstitusi dengan V, dst. ■

Penerima pesan mengembalikan lagi cipherteks dengan operasi kebalikan yang secara matematis dapat dinyatakan dengan persamaan

$$p = D(c) = (c - 3) \bmod 26 \quad (5.11)$$

Perhatikan bahwa D adalah inversi (*inverse*) dari fungsi E , yaitu $D(c) = E^{-1}(p)$. Sehingga, cipherteks

DZDVL DVWHULA GDQ WHPDQQBA REHOLA

dikembalikan menjadi plainteks asal dengan persamaan 5.11 menjadi

AWASI ASTERIX DAN TEMANNYA OBELIX

Secara umum, fungsi enkripsi dan dekripsi pada *Caesar Cipher* dapat dibuat lebih umum dengan menggeser huruf alfabet sejauh k sehingga

$$c = E(p) = (p + K) \bmod 26 \quad (5.12)$$

untuk fungsi enkripsi dan

$$p = D(c) = (c - K) \bmod 26 \quad (5.13)$$

untuk fungsi dekripsi. Pada kedua persamaan terakhir ini, K berlaku sebagai kunci rahasia.

Secara matematis, pada sistem kriptografi yang menggunakan kunci K , maka fungsi enkripsi dan dekripsi menjadi

$$E_{K1}(P) = C \quad (5.14)$$

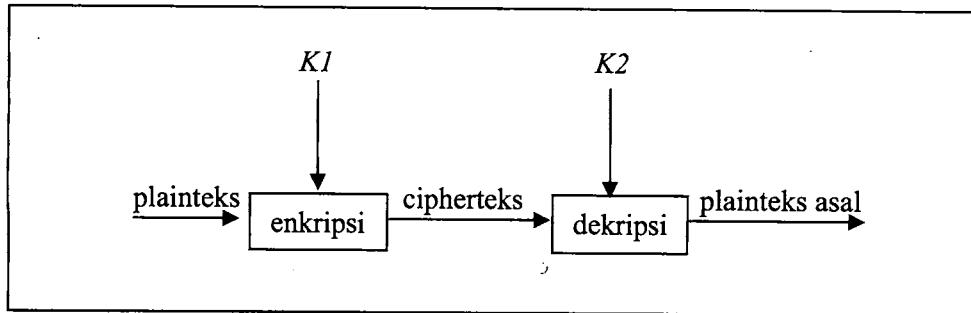
dan

$$D_{K2}(C) = P \quad (5.15)$$

Kedua fungsi ini memenuhi

$$D_{K2}(E_{K1}(P)) = P$$

Gambar 5.3 memperlihatkan diagram proses enkripsi dan dekripsi pada kriptografi yang menggunakan kunci.



Gambar 5.3. Enkripsi dan Dekripsi pada algoritma kriptografi modern.

Jika $K1 = K2$ (yaitu, kunci untuk proses enkripsi sama dengan kunci untuk dekripsi), maka sistem kriptografinya dinamakan **sistem kriptografi kunci-simetri** (*symmetric-key cryptosystem*), (yang dimaksud dengan sistem kriptografi adalah gabungan dari algoritma kriptografi, kunci, plainteks, dan cipherteks) dan algoritma kriptografinya disebut **algoritma simetri**. Contoh algoritma simetri adalah *DES* (*Data Encryption Standard*). Sebaliknya, jika $K1 \neq K2$ (yaitu, kunci untuk proses enkripsi berbeda dengan kunci untuk dekripsi), sistem kriptografinya dinamakan **sistem kriptografi nirsimetri** (*asymmetric cryptosystem*), dan algoritma kriptografinya disebut **algoritma nirsimetri**. Contoh algoritma nirsimetri adalah *RSA* (singkatan dari tiga nama penemu algoritmanya: Rivest-Shamir-Adleman).

Kriptografi simetri kadang-kadang disebut juga **kriptografi kunci-pribadi** (*private-key cryptography*) karena kunci enkripsi dan dekripsi sama dan harus dirahasiakan. Kelemahan dari sistem ini adalah baik pengirim maupun penerima pesan harus memiliki kunci yang sama, sehingga pengirim pesan harus mencari cara lain untuk memberitahukan kunci kepada penerima.

Kriptografi nirsimetri kadangkala disebut juga **kriptografi kunci-publik** (*public key cryptography*). Algoritma ini mempunyai dua buah kunci, yaitu kunci publik (*public key* – tidak rahasia) untuk enkripsi dan kunci pribadi (*secret key* – rahasia) untuk dekripsi. Pengirim pesan (*sender*) mengenkripsi pesan yang akan dikirim dengan menggunakan kunci publik si penerima pesan (*receiver*). Hanya penerima pesan yang dapat mendekripsi pesan karena hanya ia yang mengetahui kunci pribadi miliknya. Misalkan jaringan komputer menghubungkan komputer karyawan di kantor cabang dengan komputer menejer di kantor pusat. Seluruh karyawan diperintahkan bahwa kalau mereka mengirim laporan ke menejer di kantor pusat, mereka harus mengenkripsikan laporan tersebut dengan kunci publik milik menejer (jadi, kunci publik menejer diketahui oleh seluruh karyawan). Untuk mengeminkversi data tersandi ke data asal, hanya menejer yang dapat melakukannya, karena dia lah yang memegang kunci rahasia. Selama proses transmisi cipherteks dari kantor cabang ke kantor pusat melalui saluran

komunikasi mungkin saja data yang dikirim disadap oleh pihak ketiga, namun pihak ketiga ini tidak dapat mengeminkversi cipherteks ke palinteksnya karena ia tidak mengetahui kunci untuk dekripsi.

DES (*Data Encryption Standard*)

DES memadukan teknik permutasi, ekspansi, kompaksi, dan substitusi, semuanya dilakukan dalam 16 kali perulangan. Panjang kunci *DES* adalah 8 karakter atau 64 bit. Dari 64 bit tersebut, hanya 56 bit saja yang dipakai dalam proses enkripsi. Tetapi patut dicatat bahwa dengan 56 bit itu akan terdapat 2^{56} atau 72.057.594.037.927.936 kemungkinan kunci. Jika orang yang tidak berhak mencoba keseluruhan kemungkinan kunci tersebut dengan menggunakan satu juta prosesor komputer yang bekerja secara paralel, maka dengan asumsi bahwa selama 1 detik dapat dicoba satu juta kemungkinan kunci, maka seluruh kemungkinan kunci tersebut memerlukan waktu 2284 tahun untuk menemukan kunci yang benar. Ini sebuah waktu yang lama, bahkan kriptanalisis yang mencobanya pun sudah meninggal dunia sebelum waktu itu selesai. Algoritma *DES* tidak dibahas di dalam bab ini karena tidak relevan dengan pokok bahasan kita yang menyangkut penggunaan bilangan prima dan aritmetika modulo.

RSA (*Rivest-Shamir-Adleman*)

Algoritma *RSA* diperkenalkan oleh tiga peneliti dari *MIT (Massachusetts Institute of Technology)*, yaitu Ron Rivest, Adi Shamir, dan Len Adleman, pada tahun 1976. *RSA* mendasarkan proses enkripsi dan dekripsinya pada konsep bilangan prima dan aritmetika modulo. Baik kunci enkripsi maupun kunci dekripsi keduanya berupa bilangan bulat. Kunci enkripsi tidak dirahasiakan dan diketahui umum (sehingga dinamakan juga **kunci publik**), namun kunci untuk dekripsi bersifat rahasia. Kunci dekripsi dibangkitkan dari beberapa buah bilangan prima bersama-sama dengan kunci enkripsi. Untuk menemukan kunci dekripsi, orang harus memfaktorkan suatu bilangan non prima menjadi faktor primanya. Kenyataannya, memfaktorkan bilangan non prima menjadi faktor primannya bukanlah pekerjaan yang mudah. Belum ada algoritma yang mangkus (efisien) yang ditemukan untuk pemfaktoran itu. Semakin besar bilangan non primanya tentu semakin sulit pula pemfaktorannya. Semakin sulit pemfaktorannya, semakin kuat pula algoritma *RSA*. Algoritma *RSA* sebenarnya sederhana sekali. Secara ringkas, algoritma *RSA* terdiri dari tiga bagian, yaitu bagian untuk membangkitkan pasangan kunci, bagian untuk enkripsi, dan bagian untuk dekripsi:

ALGORITMA RSA

Pembakitan pasangan kunci

1. Pilih dua buah bilangan prima sembarang, sebut a dan b . Jaga kerahasiaan a dan b ini.
2. Hitung $n = a b$. Besaran n tidak perlu dirahasiakan.
3. Hitung $m = (a - 1)(b - 1)$. Sekali m telah dihitung, a dan b dapat dihapus untuk mencegah diketahuinya oleh pihak lain.
4. Pilih sebuah bilangan bulat untuk kunci publik, sebut namanya e , yang relatif prima terhadap m .
5. Hitung kunci dekripsi, d , dengan kekongruenan $ed \equiv 1 \pmod{m}$.

Enkripsi

1. Nyatakan pesan menjadi blok-blok plainteks: p_1, p_2, p_3, \dots (harus dipenuhi persyaratan bahwa nilai p_i harus terletak dalam himpunan nilai $0, 1, 2, \dots, n - 1$ untuk menjamin hasil perhitungan tidak berada di luar himpunan)
2. Hitung blok cipherteks c_i untuk blok plainteks p_i dengan persamaan

$$c_i = p_i^e \pmod{n}$$

yang dalam hal ini, e adalah kunci publik.

Dekripsi

1. Proses dekripsi dilakukan dengan menggunakan persamaan

$$p_i = c_i^d \pmod{n},$$

yang dalam hal ini, d adalah kunci pribadi.

Algoritma 5.7 Algoritma RSA

Perhatikan langkah 5 pada proses pembakitan pasangan kunci. Kekongruenan $ed \equiv 1 \pmod{m}$ sama dengan $ed \pmod{m} = 1$. Menurut persamaan (5.6) yang menyatakan bahwa $a \equiv b \pmod{m}$ ekivalen dengan $a = b + km$, maka $ed \equiv 1 \pmod{m}$ ekivalen dengan $ed = 1 + km$, sehingga d dapat dihitung dengan cara yang sederhana dengan persamaan

$$d = \frac{1 + km}{e} \tag{5.16}$$

Dalam implementasi yang sebenarnya, nilai a dan b disarankan nilai yang sangat besar (100 angka) agar pekerjaan memfaktorkan n menjadi faktor primanya menjadi sangat sukar bahkan hampir tidak mungkin dapat dilakukan.

Contoh 5.19

Kita akan mengenkripsi pesan dengan algoritma RSA. Mula-mula, bangkitkan sepasang kunci. Sebagai ilustrasi, pilih $a = 47$ dan $b = 71$ (dalam praktik, a dan b harus bilangan yang besar), maka dapat dihitung nilai $n = ab = 3337$ dan $m = (a - 1)(b - 1) = 3220$. Pilih kunci publik $e = 79$ (yang relatif prima dengan 3220 karena pembagi bersama terbesarnya adalah 1). Nilai e dan n dapat dipublikasikan ke umum.

Selanjutnya akan dihitung kunci dekripsi d seperti yang dituliskan pada langkah instruksi 4,

$$e \cdot d \equiv 1 \pmod{m}$$

Dengan menggunakan (5.16) kita menghitung kunci dekripsi d sebagai berikut:

$$d = \frac{1 + (k \times 3220)}{79}$$

Dengan mencoba nilai-nilai $k = 1, 2, 3, \dots$, diperoleh nilai d yang bulat adalah 1019. Ini adalah kunci dekripsi yang harus dirahasiakan.

Misalkan plainteks yang akan dienkripsi adalah $P = \text{HARI INI}$ (atau dalam desimal ASCII-nya adalah 7265827332737873). Pecah P menjadi blok yang lebih kecil, misalnya P dipecah menjadi enam blok yang berukuran 3 digit:

$$\begin{array}{ll} p_1 = 726 & p_4 = 273 \\ p_2 = 582 & p_5 = 787 \\ p_3 = 733 & p_6 = 003 \end{array}$$

Nilai-nilai p_i ini masih terletak di dalam rentang nilai 0 sampai $3337 - 1$. Blok pertama dienkripsi sebagai

$$726^{79} \pmod{3337} = 1,4304567688284660347123409940007 \cdot 10^{226} \pmod{337} = 215 = c_1$$

Blok kedua dienkripsi sebagai

$$582^{79} \pmod{3337} = 776 = c_2$$

Dengan melakukan proses yang sama untuk sisa blok lainnya, dihasilkan cipherteks $C = 215\ 776\ 1743\ 933\ 1731\ 158$.

Proses dekripsi dilakukan dengan menggunakan kunci rahasia $d = 1019$, jadi, blok c_1 didekripsi sebagai

$$215^{1019} \pmod{3337} = 726 = p_1$$

Blok c_2 didekripsi sebagai

$$776^{1019} \pmod{3337} = 582 = p_2$$

Blok plainteks yang lain dikembalikan dengan cara yang serupa. Akhirnya kita memperoleh kembali plainteks semula $P = 7265827332737873$ atau dalam bentuk karakter adalah $P = \text{HARI INI}$. ■

Perhitungan perpangkatan pada proses enkripsi ($c_i = p_i^e \pmod n$) dan dekripsi ($p_i = c_i^d \pmod n$) membutuhkan bilangan yang sangat besar. Untuk menghindari penggunaan bilangan yang besar, maka dapat digunakan penyederhanaan dengan persamaan berikut:

$$ab \pmod m = [(a \pmod m)(b \pmod m)] \pmod m \quad (5.17)$$

Contoh 5.20

Sebagai ilustrasi, untuk menghitung $572^{37} \pmod{713}$ dapat digunakan manipulasi dengan persamaan 3.14 sebagai berikut:

$$572^{37} = 572^{32} \cdot 572^4 \cdot 572$$

$$572^2 \pmod{713} = 327184 \pmod{713} = 630$$

$$572^4 \pmod{713} = 572^2 \cdot 572^2 \pmod{713} = [(572^2 \pmod{713})(572^2 \pmod{713})] \pmod{713} \\ = 630^2 \pmod{713} = 396900 \pmod{713} = 472$$

$$572^8 \pmod{713} = 572^4 \cdot 572^4 \pmod{713} = [(572^4 \pmod{713})(572^4 \pmod{713})] \pmod{713} \\ = 472^2 \pmod{713} = 222784 \pmod{713} = 328$$

$$572^{16} \pmod{713} = 572^8 \cdot 572^8 \pmod{713} = [(572^8 \pmod{713})(572^8 \pmod{713})] \pmod{713} \\ = 328^2 \pmod{713} = 107584 \pmod{713} = 634$$

$$572^{32} \pmod{713} = 572^{16} \cdot 572^{16} \pmod{713} = [(572^{16} \pmod{713})(572^{16} \pmod{713})] \pmod{713} \\ = 634^2 \pmod{713} = 401956 \pmod{713} = 537$$

$$572^{36} \pmod{713} = 572^{32} \cdot 572^4 \pmod{713} = [(572^{32} \pmod{713})(572^4 \pmod{713})] \pmod{713} \\ = 537 \cdot 472 \pmod{713} = 253464 \pmod{713} = 349$$

$$572^{37} \pmod{713} = 572^{36} \cdot 572 \pmod{713} = [(572^{36} \pmod{713})(572 \pmod{713})] \pmod{713} \\ = 349 \cdot 572 \pmod{713} = 199628 \pmod{713} = 701$$

Jadi, $572^{37} \pmod{713} = 701$



Kekuatan dan Keamanan RSA

Seperti yang sudah dikatakan sebelumnya, kekuatan algoritma *RSA* terletak pada tingkat kesulitan dalam memfaktorkan bilangan menjadi faktor primanya, yang dalam hal ini adalah memfaktorkan n menjadi a dan b . Sekali n berhasil difaktorkan menjadi a dan b , maka $m = (a - 1)(b - 1)$ dapat dihitung. Selanjutnya, karena kunci enkripsi e diumumkan (tidak rahasia), maka kunci dekripsi d dapat dihitung dari persamaan $ed \equiv 1 \pmod m$. Ini berarti proses dekripsi dapat dilakukan oleh orang yang tidak berhak.

Penemu algoritma *RSA* menyarankan nilai a dan b panjangnya lebih dari 100 digit. Dengan demikian hasil kali $n = ab$ akan berukuran lebih dari 200 digit. Bayangkanlah berapa besar usaha kerja yang diperlukan untuk memfaktorkan bilangan bulat 200 digit menjadi faktor primanya. Menurut Rivest dan kawan-

kawan, usaha untuk mencari faktor bilangan 200 digit membutuhkan waktu komputasi selama 4 miliar tahun! (dengan asumsi bahwa algoritma pemfaktoran yang digunakan adalah algoritma yang tercepat saat ini dan komputer yang dipakai mempunyai kecepatan 1 milidetik).

Untunglah algoritma yang paling mangkus untuk memfaktorkan bilangan yang besar belum ditemukan. Inilah yang membuat algoritma *RSA* tetap dipakai hingga saat ini. Selagi belum ditemukan algoritma yang mangkus untuk memfaktorkan bilangan bulat menjadi faktor primanya, maka algoritma *RSA* masih direkomendasikan untuk penyandian pesan.

5.11 Fungsi Hash

Data yang disimpan di dalam memori komputer perlu ditempatkan dalam suatu cara sedemikian sehingga pencarinya dapat dilakukan dengan cepat. Setiap data yang berupa record mempunyai *field kunci* yang unik yang membedakan suatu *record* dengan *record* lainnya. Fungsi *hash* (*hash function*) digunakan untuk menempatkan suatu *record* yang mempunyai nilai kunci k . Fungsi *hash* yang paling umum berbentuk

$$h(k) = k \bmod m \quad (5.18)$$

yang dalam hal ini m adalah jumlah lokasi memori yang tersedia (misalkan memori berbentuk sel-sel yang diberi indeks 0 sampai $m - 1$). Fungsi h di atas menempatkan *record* dengan kunci k pada suatu lokasi memori yang beralamat $h(k)$.

Andaikan $m = 11$, sehingga kita mempunyai sel-sel memori yang diberi indeks 0 sampai 10. Kita akan menyimpan data *record* yang masing-masing mempunyai kunci 15, 558, 32, 132, 102, dan 5 [JOH97]. Pada mulanya sel-sel memori dalam keadaan kosong.

Keenam data *record* tersebut masing-masing disimpan pada lokasi yang dihitung sebagai berikut:

$$\begin{aligned}h(15) &= 15 \bmod 11 = 4 \\h(558) &= 558 \bmod 11 = 8 \\h(32) &= 32 \bmod 11 = 10 \\h(132) &= 132 \bmod 11 = 0 \\h(102) &= 102 \bmod 11 = 3 \\h(5) &= 5 \bmod 11 = 5\end{aligned}$$

Keadaan sel-sel memori setelah penyimpanan keenam data *record* tersebut digambarkan seperti berikut ini:

132			102	15	5			558		32
0	1	2	3	4	5	6	7	8	9	10

Karena fungsi *hash* bukanlah fungsi satu-ke-satu (beberapa nilai k yang berbeda dapat menghasilkan nilai $h(k)$ yang sama), maka dapat terjadi **bentrokan** (*collision*) dalam penempatan suatu data *record*. Misalnya kita akan menempatkan data record dengan kunci 257. Perhitungan *hash* menghasilkan

$$h(257) = 257 \bmod 11 = 4$$

padahal sel memori dengan lokasi 4 sudah terisi. Kita katakan telah terjadi bentrokan. Untuk mengatasi bentrokan perlu diterapkan **kebijakan resolusi bentrokan** (*collision resolution policy*). Satu kebijakan resolusi bentrokan adalah mencari sel tak terisi tertinggi berikutnya (dengan 0 diasumsikan mengikuti 10). Jika kita terapkan kebijakan ini, maka data *record* dengan kunci 257 ditempatkan pada lokasi 6.

Jika kita ingin mencari data *record* tertentu, maka kita gunakan fungsi hash kembali. Misalkan kita akan mencari data record dengan kunci p , maka kita hitung $h(p) = p \bmod 11$, misalkan $h(p) = q$. Jika *record* p sama dengan isi sel pada lokasi q , kita katakan lokasi *record* p ditemukan. Sebaliknya, jika *record* p tidak sama dengan isi sel pada lokasi q , maka kita melihat pada posisi tertinggi berikutnya (sekali lagi, 0 diasumsikan mengikuti 10); jika *record* p tidak berada pada posisi ini, kita lihat lagi pada posisi berikutnya, demikian seterusnya. Jika kita mencapai sel kosong atau kembali ke posisi semula, kita simpulkan bahwa *record* p tidak ada.

5.12 International Standard Book Number (ISBN)

Buku-buku yang diterbitkan oleh penerbit resmi selalu disertai dengan kode *ISBN*. Kode *ISBN* terdiri dari 10 karakter, biasanya dikelompokkan dengan spasi atau garis, misalnya 0–3015–4561–9. *ISBN* terdiri atas empat bagian kode: kode yang mengidentifikasi bahasa, kode penerbit, kode yang diberikan secara unik kepada buku tersebut, dan sebuah karakter uji (dapat berupa angka atau huruf X untuk merepresentasikan angka 10). Karakter uji digunakan untuk mevalidasi *ISBN*, tepatnya untuk mendeteksi kesalahan pada karakter *ISBN* atau kesalahan karena perpindahan angka-angkanya. Karakter uji dipilih sedemikian sehingga

$$\sum_{i=1}^{10} ix_i \equiv 0 \pmod{11}$$

yang dalam hal ini x_i adalah karakter yang ke- i di dalam kode ISBN. Untuk mendapatkan karakter uji, kita cukup menghitung

$$\sum_{i=1}^9 ix_i \bmod 11 = \text{karakter uji} \quad (5.17)$$

Untuk kode ISBN 0-3015-4561-8, 0 adalah kode kelompok negara berbahasa Inggris, 3015 adalah kode penerbit, 4561 adalah kode unik untuk buku yang diterbitkan oleh penerbit tersebut, dan 8 adalah karakter uji. Karakter uji ini didapatkan sebagai berikut:

$$1 \cdot 0 + 2 \cdot 3 + 3 \cdot 0 + 4 \cdot 1 + 5 \cdot 5 + 6 \cdot 4 + 7 \cdot 5 + 8 \cdot 6 + 9 \cdot 1 = 151$$

Jadi, karakter ujinya adalah $151 \bmod 11 = 8$. Catatlah bahwa untuk kode ISBN ini,

$$\sum_{i=1}^{10} ix_i = \sum_{i=1}^9 ix_i + 10x_{10} = 151 + 10 \cdot 8 = 231$$

dan $231 \bmod 11 = 0$ atau $231 \equiv 0 \pmod{11}$.

Contoh 5.20

Nomor ISBN sebuah buku terbitan penerbit Indonesia adalah 979-939p-04-5. Tentukan p .

Penyelesaian:

Diketahui karakter uji ISBN = 5. Ini berarti

$$\sum_{i=1}^9 ix_i \bmod 11 = 5$$

Mula-mula hitung

$$\begin{aligned} \sum_{i=1}^9 ix_i &= 1 \cdot 9 + 2 \cdot 7 + 3 \cdot 9 + 4 \cdot 9 + 5 \cdot 3 + 6 \cdot 9 + 7 \cdot p + 8 \cdot 0 + 9 \cdot 4 \\ &= 9 + 14 + 27 + 36 + 15 + 54 + 7p + 0 + 36 = 191 + 7p \end{aligned}$$

Jadi,

$$(191 + 7x) \bmod 11 = 5$$

atau

$$p = \frac{11k + 5 - 191}{7} = \frac{11k - 186}{7}$$

Nilai-nilai k yang menghasilkan x bulat adalah $k = \dots, -6, 1, 8, 15, 22, 28, \dots$. Agar ISBN sah maka p haruslah memenuhi $0 \leq p \leq 9$. Untuk $k = 22$ didapatkan $p = 8$. ■

5.13 Pembangkit Bilangan Acak Semu

Bilangan acak (*random*) banyak digunakan di dalam program komputer, misalnya untuk program simulasi (misalnya mensimulasikan waktu kedatangan nasabah di bank, pompa bensin, dan sebaiknya), program kriptografi, aplikasi statistik, dan sebagainya.

Tidak ada komputasi yang benar-benar menghasilkan deret bilangan acak secara sempurna. Bilangan acak yang dihasilkan dengan rumus-rumus matematika adalah bilangan acak semu (*pseudo*), karena pembangkitan bilangannya dapat diulang kembali. Pembangkit deret bilangan acak semacam itu disebut **pembangkit bilangan acak semu** (*pseudo-random number generator* atau *PRNG*).

Metode yang paling umum digunakan untuk membangkitkan bilangan acak adalah dengan pembangkit bilangan acak kongruen-lanjar (*linear congruential generator* atau *LCG*) adalah *PRNG* yang berbentuk:

$$x_n = (ax_{n-1} + b) \bmod m \quad (5.18)$$

yang dalam hal ini,

x_n = bilangan acak ke- n dari deretnya
 x_{n-1} = bilangan acak sebelumnya
 a = faktor pengali
 b = *increment*
 m = modulus
(a , b , dan m semuanya konstanta)

Kunci pembangkit adalah x_0 yang disebut **umpam** (*seed*).

LCG mempunyai periode tidak lebih besar dari m . Jika a , b , dan m dipilih secara tepat (misalnya b seharusnya relatif prima terhadap m), maka *LCG* akan mempunyai periode maksimal, yaitu $m - 1$.

Contoh 5.21

Bangkitkan bilangan acak dengan menggunakan *LCG*, $m = 17$, $a = 7$, $b = 11$, dan $x_0 = 0$.

Penyelesaian:

Persamaan *LCG* berbentuk

$$x_n = (7x_{n-1} + 11) \bmod 17$$

Lakukan perhitungan sebagai berikut:

$$x_1 = (7x_0 + 11) \bmod 17 = (7 \cdot 0 + 11) \bmod 17 = 11 \bmod 17 = 11$$
$$x_2 = (7x_1 + 11) \bmod 17 = (7 \cdot 11 + 11) \bmod 17 = 88 \bmod 17 = 3$$

dst...

Hasil perhitungan disajikan dalam bentuk tabel seperti di bawah ini:

n	x_n
0	0
1	11
2	3
3	15
4	14
5	7
6	9
7	6
8	2
9	8
10	16
11	4
12	5
13	12
14	10
15	13
16	0
17	11
18	3
19	15
20	14
21	7
22	9
23	6
24	2

Pada $n = 16$, nilai $x_{16} = x_0$, maka bilangan acak berulang kembali. Inilah alasannya mengapa LCG termasuk ke dalam pembangkit bilangan acak semu. ■

5.14 Ragam Soal dan Penyelesaian

Contoh 5.22

- Berapa $-211 \bmod 11$?
- Misalkan $m = -101$ dan $n = 13$. Nyatakan m dan n dalam $m = nq + r$

$$4 = 17(324) + (-32)(172)$$

Jadi, $\text{PB}(324, 172)$ dapat dituliskan sebagai PB dari 324 dan 172 sebagai:

$$\begin{aligned} &= 17(324) + (-32)(172) \\ &= -15(172) + (17)(324 - 172) \\ &= -15(172) + (17)(152) \\ &= 2(152) + (-15)(172 - 152) \\ &= 2(152) + (-15)(20) \\ &= -20 + 2(152 - 7(20)) \\ &= -20 + 2(12) \\ &= 12 - (20 - 12) \\ &= 12 - 1(8) \end{aligned}$$

Substitusi dari persamaan-persamaan di atas:

Sisa pembagian terakhir sebelum 0 adalah 4 , maka $\text{PB}(324, 172) = 4$. Lakukan proses

$$\begin{aligned} 8 &= 1(4) + 0 && (\text{vi}) \\ 12 &= 1(8) + 4 && (\text{v}) \\ 20 &= 1(12) + 8 && (\text{iv}) \\ 152 &= 7(20) + 12 && (\text{iii}) \\ 172 &= 1(152) + 20 && (\text{ii}) \\ 324 &= 1(172) + 152 && (\text{i}) \end{aligned}$$

$$\text{PB}(172, 324) :$$

Lakukan pembagian dengan menggunakan algoritma Euklidian untuk mendapatkan sisa pembagian dengan menggunakan algoritma Euklidian untuk mendapatkan penyelisian:

Nyatakan PB dari 172 dan 324 sebagai kombinasi liniar dari bilangan-bilangan tersebut.

Contoh 5.23



$$\begin{aligned} \text{Jadi, } -101 &= 13(-8) + 3 \\ &= -8 \\ q &= (-101 - 3) / 13 \\ -101 &= 13q + r \end{aligned}$$

$$\begin{aligned} &= 3 \\ &= 13 - 10 \\ &= 13 - (-101 \bmod 13) \\ r &= -101 \bmod 13 \\ -101 &= 13q + r \end{aligned}$$

(ii) Ada banyak cara untuk menyeliskan soal ini, salah satunya adalah sebagai berikut:

$$\begin{aligned} (\text{i}) \quad -211 \bmod 11 &= 11 - ((-211) \bmod 11) = 11 - 2 = 9 \\ \text{Penyelesaian:} & \end{aligned}$$

Contoh 5.24

Buktikan bahwa jika a , b , dan m adalah bilangan bulat sedemikian sehingga $m \geq 2$, dan $a \equiv b \pmod{m}$ dan $c \equiv d \pmod{m}$, maka $ac \equiv bd \pmod{m}$.

Penyelesaian:

$$a \equiv b \pmod{m} \Leftrightarrow a = b + k_1 m$$

$$c \equiv d \pmod{m} \Leftrightarrow c = d + k_2 m$$

$$\begin{aligned} ac &= (b + k_1 m)(d + k_2 m) \\ &= bd + bk_2 m + dk_1 m + k_1 k_2 m^2 \end{aligned}$$

$$= bd + (bk_2 + dk_1 + k_1 k_2 m) m$$

$$ac = bd + Km \quad \text{dengan } K = bk_2 + dk_1 + k_1 k_2 m$$

$$= bd + Km \Leftrightarrow ac \equiv bd \pmod{m}$$
 ■

Contoh 5.25

Misalkan m adalah bilangan bulat positif. Buktikan bahwa jika $a \pmod{m} = b \pmod{m}$, maka $a \equiv b \pmod{m}$.

Penyelesaian:

$$a \pmod{m} = b \pmod{m}$$

$$a + k_1 m = b + k_2 m$$

$$a = b + (k_2 - k_1) m$$

$$a - b = (k_2 - k_1) m$$

Persamaan yang terakhir mengimplikasikan bahwa m habis membagi $(a - b)$, sehingga kita dapat menyatakan bahwa $a \equiv b \pmod{m}$. ■

Contoh 5.26

Sebuah area parkir mempunyai sejumlah *slot* atau *space* yang dinomori 0 sampai 30. Mobil yang hendak parkir di area tersebut ditentukan dengan sebuah fungsi *hash*. Fungsi *hash* tersebut menentukan nomor *slot* yang akan ditempati mobil yang hendak parkir berdasarkan 3 angka terakhir pada plat nomor polisinya.

- Tentukan fungsi *hash* yang dimaksudkan.
- Tentukan nomor *slot* yang ditempati mobil yang datang berturut-turut dengan 3 angka terakhir pada plat nomornya adalah 327, 100, 121, 310, 414, 110, 017

Penyelesaian:

- Fungsi *hash*: $h(x) = x \pmod{31}$

- Nomor slot yang ditempati mobil dihitung dengan fungsi *hash* (a) di atas:

$$h(327) = 327 \pmod{31} = 17$$

$$h(100) = 100 \pmod{31} = 7$$

$$h(121) = 121 \pmod{31} = 28$$

$$h(310) = 310 \pmod{31} = 0$$

$$h(414) = 414 \pmod{31} = 11$$

$$h(110) = 110 \pmod{31} = 17, \text{ karena slot sudah terisi maka isi slot kosong berikutnya, yaitu } 18$$

$$h(017) = 017 \pmod{31} = 17, \text{ karena slot sudah terisi maka } 18, \text{ tetapi karena } 18 \text{ sudah terisi, maka isi slot berikutnya, yaitu } 19$$
 ■

Contoh 5.27

Nomor *ISBN* sebuah buku yang terbaca oleh *bar code* di toko buku adalah 0-07-053965-X. Apakah nomor *ISBN* tersebut sah? Jika tidak, bagaimana seharusnya?

Penyelesaian:

Hal pertama yang harus dilakukan adalah memeriksa apakah benar karakter uji *ISBN* tersebut adalah *X* (representasi angka 10) dengan perhitungan berikut:

$$\sum_{i=1}^9 ix_i \bmod 11 = \text{karakter uji}$$

dalam hal ini,

$$\begin{aligned}\sum_{i=1}^9 ix_i &= 1.0 + 2.0 + 3.7 + 4.0 + 5.5 + 6.3 + 7.9 + 8.6 + 9.5 \\&= 21 + 25 + 18 + 63 + 48 + 45 \\&= 220\end{aligned}$$

sehingga

$$\sum_{i=1}^9 ix_i \bmod 11 = 220 \bmod 11 = 0$$

Hasil perhitungan yang terakhir adalah 0 (bukan 10) sehingga menunjukkan bahwa *X* tidak memenuhi sifat karakter uji nomor *ISBN* di atas. Oleh sebab itu, karakter uji *ISBN* di atas seharusnya bukan *X*, melainkan 0. Dengan demikian, nomor *ISBN* yang tepat adalah 0-07-053965-0.

Soal Latihan

1. Apakah 19 habis membagi bilangan bulat berikut:
(a) 89 (b) 561 (c) 209 (d) 773 (e) 8721
2. Carilah bilangan bulat q dan r sehingga $m = nq + r$
(a) $m = 45, n = 6$ (c) $m = 106, n = 12$ (e) $m = -221, n = 12$
(b) $m = 66, n = 11$ (d) $m = 0, n = 47$ (f) $m = -246, n = 49$
3. Perlihatkan bahwa jika $p \mid q$ dan $r \mid s$, maka $pq \mid rs$.
4. Misalkan m , n , dan c adalah bilangan bulat. Tunjukkan bahwa jika c adalah pembagi bersama terbesar dari m dan n , maka $c \mid (m - n)$.
5. Perlihatkan bahwa jika p , q , dan r bilangan bulat sedemikian sehingga $pr \mid qr$, maka $p \mid q$.
6. Hitung hasil pembagian modulo berikut:
(a) $-173 \text{ mod } 21$ (b) $-340 \text{ mod } 9$
(c) $0 \text{ mod } 34$ (d) $-9821 \text{ mod } 45$
7. Jika m bilangan bulat positif, perlihatkan bahwa $a \text{ mod } m \equiv b \text{ mod } m$ jika $a \equiv b \pmod{m}$.
8. Perlihatkan bahwa jika $a \equiv b \pmod{m}$ dan $c \equiv d \pmod{m}$, yang dalam hal ini a , b , c , d , dan m adalah bilangan bulat, maka $a - c \equiv (b - d) \pmod{m}$.
9. Buktikan bahwa jika a , b , k , dan m adalah bilangan bulat sedemikian sehingga $k \geq 1$, $m \geq 2$, dan $a \equiv b \pmod{m}$, maka $a^k \equiv b^k \pmod{m}$.
10. Misalkan a , dan b bilangan bulat dan m dan n adalah bilangan bulat positif lebih besar dari 1. Buktikan bahwa jika $n \mid m$ dan $a \equiv b \pmod{m}$, maka $a \equiv b \pmod{n}$.
11. Tunjukkan bahwa jika a , b , dan m bilangan bulat sedemikian sehingga $m \geq 2$, dan $a \equiv b \pmod{m}$, maka $\text{PBB}(a, m) = \text{PBB}(b, m)$.
12. Tentukan PBB dari pasangan bilangan bulat a dan b berikut:
(a) 220, 1400 (b) 315, 825
(c) 110, 273 (d) 2475, 32670 (e) -456, 688

13. Tentukan inversi (*invers*) dari a modulo m jika $a = -39$ dan $m = 14$.
14. Nyatakan PBB dari soal nomor 7 di atas dalam bentuk kombinasi lanjar $ma + nb$.
15. Tuliskan 5 buah bilangan bulat yang kongruen dengan 4 modulo 12.
16. Tentukan pasangan bilangan bulat yang relatif prima satu sama lain:
(a) 21, 34, 55 (b) 25, 41, 49, 64 (c) 17, 18, 19, 23
17. Andaikan bahwa a dan b bilangan bulat positif. Tunjukkan bahwa $\text{PBB}(a, b) = \text{PBB}(a, a + b)$.
18. Pecahkan kekongruenan lanjar berikut:
(a) $4x \equiv 5 \pmod{8}$ (b) $2x \equiv 7 \pmod{17}$ (c) $5x \equiv 10 \pmod{12}$
19. Tentukan inversi (*invers*) dari a modulo m berikut:
(a) $a = 34, m = 5$ (b) $a = 178, m = 62$ (c) $a = -341, m = 17$
20. Tunjukkan bahwa $2^{340} \equiv 1 \pmod{11}$ dengan Teorema Fermat dan memperhatikan bahwa $2^{340} = (2^{10})^{34}$.
21. Misalkan Julius Caesar mengenkripsi pesan dengan cara menggeser huruf abjad 8 posisi ke kanan.
 - (a) Nyatakan fungsi matematik yang memetakan plainteks ke cipherteks dengan metode di atas.
 - (b) Misalkan $A = 0, B = 1, \dots, Z = 25$. Tentukan cipherteks dari pesan “DIRUMAH” dengan fungsi tsb.
22. Enkripsi pesan HELLO WORLD dengan algoritma RSA dan menggunakan nilai-nilai $a = 23, b = 31$, dan $e = 29$.
23. Sembilan angka pertama dari kode ISBN sebuah buku adalah 0-07-053965. Tentukan karakter uji untuk buku ini.
24. ISBN sebuah buku mengenai algoritma adalah 0-201-57p859-1, yang dalam hal ini p adalah angka. Berapa nilai p ?
25. Tunjukkan bagaimana sekumpulan data dengan kunci-kunci sebagai berikut: 714, 631, 26, 373, 775, 906, 509, 2032, 42, 4, 136, 1028 ditempatkan di dalam memori dengan fungsi hash $h(k) = k \pmod{17}$.

26. Tentukan bilangan acak yang dihasilkan oleh $x_{n+1} = (4x_n + 1) \pmod{7}$ dengan umpan $x_0 = 7$.
27. Tentuka solusi dari sistem kekongruenan berikut: $x \equiv 5 \pmod{6}$, $x \equiv 3 \pmod{10}$, $x \equiv 8 \pmod{13}$.

BAB 6

Kombinatorial dan Peluang Diskrit

Hidup adalah penjumlahan semua pilihan yang ada.
(Albert Camus)

Kombinatorial (*combinatoric*) adalah cabang matematika yang mempelajari pengaturan objek-objek. Solusi yang ingin kita peroleh dengan kombinatorial ini adalah jumlah cara pengaturan objek-objek tertentu di dalam himpunannya. Tiga buah contoh ilustrasi berikut dikemukakan untuk memperjelas masalah seperti apa yang akan dipecahkan dengan kombinatorial.

- (i) Contoh pertama, misalkan nomor plat mobil di negara X terdiri atas 5 angka angka diikuti dengan 2 huruf. Angka pertama tidak boleh 0. Berapa banyak nomor plat mobil yang dapat dibuat?
- (ii) Contoh kedua, sandi-lewat (*password*) sistem komputer panjangnya enam sampai delapan karakter. Tiap karakter boleh berupa huruf atau angka; huruf besar dan huruf kecil tidak dibedakan. Berapa banyak sandi-lewat yang dapat dibuat?

- (iii) Dari 20 anggota Fraksi X di DPR, akan dibentuk sebuah komisi yang beranggotakan 6 orang. Berapa banyak cara memilih anggota komisi bila seorang anggota yang bernama A harus termasuk di dalam komisi tersebut?

Cara yang paling sederhana untuk menyelesaikan persoalan semacam di atas adalah dengan mengenumerasi semua kemungkinan jawabannya. Mengenumerasi artinya mencacah atau menghitung (*count*) satu persatu setiap kemungkinan jawaban. Untuk persoalan dengan jumlah objek sedikit, mengenumerasi setiap kemungkinan jawaban masih dapat dilakukan, tetapi untuk persoalan dengan jumlah objek yang banyak, cara enumerasi jelas tidak sangkil. Misalnya pada persoalan contoh pertama, bila kita mengenumerasi semua kemungkinan jawabannya adalah seperti di bawah ini:

12345AB
12345AC
12345BC
...
34567MT
34568ML
...
dan seterusnya...

Mungkin kita sudah lelah sebelum usaha mengenumerasi semua kemungkinan nomor plat mobil selesai, karena nomor plat mobil yang dibentuk sangat banyak. Di sinilah peran kombinatorial, yang merupakan “seni berhitung”, menyelesaikan persoalan semacam ini dengan cepat. Kombinatorial dapat digunakan untuk menjawab persoalan semacam ini tanpa kita perlu mengenumerasi semua kemungkinan jawabannya. Hal ini dapat dilakukan karena di dalam kombinatorial terdapat kaidah dasar menghitung (akan dijelaskan di dalam upabab 6.2). Dengan kaidah ini, berbagai persoalan menghitung jumlah cara pengaturan objek dapat diselesaikan. Kita juga menggunakan kaidah dasar menghitung untuk mengetahui berapa jumlah kemungkinan sandi-lewat yang harus dicoba oleh seorang penyusup (*intruder*) untuk memasuki sebuah sistem komputer. Terakhir, kombinatorial digunakan pada teori peluang diskrit untuk menghitung peluang suatu kejadian terjadi.

6.1 Percobaan

Kombinatorial didasarkan pada hasil yang diperoleh dari dari suatu percobaan (*experiment*). Percobaan adalah proses fisik yang hasilnya dapat diamati.

Contoh-contoh percobaan dan hasilnya (*outcome*) adalah:

1. Melempar dadu
Enam hasil percobaan yang mungkin untuk pelemparan dadu adalah muka dadu 1, 2, 3, 4, 5, atau 6
2. Melempar koin uang Rp100
Hasil percobaan melempar koin 100 ada dua kemungkinan: muka koin yang bergambar rumah gadang atau muka koin yang bergambar wayang
3. Memilih lima orang wakil dari 100 orang mahasiswa
Hasil yang diperoleh adalah perwakilan yang beranggotakan lima orang mahasiswa. Kemungkinan perwakilan yang dapat dibentuk banyak sekali.
4. Menyusun jumlah kata yang panjangnya 5 huruf yang dapat dibentuk dari huruf-huruf a, b, c, d, e , tidak boleh ada huruf yang berulang di dalam kata.
Hasil yang diperoleh adalah string yang disusun oleh huruf-huruf tersebut, misalnya $abcde, abced, acdeb$, dan seterusnya.

6.2 Kaidah Dasar Menghitung

Di dalam kombinatorial, kita harus menghitung (*counting*) semua kemungkinan pengaturan objek. Dua kaidah dasar yang digunakan sebagai teknik menghitung dalam kombinatorial adalah **kaidah perkalian** (*rule of product*) dan **kaidah penjumlahan** (*rule of sum*). Kedua kaidah ini dapat digunakan untuk memecahkan banyak masalah persoalan menghitung.

1. Kaidah perkalian (*rule of product*)

Bila percobaan 1 mempunyai p hasil percobaan yang mungkin terjadi (atau menghasilkan p kemungkinan jawaban), percobaan 2 mempunyai q hasil percobaan yang mungkin terjadi (atau menghasilkan q kemungkinan jawaban), maka bila percobaan 1 dan percobaan 2 dilakukan, maka terdapat $p \times q$ hasil percobaan (atau menghasilkan $p \times q$ kemungkinan jawaban).

2. Kaidah penjumlahan (*rule of sum*)

Bila percobaan 1 mempunyai p hasil percobaan yang mungkin terjadi (atau menghasilkan p kemungkinan jawaban), percobaan 2 mempunyai q hasil percobaan yang mungkin terjadi (atau menghasilkan q kemungkinan jawaban), maka bila hanya satu percobaan saja yang dilakukan (percobaan 1 atau percobaan 2), terdapat $p + q$ kemungkinan hasil percobaan (menghasilkan $p + q$ kemungkinan jawaban) yang mungkin terjadi.

Perhatikanlah kata yang digarisbawahi pada kedua di atas: dan serta atau. Kedua kata ini adalah kata kunci untuk mengidentifikasi apakah suatu persoalan menghitung diselesaikan dengan kaidah perkalian atau kaidah penjumlahan. Kaidah perkalian

menyatakan bahwa kedua percobaan dilakukan secara simultan atau serempak, sedangkan pada kaidah penjumlahan, kedua percobaan dilakukan tidak simultan.

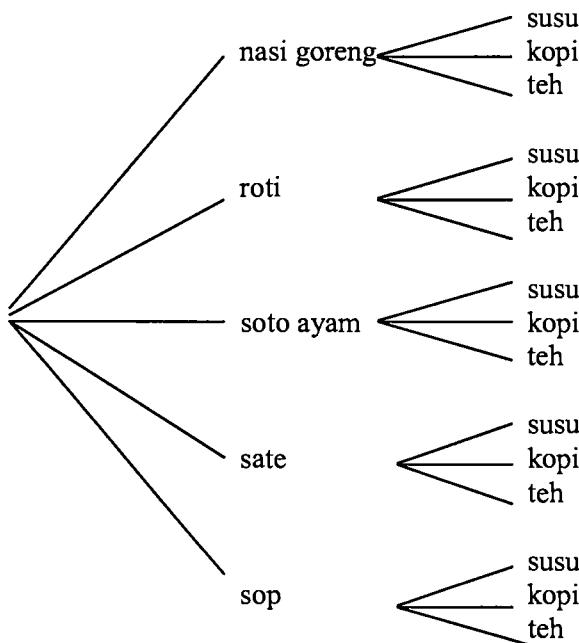
Contoh 6.1 sampai 6.7 berikut mempelihatkan penggunaan kaidah perkalian dan kaidah penjumlahan untuk menghitung pengaturan objek-objek. Kita harus dapat menganalisis kapan menggunakan kaidah perkalian dan kaidah penjumlahan.

Contoh 6.1

Sebuah restoran menyediakan lima jenis makanan, misalnya nasi goreng, roti, soto ayam, sate, dan sop, serta tiga jenis minuman, misalnya susu, kopi, dan teh. Jika setiap orang boleh memesan satu makanan dan satu minuman, berapa banyak pasangan makanan dan minuman yang dapat dipesan?

Penyelesaian:

Kita dapat menggunakan diagram pohon untuk menentukan jumlah pasangan makanan yang minuman yang dapat dipesan. Pada diagram pohon tersebut, akar adalah awal pemilihan, cabang adalah alternatif solusi dan daun merupakan akhir solusi.



Berdasarkan diagram pohon di atas, kita mengenumerasi semua kemungkinan pasangan makanan dan minuman yang dapat dipesan, yaitu:

<i>nasi goreng dan susu</i>	<i>roti dan susu</i>	<i>soto ayam dan susu</i>	<i>sate dan susu</i>	<i>sop dan susu</i>
<i>nasi goreng dan kopi</i>	<i>roti dan kopi</i>	<i>soto ayam dan kopi</i>	<i>sate dan kopi</i>	<i>sop dan kopi</i>
<i>nasi goreng dan teh</i>	<i>roti dan teh</i>	<i>soto ayam dan teh</i>	<i>sate dan teh</i>	<i>sop dan teh</i>

semuanya ada 15 pasang.

Dalam kombinatorial, kita memandang bahwa dalam kejadian ini orang harus memilih makanan dan minuman. Ada 5 kemungkinan memilih makanan, yaitu nasi goreng, roti, soto ayam, sate, dan sop. Ada 3 kemungkinan memilih minuman, yaitu susu, kopi, dan teh, sehingga dengan menggunakan kaidah perkalian, jumlah kemungkinan pasangan makanan dan minuman yang dapat dipesan adalah $5 \times 3 = 15$ pasang. ■

Contoh 6.2

Jabatan ketua himpunan dapat diduduki oleh mahasiswa angkatan tahun 1997 atau angkatan tahun 1998. Jika terdapat 45 orang mahasiswa angkatan 1997 dan 52 orang mahasiswa angkatan 1998, berapa cara memilih penjabat ketua himpunan?

Penyelesaian:

Jabatan yang ditawarkan hanya ada satu, yang dapat diduduki oleh salah seorang mahasiswa dari dua angkatan yang ada. Ada 45 cara memilih satu orang mahasiswa dari Angkatan 1997, dan 52 cara memilih satu orang dari Angkatan 1998, namun hanya satu dari kedua angkatan itu yang terpilih (angkatan 1997 atau angkatan 1998). Dalam kombinatorial, dari kedua kajadian, hanya satu dari dua kejadian yang dilakukan, sehingga dengan menggunakan kaidah penjumlahan, jumlah cara memilih penjabat ketua himpunan tersebut sama dengan jumlah mahasiswa pada kedua angkatan, yaitu $45 + 52 = 97$ cara. ■

Contoh 6.3

Sekelompok mahasiswa terdiri atas 4 orang pria dan 3 orang wanita. Berapa jumlah cara memilih satu orang wakil pria dan satu orang wakil wanita?

Penyelesaian:

Ada 4 kemungkinan memilih satu wakil pria, dan 3 kemungkinan memilih satu wakil wanita. Jika dua orang wakil harus dipilih, masing-masing 1 pria dan 1 wanita, maka jumlah kemungkinan perwakilan yang dapat dipilih adalah $4 \times 3 = 12$. ■

Contoh 6.4

Sekelompok mahasiswa terdiri atas 4 orang pria dan 3 orang wanita. Berapa jumlah cara memilih satu orang yang mewakili kelompok tersebut (tidak peduli pria atau wanita)?

Penyelesaian:

Ada 4 kemungkinan memilih satu wakil pria, dan 3 kemungkinan memilih satu wakil wanita. Jika hanya satu orang wakil yang harus dipilih (pria atau wanita), maka jumlah kemungkinan wakil yang dapat dipilih adalah $4 + 3 = 7$. ■

Contoh 6.5

Misalkan himpunan $A = \{a, b, c, d, e\}$ dan himpunan $B = \{1, 2, 3\}$. Berapa banyak pasangan terurut (*ordered pairs*) yang dapat dibentuk antara anggota himpunan A dengan anggota himpunan B (yaitu $A \times B$) ?

Penyelesaian:

Nyatakan pasangan terurut sebagai (a, b) , yang dalam hal ini $a \in A$ dan $b \in B$. Karena anggota himpunan A yang mungkin menjadi elemen pertama pasangan ada 5 buah dan anggota himpunan B yang mungkin menjadi elemen kedua pasangan ada 3 buah, maka banyaknya pasangan yang dapat terbentuk adalah $5 \times 3 = 15$, yang merupakan jumlah elemen himpunan hasil operasi $A \times B$ (*cartesian product*). ■

Contoh 6.6

Kursi-kursi di dalam ruang aula akan diberi nomor dengan sebuah huruf diikuti dengan bilangan bulat positif yang tidak lebih dari 50 (misalnya A12, B36, dan seterusnya). Berapa jumlah maksimum kursi yang dapat dinomori?

Penyelesaian:

Ada 26 kemungkinan memilih huruf alfabet untuk nomor kursi dan 50 kemungkinan bilangan bulat positif yang dapat digunakan. Huruf alfabet dan bilangan bulat keduanya harus digunakan untuk penomoran. Jumlah penomoran kursi yang dapat dibuat adalah $26 \times 50 = 1300$. Jadi, jumlah maksimum kursi yang dinomori adalah 1300 buah. ■

Contoh 6.7

Terdapat empat rute yang dapat dilalui kendaraan dari Jakarta ke Bandung, dan tiga rute dari Bandung ke Yogyakarta.

- Berapa banyak cara seseorang bepergian dengan kendaraan dari Jakarta ke Yogyakarta melalui Bandung?
- Berapa banyak cara seseorang bepergian pulang-pergi dengan kendaraan dari Jakarta ke Yogyakarta melalui Bandung?

Penyelesaian:

- Sesorang dari Jakarta ke Yogyakarta harus melewati rute Jakarta-Bandung dan rute Bandung-Yogyakarta. Ada 4 pilihan rute dari Jakarta ke Bandung dan 3 pilihan rute dari Bandung ke Yogyakarta, sehingga jumlah pilihan rute dari Jakarta ke Yogyakarta via Bandung adalah $4 \times 3 = 12$.
- Ada 12 rute dari Jakarta ke Yogyakarta via Bandung dan 12 rute dari Yogyakarta ke Jakarta via Bandung. Karena perjalanan pulang-pergi (Jakarta-Yogyakarta dan Yogyakarta-Jakarta), maka jumlah pilihan rute seluruhnya adalah $12 \times 12 = 144$ cara untuk berkendaraan pulang-pergi. ■

6.3 Perluasan Kaidah Menghitung

Kaidah perkalian dan kaidah penjumlahan di atas dapat diperluas hingga mengandung lebih dari dua buah percobaan. Jika n buah percobaan masing-masing mempunyai p_1, p_2, \dots, p_n , hasil percobaan yang mungkin terjadi yang dalam hal ini setiap p_i

tidak bergantung pada pilihan sebelumnya, maka jumlah hasil percobaan yang mungkin terjadi adalah:

- (a) $p_1 \times p_2 \times \dots \times p_n$ untuk kaidah perkalian.
 - (b) $p_1 + p_2 + \dots + p_n$ untuk kaidah penjumlahan
-

Contoh 6.8

Jika ada sepuluh pertanyaan yang masing-masing bisa dijawab benar atau salah (B atau S), berapakah kemungkinan kombinasi jawaban yang dapat dibuat ?

Penyelesaian:

Andaikan 10 pertanyaan tersebut sebagai 10 buah kotak, masing-masing kotak hanya berisi 2 kemungkinan jawaban, B atau S:

B/S —									
1	2	3	4	5	6	7	8	9	10

Di sini kita menggunakan kaidah perkalian, karena kesepuluh kotak ini harus terisi dengan jawaban B atau S (kotak 1 dan kotak 2 dan kotak 3 dan ... dan kotak 10). Jumlah kombinasi jawaban yang dapat dibuat:

$$(2)(2)(2)(2)(2)(2)(2)(2)(2)(2) = 2^{10}.$$

Contoh 6.9

- (a) Berapa banyak jumlah kata 5-huruf yang dapat dibentuk dari huruf-huruf *a, b, c, d, e* jika tidak boleh ada huruf yang berulang di dalam kata.
- (b) Berapa banyak jumlah kata 5-huruf yang dapat dibentuk dari huruf-huruf *a, b, c, d, e* jika pengulangan huruf diperbolehkan.
- (c) Berapa banyak jumlah kata pada jawaban soal (a) yang diawali oleh huruf *a*?
- (d) Berapa banyak jumlah kata pada jawaban soal (a) yang tidak diawali oleh huruf *a*?

Penyelesaian:

- (a) Andaikan posisi 5 huruf di dalam kata sebagai 5 buah kotak. Kotak pertama dapat diisi dengan salah satu dari 5 huruf (jadi, ada 5 cara). Kotak dapat diisi dengan 4 cara (karena 1 huruf sudah dipakai untuk kotak pertama). Kotak ketiga dapat diisi dengan 3 huruf (karena 2 huruf lain sudah dipakai untuk kotak pertama dan kedua). Kotak keempat dapat diisi dengan 2 cara dan kotak kelima dapat diisi dengan 1 cara.

5 cara 4 cara 3 cara 2 cara 1 cara

Karena setiap kotak harus diisi dengan 1 huruf (kotak 1 dan kotak 2 dan kotak 3 dan kotak 4 dan kotak 5), maka kita menggunakan kaidah perkalian. Jumlah kata yang dapat dibentuk adalah $5 \times 4 \times 3 \times 2 \times 1 = 120$ buah.

- (b) Jika pengulangan huruf dibolehkan di dalam kata, maka setiap kotak dapat diisi dengan 5 cara. Maka, jumlah kata yang dapat disusun adalah $5 \times 5 \times 5 \times 5 \times 5 = 5^5 = 3125$.
- (c) Kotak 1 hanya dapat diisi dengan 1 cara (yaitu huruf a). Kotak kedua 4 cara (selain huruf a), kotak ketiga 3 cara, kotak keempat 2 cara, dan kotak kelima 1 cara. Maka, jumlah kata yang dapat disusun adalah $1 \times 4 \times 3 \times 2 \times 1 = 24$.
- (d) Kotak 1 hanya dapat diisi dengan 4 cara (selain huruf a). Kotak kedua 4 cara (1 huruf sudah dipakai untuk kotak pertama, tersisa 4 huruf, termasuk huruf a), kotak ketiga 3 cara, kotak keempat 2 cara, dan kotak kelima 1 cara. Maka, jumlah kata yang dapat disusun adalah $4 \times 4 \times 3 \times 2 \times 1 = 96$. Kita juga dapat menyelesaikan soal (d) ini dengan melihat bahwa jumlah kata yang tidak diawali oleh huruf a adalah jumlah semua kata dikurangi dengan jumlah kata yang diawali oleh huruf a , yaitu jawaban (a) dikurangi dengan jawaban (c), $120 - 24 = 96$. ■

Contoh 6.10

Berapa Perpustakaan memiliki 6 buah buku berbahasa Inggris, 8 buah buku berbahasa Perancis, dan 10 buah buku berbahasa Jerman. Masing-masing buku berbeda judulnya. Berapa jumlah cara memilih (a) 3 buah buku, masing-masing dari tiap bahasa berbeda, dan (b) 1 buah buku (sembarang bahasa).

Penyelesaian:

- (a) Jumlah cara memilih 3 buah buku, masing-masing dari tiap bahasa adalah $(6)(8)(10) = 480$ cara.
- (b) Jumlah cara memilih 1 buah buku (sembarang bahasa) = $6 + 8 + 10 = 24$ cara. ■

Contoh 6.11

Berapa banyak bilangan ganjil antara 1000 dan 9999 (termasuk 1000 dan 9999 itu sendiri) yang (i) semua angkanya berbeda, dan (ii) boleh ada angka yang berulang.

Penyelesaian:

- (i) Karena yang diminta adalah bilangan ganjil, kita harus memulai dari angka satuan terlebih dahulu, baru kemudian dari angka ribuan, ratusan, dan puluhan).

untuk posisi satuan: ada 5 kemungkinan angka (yaitu 1, 3, 5, 7 dan 9);

untuk posisi ribuan: ada 8 kemungkinan angka (yaitu 1 sampai 9, kecuali yang sudah dipakai untuk angka satuan);

untuk posisi ratusan: ada 8 kemungkinan angka (yaitu 0 sampai 9, kecuali dua angka yang sudah dipakai untuk angka satuan dan angka ribuan);

untuk posisi puluhan: ada 7 kemungkinan angka (yaitu 0 sampai 9, kecuali tiga angka yang sudah dipakai untuk angka satuan dan angka ribuan);

Banyak bilangan ganjil seluruhnya = $(5)(8)(8)(7) = 2240$ buah.

- (ii) Jika perulangan angka dibolehkan, maka untuk posisi satuan tetap ada 5 kemungkinan angka, untuk posisi ribuan ada 9 kemungkinan angka (1 sampai 9), untuk posisi ratusan ada 10 kemungkinan (0 sampai 9), dan untuk posisi puluhan ada 10 kemungkinan (0 sampai 9). Banyak bilangan ganjil seluruhnya adalah $(5)(9)(10)(10) = 4500$ buah. ■

Contoh 6.12

[ROS99]. Berapa nilai k sesudah kode program Pascal berikut dieksekusi?

```
k := 0
for p1 := 1 to n1 do
    k := k + 1;
for p2 := 1 to n2 do
    k := k + 1;
:
for pm := 1 to nm do
    k := k + 1;
```

Penyelesaian:

Program di atas memiliki m buah kalang (pengulangan) *for*. Setiap kalang ke- i ($i = 1, 2, \dots, m$) dieksekusi sebanyak n_i kali. Pada setiap kalang, nilai k selalu ditambah 1 (nilai k pada awalnya 0). Karena setiap kalang dilaksanakan tidak secara bersamaan, maka nilai k dapat dihitung dengan kaidah penjumlahan. Nilai k di akhir program sama dengan berapa kali seluruh kalang dieksekusi. Jadi, $k = n_1 + n_2 + \dots + n_m$. ■

Contoh 6.13

[ROS99]. Berapa nilai k sesudah kode program Pascal berikut dieksekusi?

```
k := 0
for p1 := 1 to n1 do
    for p2 := 1 to n2 do
        :
    for pm := 1 to nm do
        k := k + 1;
```

Penyelesaian:

Program di atas memiliki m buah kalang (pengulangan) *for-do* bersarang (*nested*). Setiap kalang ke- i ($i = 1, 2, \dots, m$) dieksekusi sebanyak n_i kali. Pada setiap kalang, nilai k selalu ditambah 1 (nilai k pada awalnya 0). Karena setiap kalang dilaksanakan secara bersamaan, maka nilai k dapat dihitung dengan kaidah perkalian. Nilai k di akhir program sama dengan berapa kali seluruh kalang dieksekusi. Jadi, $k = n_1 \times n_2 \times \dots \times n_m$. ■

Contoh 6.14

Salah satu terapan kombinatorial adalah dalam bidang kriptografi. Misalnya pesan-jelas (*plaintext*) "Informatika" dengan menggunakan algoritma kriptografi tertentu disandikan menjadi pesan-tersandi (*chipertext*) "%r\$ht&90dt". Melalui proses yang berkebalikan,

pesan-tersandi dapat dikembalikan menjadi pesan-jelas. Algoritma kriptografi *DES* (*Data Encryption Standard*) menggunakan kunci (*key*) untuk menyandikan pesan yang akan dikirim melalui saluran komunikasi. Panjang kunci *DES* adalah delapan karakter atau 64 bit. Orang yang ingin memecahkan pesan-tersandi (*chiphertext*) menjadi pesan-jelas (*plaintext*) harus mencoba seluruh kemungkinan kunci yang panjangnya 64 bit itu. Berapa banyak kemungkinan kunci yang harus dicoba untuk memecahkan *chiphertext*?

Penyelesaian:

Karena ada 64 posisi pengisian bit yang masing-masing memiliki dua kemungkinan nilai, 0 atau 1, maka jumlah kombinasi kunci yang harus dicoba adalah:

$$(2)(2)(2)(2)(2) \dots (2)(2) \text{ (sebanyak 64 kali)} = 2^{64} = 18.446.744.073.709.551.616$$

Andaikan tersedia komputer dengan sejuta prosesor paralel yang dapat mencoba satu juta kunci setiap detik untuk memecahkan *chiphertext*, maka dibutuhkan waktu sekitar 584.942 tahun untuk mencoba seluruh kemungkinan kunci (*brute force attack*). Inilah yang merupakan kekuatan algoritma DES, yaitu terletak pada usaha yang sangat sulit untuk menemukan kunci kriptografi. ■

Beberapa persoalan kombinatorial yang lebih kompleks dapat diselesaikan dengan menggunakan kaidah dasar menghitung di atas. Beberapa persoalan tidak dapat diselesaikan dengan satu kaidah saja, tetapi kita harus menggunakan dua kaidah sekaligus. Kedua hal ini diilustrasikan pada Contoh 6.15 sampai Contoh 6.17 berikut.

Contoh 6.15

Suatu bilangan dibentuk dari angka-angka 2, 3, 4, 5, 7, 8, dan 9. Misalkan pengulangan angka tidak dibolehkan. Berapa banyak bilangan 4-angka yang kurang dari 5000 namun habis dibagi 5 yang dapat dibentuk dari angka-angka tersebut? ■

Penyelesaian:

Ada 4 angka bilangan yang akan dibentuk: _____

Karena disyaratkan bilangan kelipatan 5, maka angka paling kanan hanya dapat diisi dengan angka 5 saja (satu cara).

Angka posisi ke-1 dapat diisi dengan 3 cara (yaitu 2, 3, dan 4).

Angka posisi ke-2 dapat diisi dengan 5 cara (2 angka lain sudah dipakai untuk posisi ke-1 dan ke-4).

Angka posisi ke-3 dapat diisi dengan 4 cara (3 angka lain sudah dipakai untuk posisi ke-1, ke-2 dan ke-4).

Karena seluruh posisi angka harus terisi, maka kita menggunakan kaidah perkalian untuk menghitung jumlah bilangan bulat yang dapat dibentuk, yaitu $3 \times 5 \times 4 \times 1 = 60$ buah. ■

Contoh 6.16

Lihat kembali contoh ilustrasi pada awal bab ini. Sandi-lewat (*password*) pada sistem komputer panjangnya enam sampai delapan karakter. Tiap karakter boleh berupa huruf atau angka; huruf besar dan huruf kecil tidak dibedakan. Berapa banyak sandi-lewat yang dapat dibuat?

Penyelesaian:

Banyaknya huruf alfabet adalah 26 (A-Z) dan banyak angka desimal adalah 10 (0-9), jadi seluruhnya 36 karakter. Masing-masing huruf atau angka dapat menjadi pilihan untuk posisi karakter di dalam *password*.

Untuk sandi-lewat dengan panjang 6 karakter, jumlah kemungkinan sandi-lewat adalah

$$(36)(36)(36)(36)(36)(36) = 36^6 = 2.176.782.336$$

untuk sandi-lewat dengan panjang 7 karakter, jumlah kemungkinan sandi-lewat adalah

$$(36)(36)(36)(36)(36)(36)(36) = 36^7 = 78.364.164.096$$

dan untuk sandi-lewat dengan panjang 8 karakter, jumlah kemungkinan sandi-lewat adalah

$$(36)(36)(36)(36)(36)(36)(36)(36) = 36^8 = 2.821.109.907.456$$

Dengan menggunakan kaidah penjumlahan, jumlah seluruh sandi-lewat adalah $2.176.782.336 + 78.364.164.096 + 2.821.109.907.456 = 2.901.650.833.888$ buah. ■

Contoh 6.17

Lihat kembali contoh ilustrasi pada awal bab ini. Misalkan nomor plat mobil di negara X terdiri atas 5 angka angka diikuti dengan 2 huruf. Angka pertama tidak boleh 0. Berapa banyak nomor plat mobil yang dapat dibuat?

Penyelesaian:

Ada 7 karakter di dalam susunan plat mobil (5 angka dan 2 huruf): _____
Perhatikan bahwa huruf dan angka boleh digunakan berulang (asumsi ini kita ambil kecuali jika disebutkan secara khusus bahwa pengulangan tidak dibolehkan). Ada 9 pilihan angka untuk mengisi karakter pertama (karena 0 tidak dibolehkan). Karakter kedua dapat diisi dengan 10 pilihan angka (0 diperbolehkan). Karakter ketiga 10 pilihan angka. Karakter keempat 10, dan karakter kelima 10 pilihan angka. Karakter keenam dapat diisi dengan 26 pilihan huruf, dan karakter ketujuh dapat diisi dengan 26 kemungkinan huruf. Jumlah nomor plat mobil yang dapat dibuat adalah $9 \times 10 \times 10 \times 10 \times 10 \times 26 \times 26 = 60.840.000$ buah. ■

6.4 Prinsip Inklusi-Eksklusi

Sekarang kita akan melihat contoh penggunaan prinsip inklusi-eksklusi untuk menghitung kombinatorial. Seperti kita ketahui informasi terkecil yang dapat disimpan di dalam memori komputer adalah *byte*. Setiap *byte* disusun oleh 8-bit. Berapa banyak jumlah *byte* yang dimulai dengan ‘11’ atau berakhir dengan ‘11’? Penyelesaiannya adalah sebagai berikut:

Misalkan

$$A = \text{himpunan byte yang dimulai dengan '11'},$$

$$B = \text{himpunan byte yang diakhiri dengan '11'}$$

$$A \cap B = \text{himpunan byte yang berawal dan berakhir dengan '11'}$$

maka

$$A \cup B = \text{himpunan byte yang berawal dengan '11' atau berakhir dengan '11'}$$

Jumlah *byte* yang dimulai dengan ‘11’ adalah $2^6 = 64$ buah, karena 2 posisi pertama sudah diisi dengan ‘11’, sehingga kita cukup mengisi 6 posisi bit sisanya. Jadi $|A| = 64$.

Dengan cara yang sama, jumlah *byte* yang diakhiri dengan ‘11’ adalah $2^6 = 64$ buah. Jadi, $|B| = 64$.

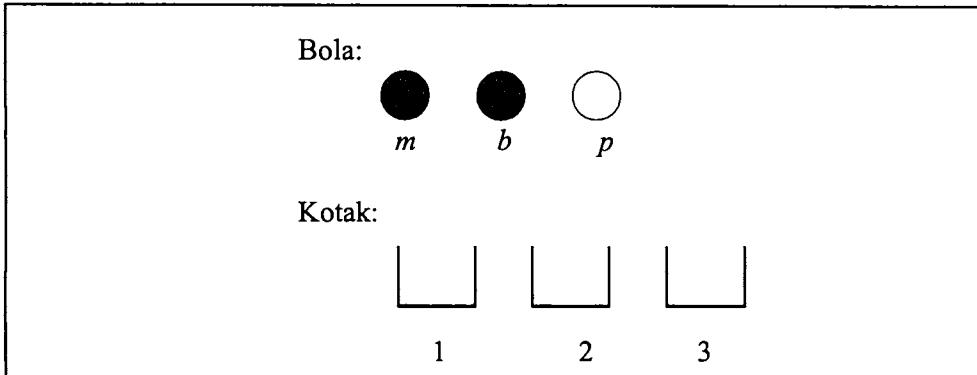
Jumlah *byte* yang berawal dan berakhir dengan ‘11’ ada $2^4 = 16$ buah, karena 2 posisi pertama dan 2 posisi terakhir sudah diisi dengan ‘11’, sehingga kita tinggal mengisi 4 posisi bit di tengah saja. Jadi, $|A \cap B| = 16$.

Dengan menggunakan prinsip inklusi-eksklusi, maka jumlah *byte* yang dimulai dengan ‘11’ atau berakhir dengan ‘11’ adalah sebanyak

$$|A \cup B| = |A| + |B| - |A \cap B| = 2^6 + 2^6 - 16 = 64 + 64 - 16 = 112 \text{ buah.}$$

6.5 Permutasi

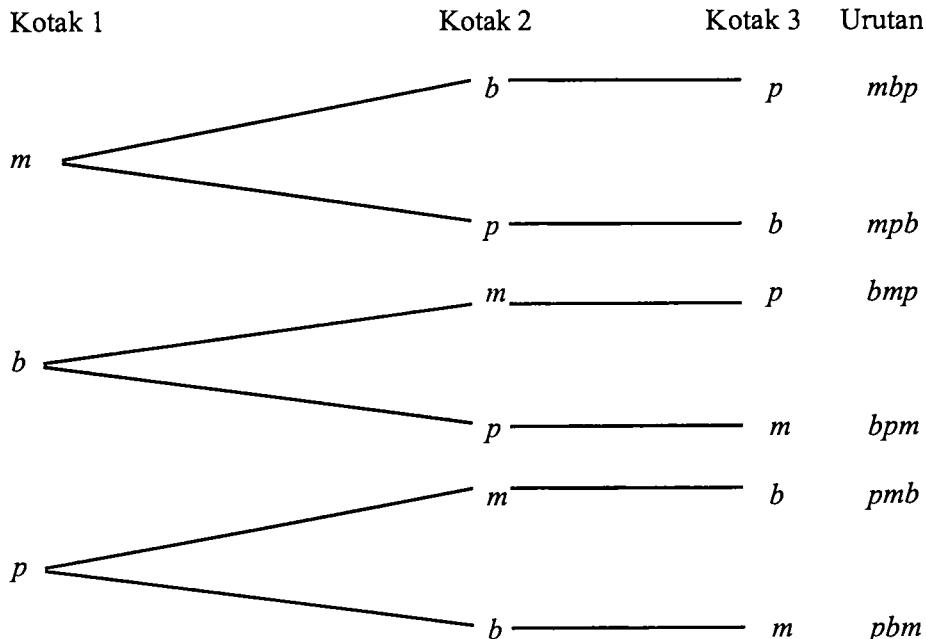
Misalkan ada tiga buah bola yang berbeda warnanya, yaitu merah (*m*), biru (*b*), dan putih (*p*). Kita akan memasukkan ketiga buah bola itu ke dalam tiga buah kotak, masing-masing kotak 1 buah bola (Gambar 6.1). Berapa jumlah urutan berbeda yang mungkin dibuat dari penempatan bola ke dalam kotak-kotak tersebut?



Gambar 6.1 Ilustrasi untuk menjelaskan permutasi

Misalkan urutan itu kita simbolkan xyz . Urutan pertama (x) mungkin ditempati oleh salah satu dari 3 buah bola, urutan kedua (y) mungkin ditempati oleh salah satu dari 2 buah bola (karena 1 bola lagi sudah dipakai untuk x), dan urutan ketiga (z) ditempati oleh 1 buah bola yang tersisa, sehingga jumlah kemungkinan urutan berbeda dari penempatan bola ke dalam kotak adalah $(3)(2)(1) = 3! = 6$.

Perhatikan diagram berikut:



Dari diagram di atas terlihat bahwa kemungkinan urutan berbeda dari penempatan bola ke dalam tiga buah kotak ada enam buah, yaitu mbp , mpb , bmp , bpm , pmb , dan pbm . Semua urutan berbeda tersebut dinamakan **permutasi**.

DEFINISI 6.1.

Permutasi adalah jumlah urutan berbeda dari pengaturan objek-objek.

Permutasi merupakan bentuk khusus aplikasi aturan perkalian. Misalkan jumlah objek adalah n , maka urutan pertama dipilih dari n objek, urutan kedua dipilih dari $n - 1$ objek, urutan ketiga dipilih dari $n - 2$ objek, begitu seterusnya, dan urutan terakhir dipilih dari 1 objek yang tersisa. Menurut kaidah perkalian, permutasi dari n objek adalah

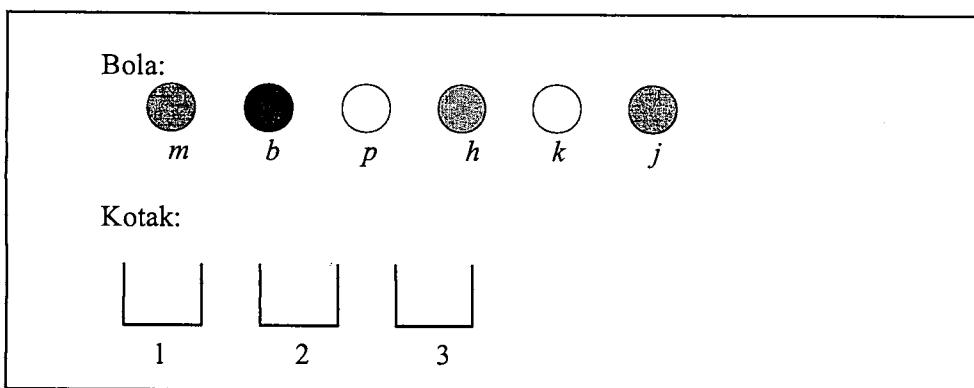
$$n(n - 1)(n - 2) \dots (2)(1) = n! \quad (6.1)$$

Sekarang misalkan ada enam buah bola yang berbeda warnanya, yaitu merah (m), biru (b), putih (p), hijau (h), kuning (k), dan jingga (j). Kita akan memasukkan keenam buah bola itu ke dalam tiga buah kotak, masing-masing kotak hanya boleh diisi 1 buah bola (Gambar 6.2). Berapa jumlah urutan berbeda yang mungkin dibuat dari penempatan bola ke dalam kotak-kotak tersebut?

Perhitungannya adalah sebagai berikut:

- kotak 1 dapat diisi oleh salah satu dari 6 bola (ada 6 pilihan);
- kotak 2 dapat diisi oleh salah satu dari 5 bola (ada 5 pilihan);
- kotak 3 dapat diisi oleh salah satu dari 4 bola (ada 4 pilihan).

Menurut kaidah perkalian, jumlah urutan berbeda dari penempatan bola = $(6)(5)(4) = 120$ buah¹.



Gambar 6.2 Ilustrasi untuk menjelaskan permutasi-. Ada 6 buah bola dan 3 buah kotak, tiap kotak akan diisi dengan 1 buah bola.

¹ Jika jumlah kotak lebih banyak daripada jumlah bola, maka cara penyelesaiannya tetap sama. Misalkan 3 buah bola berbeda akan dimasukkan ke dalam 6 buah kotak, tiap kotak paling banyak hanya boleh berisi 1 buah bola. Ada 6 pilihan kotak untuk memasukkan bola ke-1, 5 pilihan kotak untuk memasukkan bola ke-2, dan 4 pilihan kotak untuk memasukkan bola ke-3. Menurut kaidah perkalian, ada $(6)(5)(4) = 120$ cara untuk memasukkan 3 buah bola berbeda ke dalam 6 buah kotak.

Jika contoh tadi kita rampatkan sedemikian hingga ada n buah bola yang berbeda warnanya dan r buah kotak ($r \leq n$), maka

kotak ke- r dapat diisi oleh salah satu dari $(n - (r - 1))$ bola (ada $n - r + 1$ pilihan);

$$n(n-1)(n-2) \quad (n = (r-1))$$

Masalah yang mirip dengan memasukkan bola ke dalam kotak adalah menyusun r objek yang dipilih dari n objek berbeda. Misalnya, dari himpunan $A = \{a, b, c\}$, kita dapat membentuk 6 buah susunan 2-elemen yang dipilih dari himpunan A , yaitu ab, ac, ba, bc, ca , dan cb . Untuk menyusun r objek yang dipilih dari n objek berbeda, kita menganalogikan r buah posisi dengan r buah kotak, yang setiap posisi akan diisi dengan salah satu dari n objek berbeda (dianalogikan dengan n buah bola yang berbeda):

posisi ke-1 dapat diisi oleh salah satu dari n objek (ada n pilihan);
 posisi ke-2 dapat diisi oleh salah satu dari $(n - 1)$ objek (ada $n - 1$ pilihan);
 posisi ke-3 dapat diisi oleh salah satu dari $(n - 2)$ objek (ada $n - 2$ pilihan);
 . . .
 posisi ke- r dapat diisi oleh salah satu dari $(n - (r - 1))$ objek (ada $n - r + 1$ pilihan);

Menurut kaidah perkalian, ada sebanyak

$$n(n-1)(n-2)\dots(n-(r-1))$$

buah susunan berbeda dari penyusunan r objek yang dipilih dari n objek. Jumlah susunan berbeda dari pemilihan r objek yang diambil dari n objek disebut *permutasi- r* , dilambangkan dengan $P(n, r)$, yaitu

$$P(n, r) = n(n-1)(n-2)\dots(n-(r-1)) = \frac{n!}{(n-r)!} \quad (6.2)$$

Menurut persamaan (6.2) ini, jumlah cara memasukkan 6 buah bola yang berbeda warnanya ke dalam 3 buah kotak adalah $P(6, 3) = 6!/(6 - 3)! = 120$, dan jumlah kemungkinan urutan 2 dari 3 elemen himpunan $A = \{a, b, c\}$ adalah $P(3, 2) = 3!/(3 - 2)! = 6$.

DEFINISI 6.2. Permutasi r dari n objek adalah jumlah kemungkinan urutan r buah objek yang dipilih dari n buah objek, dengan $r \leq n$, yang dalam hal ini, pada setiap kemungkinan urutan tidak ada objek yang sama.

Perhatikanlah bahwa bila $r = n$, maka persamaan (6.2) menjadi sama dengan (6.1), yaitu

$$P(n, n) = \frac{n!}{(n - n)!} = \frac{n!}{0!} = \frac{n!}{1} = n!$$

Contoh 6.18

Buktikan dengan induksi matematik bahwa jumlah permutasi r elemen yang diambil dari himpunan n elemen, $P(n, r)$, dapat dihitung dengan rumus $n!/(n - r)!$.

Penyelesaian:

Kita dapat membuktikan pernyataan ini dengan melakukan induksi terhadap n . Untuk $n \geq 0$, misalkan $p(n)$ adalah pernyataan " $P(n, r) = n!/(n - r)!$ untuk $r = 0, 1, \dots$

Basis induksi.

Untuk $n = 0$, $P(0, 0)$ adalah jumlah cara memilih 0 buah elemen dari himpunan kosong = $0!/0! = 1$, yang jelas benar.

Langkah induksi.

Asumsikan bahwa $p(n)$ benar, yaitu mengasumsikan bahwa $P(n, r) = n!/(n - r)!$ untuk $r = 0, 1, \dots$. Kita tinggal membuktikan bahwa $p(n + 1) \equiv P(n + 1, r) = \frac{(n + 1)!}{(n + 1 - r)!}$ juga benar. Untuk menunjukkan hal ini, ada dua kasus yang harus dipertimbangkan.

Kasus 1:

jika $r = 0$, maka ada satu cara memilih memilih 0 buah elemen dari himpunan $(n + 1)$ elemen, dan di sini

$$P(n + 1, 0) = (n + 1)!/(n + 1 - 0)! = 1, \text{ yang jelas benar.}$$

Kasus 2:

jika $r > 0$. Di sini kita menghitung nilai $P(n + 1, r)$ dengan (i) menghitung jumlah cara memilih elemen pertama di dalam susunan yang diambil, dan (ii) kemudian menghitung jumlah cara mengambil $r - 1$ elemen dengan menggunakan hipotesis induksi. Ada $(n + 1)$ cara memilih untuk elemen pertama. Karena tinggal n buah elemen yang belum diambil untuk mengisi $r - 1$ posisi lainnya, maka ada $P(n, r - 1)$ cara melengkapi $r - 1$ posisi itu. Dengan aturan perkalian, maka

$$\begin{aligned} P(n + 1, r) &= (n + 1) P(n, r - 1) \\ &= \frac{(n + 1)n!}{(n - (r - 1))!} \\ &= \frac{(n + 1)!}{(n + 1 - r)!} \end{aligned}$$

■

Contoh 6.19

Berapa banyak "kata" yang terbentuk dari kata "BOSAN"?

Penyelesaian:

Anggap setiap huruf di dalam kata "BOSAN" sebagai bola yang berbeda warnanya, dan 5 buah kotak yang akan diisi dengan 1 bola pada setiap kotak.

Cara 1: $(5)(4)(3)(2)(1) = 120$ buah kata

Cara 2: $P(5, 5) = 5! = 120$ buah kata

Contoh 6.20

Berapa banyak cara mengurutkan nama 25 orang mahasiswa?

Penyelesaian:

Analogikan dengan mengisi 25 kotak dengan 25 bolahuruf berbeda, setiap kotak diisi dengan 1 bola. Jadi, jumlah cara pengurutan nama maahasiswa sama dengan jumlah susunan 25 bola ke dalam 25 kotak, yaitu $P(25, 25) = 25!$

Contoh 6.21

Tiga buah ujian dilakukan dalam suatu periode enam hari (Senin sampai Sabtu). Berapa banyak pengaturan jadwal yang dapat dilakukan sehingga tidak ada dua ujian atau lebih yang dilakukan pada hari yang sama.

Penyelesaian:

Cara 1 (dengan kaidah perkalian): sama seperti menempatkan 3 bola (ujian) berbeda ke dalam enam kotak (hari).

ujian pertama dapat ditempatkan pada salah satu dari enam hari;

ujian kedua dapat ditempatkan pada salah satu dari lima hari;

ujian ketiga dapat ditempatkan pada salah satu dari empat hari;

Jumlah pengaturan jadwal ujian = $(6)(5)(4) = 120$

Cara 2 (dengan rumus permutasi): $P(6, 3) = 6! / (6 - 3)! = 120$

Contoh 6.22

Sebuah bioskop mempunyai jajaran kursi yang disusun per baris. Tiap baris terdiri dari 6 tempat kursi. Jika dua orang akan duduk, berapa banyak pengaturan tempat duduk yang mungkin pada suatu baris?

Penyelesaian:

Orang pertama mempunyai 6 pilihan kursi, dan orang kedua mempunyai 5 pilihan kursi. Jadi, jumlah pengaturan tempat duduk = $(6)(5) = 30$ atau $P(6, 2) = 6! / 4! = (6)(5) = 30$ cara.

Contoh 6.23

Berapa banyak *string* yang dapat dibentuk yang terdiri dari 4 huruf berbeda dan diikuti dengan 3 angka yang berbeda pula?

Penyelesaian:

Ada $P(26, 4)$ cara mengisi posisi 4 huruf dan $P(10, 3)$ cara untuk mengisi posisi 3 buah angka. Karena *string* disusun oleh 4 huruf dan 3 angka, maka jumlah *string* yang dapat dibuat adalah $P(26, 4) \times P(10, 3) = 258.336.000$ ■

Contoh 6.24

Berapa banyak cara penyusunan 15-puzzle seperti contoh di bawah ini?

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	

Penyelesaian:

Cara 1: $P(16,15) \rightarrow$ salah, karena sel kosong tidak dianggap sebagai sebuah objek berbeda dari yang lain.

$$P(16,16) = 16! \rightarrow \text{betul}$$

Cara 2: Posisi ubin pertama dapat diisi dengan salah satu dari 15 angka atau 1 ubin kosong (16 cara)

Posisi ubin kedua dapat diisi dengan salah satu dari 14 angka atau 1 ubin kosong (15 cara)

Posisi ubin ketiga dapat diisi dengan salah satu dari 13 angka atau 1 ubin kosong (14 cara)

...

Posisi ubin ke-16 dapat disi dengan yang tersisa (1 cara)

$$\text{Jumlah cara penyusunan} = (16)(15)(14)(13)\dots(2)(1) = 16! \text{ cara} \quad \blacksquare$$

Dari contoh-contoh di atas dapat dilihat bahwa permutasi merupakan suatu bagian dari kaidah perkalian, yaitu untuk kasus bahwa tidak ada elemen yang berulang. Jika pada persoalan diizinkan munculnya sebuah elemen lebih dari satu kali, maka persoalan tersebut tidak dapat diselesaikan dengan cara permutasi, tetapi masih dapat diselesaikan dengan memanfaatkan kaidah perkalian. Contoh-contoh berikut memperlihatkan penggunaan permutasi dan jika tidak mungkin, menggunakan kaidah perkalian.

Contoh 6.25

Berapakah jumlah kemungkinan membentuk 3 angka dari 5 angka berikut: 1, 2, 3, 4, 5, jika:

- (a) tidak boleh ada pengulangan angka, dan
- (b) boleh ada pengulangan angka.

Penyelesaian:

- (a) Ada 3 posisi yang akan dipilih dari 5 angka. Posisi pertama dapat diisi oleh salah satu dari 5 angka, posisi kedua oleh salah satu dari 4 angka, dan posisi ketiga oleh salah satu dari 3 angka. Sehingga, jumlah urutan 3-angka yang dapat dibentuk adalah $(5)(4)(3) = 120$ buah, atau dengan rumus permutasi $P(5, 3) = 5!(5 - 3)! = 120$
- (b) Persoalan ini tidak dapat diselesaikan dengan rumus permutasi, tetapi masih dapat diselesaikan dengan kaidah perkalian. Posisi pertama dapat diisi dengan salah satu dari 5 angka (5 cara), posisi kedua juga 5 cara (karena boleh sama dengan angka pada posisi pertama), posisi ketiga juga 5 cara. Sehingga jumlah urutan 3-angka yang dapat dibentuk adalah $(5)(5)(5) = 5^3 = 125$. ■

Contoh 6.26

Berapakah banyak *string* yang dibentuk dari permutasi huruf-huruf pada kata “sarung” sedemikian sehingga huruf-huruf vokal terletak pada posisi saling bersebelahan?

Penyelesaian:

Huruf vokal di dalam kata “sarung” adalah *u* dan *a*. Yang ditanyakan adalah jumlah *string* yang mengandung *au* atau *ua*. Karena *au* atau *ua* harus muncul dalam satu blok, maka kita harus menghitung jumlah permutasi blok *au* atau *ua* dengan huruf-huruf *s*, *r*, *n*, dan *g*. Untuk *au*, *s*, *r*, *n*, dan *g*, jumlah kata yang dapat dibentuk adalah $P(5, 5) = 5!$, dan Untuk *ua*, *s*, *r*, *n*, dan *g*, jumlah kata yang dapat dibentuk adalah $P(5, 5) = 5$. Jumlah kata seluruhnya adalah $5! + 5! = 240$. ■

Permutasi Melingkar

Misalkan ada 10 orang yang duduk pada satu barisan kursi yang terdiri dari 10 kursi. Menurut rumus permutasi, ada sebanyak $P(10, 10) = 10!$ cara pengaturan tempat duduk bagi 10 orang tersebut. Sekarang, misalkan mereka disuruh duduk mengelilingi meja melingkar. Berapa banyak cara pengaturan tempat duduk bagi mereka tersebut? Satu orang dapat duduk pada tempat duduk mana saja. Sembilan orang lainnya dapat duduk dalam $9 \times 8 \times 7 \times 6 \times 5 \times 4 \times 3 \times 2 \times 1 = 9!$ cara. Meskipun orang pertama dapat memilih tempat duduk mana saja, namun susunan tempat duduk yang dihasilkan oleh 9 orang lainnya tetap sama. Ini dinamakan **permutasi melingkar** yang didefinisikan sebagai berikut:

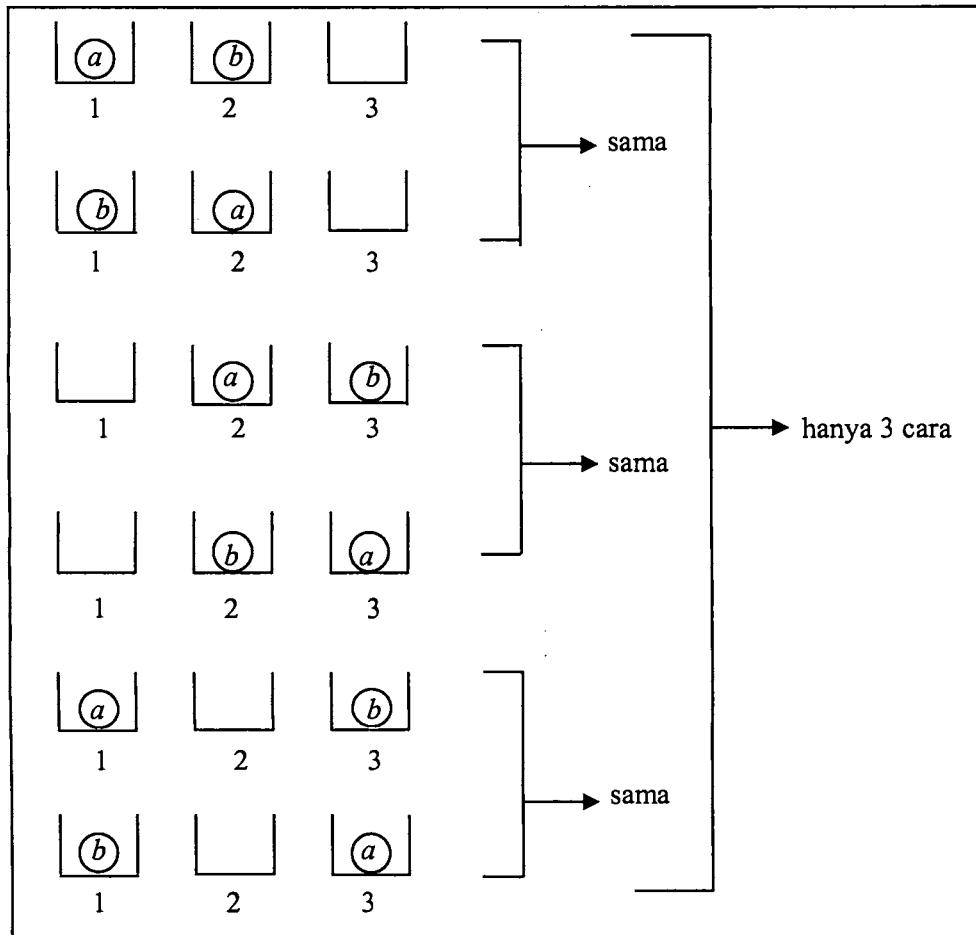
DEFINISI 6.3. Permutasi melingkar dari n objek adalah penyusunan objek-objek yang mengelilingi sebuah lingkaran (atau kurva tertutup sederhana). Jumlah susunan objek yang mengelilingi lingkaran adalah $(n - 1)!$.

Pembuktian permutasi melingkar cukup sederhana: objek pertama dapat ditempatkan di mana saja pada lingkaran dengan 1 cara. Sisa $n - 1$ objek lainnya dapat diatur searah jarum jam (misalnya) dengan $P(n - 1, n - 1) = (n - 1)!$ cara.

6.6 Kombinasi

Bentuk khusus dari permutasi adalah kombinasi. Jika pada permutasi urutan kemunculan diperhitungkan, maka pada kombinasi, urutan kemunculan diabaikan. Urutan acb , bca , dan acb dianggap sama dan dihitung sekali.

Misalkan ada 2 buah bola yang warnanya sama, misalnya merah semua (untuk membedakan masing-masing bola, kita namakan bola a dan bola b), dan 3 buah kotak. Kita ingin memasukkan bola ke dalam kotak, setiap kotak hanya boleh berisi paling banyak 1 bola. Gambar 6.3 mengilustrasikan penempatan bola ke dalam kotak. Hasil akhir penempatan bola a ke kotak 1 dan bola b ke kotak 2 sama saja dengan hasil akhir penempatan bola b ke kotak 1 dan bola a ke kotak 2. Susunan yang diperoleh hanya dihitung sekali (1 cara). Hal yang sama juga dihitung pada waktu menempatkan bola a dan b ke kotak 2 dan 3, dan menempatkan bola a dan b ke kotak 1 dan 3.



Gambar 6.3 Ilustrasi untuk menjelaskan kombinasi- r

$$\text{Jumlah cara memasukkan bola ke dalam kotak} = \frac{P(3,2)}{2} = \frac{P(3,2)}{2!} = \frac{\frac{3!}{1!}}{2!} = \frac{(3)(2)}{2} = 3.$$

Sekarang bila jumlah bola 3 dan jumlah kotak 10, maka jumlah cara memasukkan bola ke dalam kotak adalah $\frac{P(10,3)}{3!} = \frac{\frac{10!}{7!}}{3!} = \frac{(10)(9)(8)}{3!}$ karena ada 3! cara memasukkan bola yang warnanya merah semua.

Secara umum, jumlah cara memasukkan r buah bola yang berwarna sama ke dalam n buah kotak adalah

$$\frac{n(n-1)(n-2)\dots(n-(r-1))}{r!} = \frac{n!}{r!(n-r)!}$$

Rumus $\frac{n!}{r!(n-r)!}$ disebut rumus **kombinasi- r** , dan dilambangkan dengan $C(n, r)$ atau $\binom{n}{r}$.

Jadi,

$$C(n, r) = \frac{n!}{r!(n-r)!} \quad (6.3)$$

$C(n, r)$ sering dibaca "n diambil r ", artinya r objek diambil dari n buah objek.

DEFINISI 6.4. Kombinasi r elemen dari n elemen adalah jumlah pemilihan yang tidak terurut r elemen yang diambil dari n buah elemen.

Persamaan (6.3) dapat juga dibuktikan dengan cara membentuk permutasi- r dari n elemen. Mula-mula hitung kombinasi- r , yaitu $C(n, r)$, kemudian urutkan elemen-elemen di dalam setiap kombinasi- r . Pengurutan ini dapat dilakukan dengan $P(r, r)$ cara. Dengan demikian, permutasi- r dari n elemen adalah

$$P(n, r) = C(n, r) P(r, r)$$

Dari persamaan di atas kita peroleh

$$C(n, r) = \frac{P(n, r)}{P(r, r)} = \frac{n!/(n-r)!}{r!(r-r)!} = \frac{n!}{r!(n-r)!}$$

Interpretasi Kombinasi

1. Persoalan kombinasi, $C(n, r)$, sama dengan menghitung banyaknya himpunan bagian yang terdiri dari r elemen yang dapat dibentuk dari himpunan dengan n elemen. Dua atau lebih himpunan bagian dengan elemen-elemen yang sama dianggap sebagai himpunan yang sama, meskipun urutan elemen-elemennya berbeda.

Misalkan $A = \{1, 2, 3\}$

Jumlah Himpunan bagian dengan 2 elemen yang dapat dibentuk dari himpunan A ada 3 buah, yaitu:

$$\begin{aligned}\{1, 2\} &= \{2, 1\} \\ \{1, 3\} &= \{3, 1\} \\ \{2, 3\} &= \{3, 2\}\end{aligned}\quad \begin{array}{l}> \\ > \\ >\end{array}\quad 3 \text{ buah}$$

$$\text{atau } \binom{3}{2} = \frac{3!}{(3-2)!2!} = \frac{3!}{1!2!} = 3 \text{ buah}$$

2. Persoalan kombinasi, $C(n, r)$, dapat dipandang sebagai cara memilih r buah elemen dari n buah elemen yang ada, tetapi urutan elemen di dalam susunan hasil pemilihan tidak penting.

Sebagai contoh, misalkan sebuah klub memiliki 25 orang anggota. Kita akan memilih lima orang sebagai panitia. Panitia atau komite adalah kelompok yang tidak terurut, artinya setiap anggota di dalam panitia kedudukannya sama. Misalnya jika ada lima orang yang dipilih, A, B, C, D , dan E , maka urutan penempatan masing-masingnya di dalam panitia tidak penting ($ABCDE$ sama saja dengan $BACED$, $ADCEB$, dan seterusnya). Banyaknya cara memilih anggota panitia yang terdiri dari 5 orang anggota adalah $C(25,5) = 53130$ cara.

Contoh 6.27 sampai 6.35 berikut ini dapat menggambarkan persoalan-persoalan yang diselesaikan dengan rumus kombinasi.

Contoh 6.27

Ada berapa cara kita dapat memilih 3 dari 4 elemen himpunan $A = \{a, b, c, d\}$?

Penyelesaian:

Ini adalah persoalan kombinasi karena urutan kemunculan ketiga elemen tersebut tidak penting.

Perhatikan tabel berikut:

Himpunan bagian A dengan 3 elemen	Permutasi setiap himpunan bagian
$\{a, b, c\}$	$abc, acb, bca, bac, cab, cba$
$\{a, b, d\}$	$abd, adb, bda, bad, dab, dba$
$\{a, c, d\}$	$acd, adc, cda, cad, dac, dca$
$\{b, c, d\}$	$bcd, bdc, cdb, cbd, dbc, dc b$

Untuk setiap 3 elemen ada $3! = 6$ urutan yang berbeda (permutasi).

Jadi, jumlah cara memilih 3 dari 4 elemen himpunan adalah $C(4, 3) = \frac{4!}{3!(4-3)!} = 4$, yaitu himpunan $\{a, b, c\}$, $\{a, b, d\}$, $\{a, c, d\}$, dan $\{b, c, d\}$. ■

Contoh 6.28

Berapa banyak cara menyusun menu nasi goreng tiga kali seminggu untuk sarapan pagi?

Penyelesaian:

Bayangkan tiga kali menu nasi goreng sebagai tiga buah bola dan tujuh hari dalam seminggu sebagai 7 buah kotak. Persoalan ini sama dengan menempatkan 3 buah bola ke dalam 7 buah kotak. Banyaknya pengaturan jadwal menu nasi goreng adalah $C(7, 3) = \frac{7!}{3!4!} = 35$ cara. ■

Contoh 6.29

String biner yang panjangnya 32 bit disusun oleh angka 1 atau 0. Berapa banyak *string* biner yang tepat berisi 7 buah bit 1?

Penyelesaian:

Analogikan 7 bit 1 sebagai 7 buah bola, dan 32 posisi bit sebagai 32 buah kotak. Persoalan ini sama dengan memasukkan 7 bola ke dalam 32 kotak, sisanya kosong (0). Banyak *string* biner yang terbentuk adalah $C(32, 7)$. ■

Contoh 6.30

Sebuah koin yang mempunyai sisi A dan sisi B dilempar ke atas sebanyak empat kali. Berapakah jumlah kemungkinan munculnya sisi A sebanyak tiga kali?

Penyelesaian:

Ini adalah persoalan dari kombinasi karena kita tidak mementingkan kapan sisi A tersebut muncul. Jadi, jumlah kemungkinan munculnya sisi A sebanyak tiga kali adalah $C(4, 3) = 4$.

Contoh 6.31

Sebuah karakter dalam sistem ASCII berukuran 1 *byte* atau 8 bit (1 atau 0).

- (a) Berapa banyak pola bit yang terbentuk? (atau, berapa banyak karakter yang dapat direpresentasikan?)
- (b) Berapa banyak pola bit yang mempunyai 3 bit 1?
- (c) Berapa banyak pola bit yang mempunyai bit 1 sejumlah genap?

Penyelesaian:

- (a) Posisi bit dalam 1 byte: _____
- | | | | | | | | |
|---|---|---|---|---|---|---|---|
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|

Posisi 0 dapat diisi dengan 2 cara (1 atau 0)

Posisi 1 dapat diisi dengan 2 cara (1 atau 0)

Posisi 2 dapat diisi dengan 2 cara (1 atau 0)

...

Posisi 7 dapat diisi dengan 2 cara (1 atau 0)

Semua posisi harus diisi, jadi jumlah pola bit yang terbentuk
 $= (2)(2)(2)(2)(2)(2)(2) = 2^8$

(b) $C(8, 3) = 56$

- (c) Banyaknya pola bit yang mempunyai 0 buah bit 1 = $C(8, 0)$
 Banyaknya pola bit yang mempunyai 2 buah bit 1 = $C(8, 2)$
 Banyaknya pola bit yang mempunyai 4 buah bit 1 = $C(8, 4)$
 Banyaknya pola bit yang mempunyai 6 buah bit 1 = $C(8, 6)$
 Banyaknya pola bit yang mempunyai 8 buah bit 1 = $C(8, 8)$

Jadi, banyaknya pola bit yang mempunyai bit 1 sejumlah genap
 $= C(8, 0) + C(8, 2) + C(8, 4) + C(8, 6) + C(8, 8)$ ■

Contoh 6.32

Di antara 10 orang mahasiswa Teknik Informatika Angkatan 1998, berapa banyak cara membentuk sebuah perwakilan beranggotakan 5 orang sedemikian sehingga:

- (a) mahasiswa bernama *A* selalu termasuk di dalamnya;
- (b) mahasiswa bernama *A* tidak termasuk di dalamnya;
- (c) mahasiswa bernama *A* selalu termasuk di dalamnya, tetapi *B* tidak;
- (d) mahasiswa bernama *B* selalu termasuk di dalamnya, tetapi *A* tidak;
- (e) mahasiswa bernama *A* dan *B* termasuk di dalamnya;
- (f) setidaknya salah satu dari mahasiswa yang bernama *A* atau *B* termasuk di dalamnya.

Penyelesaian:

- (a) Masukkan *A* ke dalam perwakilan sehingga sekarang tersisa 9 orang mahasiswa. Dari 9 orang ini, pilih 4 orang lagi sebagai anggota perwakilan. Ini dapat dilakukan dengan $C(9, 4)$ cara. Jadi, ada $C(9, 4) = 126$ cara untuk membentuk perwakilan yang beranggotakan 5 orang sedemikian sehingga *A* selalu termasuk di dalamnya.
- (b) Keluarkan *A* dari kelompok mahasiswa, sehingga sekarang tersisa 9 orang mahasiswa. Dari 9 orang ini, pilih 5 orang sebagai anggota perwakilan. Ini dapat dilakukan

- dengan $C(9, 5)$ cara. Jadi, ada $C(9, 5) = 126$ cara untuk membentuk perwakilan yang beranggotakan 5 orang sedemikian sehingga A tidak termasuk di dalamnya.
- (c) Keluarkan B dari kelompok mahasiswa (tersisa 9 orang). Masukkan A ke dalam perwakilan (tersisa 8 orang). Dari 8 orang ini, pilih 4 orang lagi sebagai anggota perwakilan. Ini dapat dilakukan dengan $C(8, 4)$ cara. Jadi, ada $C(8, 4) = 70$ cara untuk membentuk perwakilan yang beranggotakan 5 orang sedemikian sehingga A termasuk di dalamnya, tetapi B tidak.
- (d) Dengan cara yang sama seperti jawaban (c), terdapat $C(8, 4) = 70$ cara untuk membentuk perwakilan yang beranggotakan 5 orang sedemikian sehingga B termasuk di dalamnya, tetapi A tidak.

- (e) Masukkan A dan B ke dalam perwakilan sehingga sekarang tersisa 8 orang mahasiswa. Dari 8 orang ini, pilih 3 orang lagi sebagai anggota perwakilan. Ini dapat dilakukan dengan $C(8, 3)$ cara. Jadi, ada $C(8, 3) = 56$ cara untuk membentuk perwakilan yang beranggotakan 5 orang sedemikian sehingga A dan B selalu termasuk di dalamnya.
- (f) Jumlah cara membentuk perwakilan sedemikian sehingga setidaknya salah satu dari A atau B termasuk di dalamnya

$$\begin{aligned}
 &= \text{jumlah cara membentuk perwakilan sehingga } A \text{ termasuk di dalamnya, } B \text{ tidak} + \\
 &\quad \text{jumlah cara membentuk perwakilan sehingga } B \text{ termasuk di dalamnya, } A \text{ tidak} + \\
 &\quad \text{jumlah cara membentuk perwakilan sehingga } A \text{ dan } B \text{ termasuk di dalamnya} \\
 &= 70 + 70 + 56 = 196
 \end{aligned}$$

Soal (f) ini dapat juga diselesaikan dengan prinsip inklusi-eksklusi. Misalkan X adalah jumlah cara membentuk perwakilan yang menyertakan A , Y adalah jumlah cara membentuk perwakilan yang menyertakan B , dan $X \cap Y$ adalah jumlah cara membentuk perwakilan yang menyertakan A dan B , maka

$$|X| = C(9, 4) = 126; \quad |Y| = C(9, 4) = 126; \quad |X \cap Y| = C(8, 3) = 56;$$

sehingga

$$|X \cup Y| = |X| + |Y| - |X \cap Y| = 126 + 126 - 56 = 196 \quad \blacksquare$$

Contoh 6.33

Sebuah klub beranggotakan 8 pria dan 10 wanita. Berapa banyak cara memilih panitia yang terdiri dari 6 orang dengan jumlah wanita lebih banyak daripada pria?

Penyelesaian:

Panitia: 6 orang, jumlah wanita lebih banyak daripada jumlah pria

Panitia terdiri dari 5 wanita, 1 pria \rightarrow dapat dibentuk dengan $C(10,5) \times C(8,1)$

Panitia terdiri dari 4 wanita, 2 pria \rightarrow dapat dibentuk dengan $C(10,4) \times C(8,2)$

Panitia terdiri dari 6 wanita, 0 pria \rightarrow dapat dibentuk dengan $C(10,6) \times C(8,0)$

Jumlah cara pembentukan panitia seluruhnya $= C(10,5) \times C(8,1) + C(10,4) \times C(8,2) + C(10,6) \times C(8,0)$. \blacksquare

Contoh 6.34

Tiga buah apartemen A , B , dan C disewakan untuk mahasiswa. Tiap unit apartemen dapat menampung 3 atau 4 orang mahasiswa. Berapa jumlah cara menyewakan apartemen kepada 10 orang mahasiswa?

Penyelesaian:

- andaikan apartemen A , B , C ditempati masing-masing oleh 4, 3, dan 3 orang mahasiswa. Jumlah cara menyewakan = $C(10,4) \times C(6,3) \times C(3,3)$
- andaikan apartemen A , B , dan C ditempati masing-masing oleh 3, 4, dan 3 orang mahasiswa. Jumlah cara menyewakan = $C(10,3) \times C(7,4) \times C(3,3)$
- andaikan apartemen A , B , dan C ditempati masing-masing oleh 3, 3, dan 4 orang mahasiswa. Jumlah cara menyewakan = $C(10,3) \times C(7,3) \times C(4,4)$

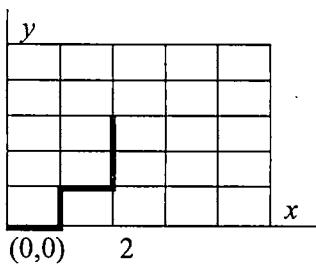
$$\begin{aligned}\text{Total seluruh cara menyewakan} &= C(10,4)C(6,3) + C(10,3)C(7,4) + C(10,3)C(7,3) \\ &= 3C(10,4)C(6,3).\end{aligned}$$

Contoh 6.35

Pandang sebuah bidang kartesian dengan koordinat positif (Gambar 6.4). Tiap titik koordinatnya adalah (x, y) . Seekor semut bergerak dari $(0,0)$ ke titik $A(m, n)$, m dan $n > 0$. Lintasan yang dilalui semut memiliki ketentuan sebagai berikut:

- dimulai dari titik asal $(0,0)$,
- melangkah selalu sejajar sumbu- X atau sumbu- Y positif,
- boleh membelok hanya pada titik-titik *grid*,
- berhenti di A .

Berapakah jumlah langkah (panjang lintasan) dan banyaknya lintasan dari $(0,0)$ ke $A(m, n)$?



Gambar 6.4 Persoalan menentukan banyaknya lintasan dari $(0, 0)$ ke titik (x, y)

Penyelesaian:

Panjang lintasan = $m + n$ langkah (m horizontal dan n vertikal)

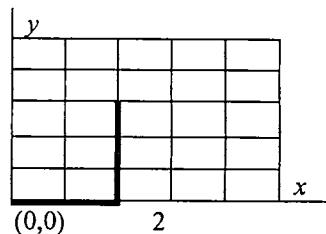
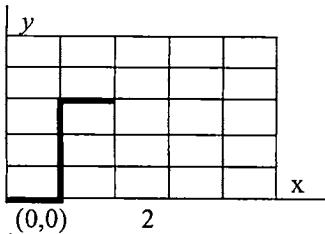
Contohnya, pada gambar 6.4 di atas, panjang lintasan dari $(0,0)$ ke $A(2,3) = 2 + 3 = 5$.

Banyaknya lintasan = $C(m + n, m) = C(m + n, n)$

$$= C(2 + 3, 2) = C(5, 2) = \frac{5!}{(2!)(3!)} = 10.$$

$$= C(2+3,3) = C(5,3) = \frac{5!}{(3!)(2!)} = 10$$

Contoh beberapa lintasan lain dari $(0,0)$ ke $A(2,3)$ diperlihatkan di bawah ini:



6.7 Permutasi dan Kombinasi Bentuk Umum

Kita mempunyai n buah bola yang tidak seluruhnya berbeda warna (jadi, ada beberapa bola yang warnanya sama - *indistinguishable*). Misalkan dari n buah bola itu terdapat

- n_1 bola diantaranya berwarna 1,
- n_2 bola diantaranya berwarna 2,
- \vdots
- n_k bola diantaranya berwarna k ,

dan $n_1 + n_2 + \dots + n_k = n$. Kita diminta memasukkan n buah bola ini ke dalam n buah kotak, masing-masing kotak berisi paling banyak 1 buah bola. Berapa jumlah cara pengaturan n buah bola ke dalam kotak-kotak tersebut?

Perhitungannya adalah sebagai berikut:

Jika n buah bola itu kita anggap berbeda semuanya, maka jumlah cara pengaturan n buah bola ke dalam n buah kotak adalah $P(n, n) = n!$.

Tetapi, karena tidak seluruh bola berbeda, maka dari pengaturan n buah bola itu, ada $n_1!$ cara memasukkan bola berwarna 1 (jika masing-masingnya dibedakan); ada $n_2!$ cara memasukkan bola berwarna 2 (jika masing-masingnya dibedakan); \vdots ; ada $n_k!$ cara memasukkan bola berwarna k (jika masing-masingnya dibedakan).

Dengan demikian, permutasi n buah bola yang mana n_1 diantaranya berwarna 1, n_2 bola berwarna 2, ..., n_k bola berwarna k adalah:

$$P(n; n_1, n_2, \dots, n_k) = \frac{P(n, n)}{n_1! n_2! \dots n_k!} = \frac{n!}{n_1! n_2! \dots n_k!} \quad (6.4)$$

Persamaan (6.4) ini diterapkan untuk menghitung pengaturan (atau pengurutan) n buah objek dari himpunan ganda S (himpunan S terdiri dari n buah obyek yang *tidak perlu* semuanya berbeda). Persamaan tersebut dinamakan **permutasi bentuk umum** (*generalized permutation*) terhadap S .

Persamaan (6.4) dapat pula kita peroleh sebagai berikut:

Mula-mula kita menempatkan bola-bola yang berwarna 1 ke dalam n buah kotak. Ada $C(n, n_1)$ cara untuk menempatkan n_1 buah bola yang berwarna 1. Setelah bola berwarna 1 dimasukkan, sekarang terdapat $n - n_1$ kotak yang belum diisi. Kita masukkan bola-bola yang berwarna 2. Ada $C(n - n_1, n_2)$ cara untuk menempatkan n_2 buah bola berwarna 2. Setelah bola berwarna 2 dimasukkan, sekarang terdapat $n - n_1 - n_2$ kotak yang belum diisi. Kita masukkan bola-bola yang berwarna 3. Ada $C(n - n_1 - n_2, n_3)$ cara untuk menempatkan n_3 buah bola berwarna 3. Demikian seterusnya, sehingga akhirnya terdapat $C(n - n_1 - n_2 - \dots - n_{k-1}, n_k)$ cara untuk menempatkan n_k buah bola berwarna k .

Jumlah cara pengaturan seluruh bola kedalam kotak adalah:

$$\begin{aligned}
 C(n; n_1, n_2, \dots, n_k) &= C(n, n_1) C(n - n_1, n_2) C(n - n_1 - n_2, n_3) \dots \\
 &\quad C(n - n_1 - n_2 - \dots - n_{k-1}, n_k) \\
 &= \frac{n!}{n_1!(n-n_1)!} \frac{(n-n_1)!}{n_2!(n-n_1-n_2)!} \frac{(n-n_1-n_2)!}{n_3!(n-n_1-n_2-n_3)!} \\
 &\quad \dots \frac{(n-n_1-n_2-\dots-n_{k-1})!}{n_k!(n-n_1-n_2-\dots-n_{k-1}-n_k)!} \\
 &= \frac{n!}{n_1!n_2!n_3!\dots n_k!} \tag{6.5}
 \end{aligned}$$

Persamaan (6.5) ini dinamakan **kombinasi bentuk umum** (*generalized combination*). Kita dapat melihat bahwa tidak ada perbedaan antara permutasi bentuk umum dengan kombinasi bentuk umum. Keduanya dapat dihitung dengan rumus yang sama.

Jadi, apabila S adalah himpunan ganda dengan n buah objek yang di dalamnya terdiri atas k jenis obyek berbeda, dan tiap obyek memiliki multiplisitas n_1, n_2, \dots, n_k (jumlah objek seluruhnya $n_1 + n_2 + \dots + n_k = n$), maka jumlah cara menyusun seluruh objek adalah:

$$P(n; n_1, n_2, \dots, n_k) = C(n; n_1, n_2, \dots, n_k) = \frac{n!}{n_1!n_2!\dots n_k!}$$

Contoh 6.36

Contoh permasalahan yang sering dikemukakan untuk mengilustrasikan permutasi bentuk umum adalah masalah berikut: Berapa banyak *string* yang dapat dibentuk dengan menggunakan huruf-huruf dari kata *MISSISSIPPI*?

Penyelesaian:

$$S = \{M, I, S, S, I, S, S, I, P, P, I\}$$

huruf $M = 1$ buah (n_1)

huruf $I = 4$ buah (n_2)

huruf $S = 4$ buah (n_3)

huruf $P = 2$ buah (n_4)

$n = 1 + 4 + 4 + 2 = 11$ buah = jumlah elemen himpunan S

Ada dua cara yang dapat digunakan untuk menyelesaikan persoalan ini, keduanya memberikan hasil yang sama:

Cara 1: Jumlah *string* = $P(11; 1, 4, 4, 2) = \frac{11!}{(1!)(4!)(4!)(2!)} = 34650$ buah.

Cara 2: Jumlah *string* = $C(11, 1)C(10, 4)C(6, 4)C(2, 2)$

$$\begin{aligned} &= \frac{11!}{(1!)(10!)} \cdot \frac{10!}{(4!)(6!)} \cdot \frac{6!}{(4!)(2!)} \cdot \frac{2!}{(2!)(0!)} \\ &= \frac{11!}{(1!)(4!)(4!)(2!)} \\ &= 34650 \text{ buah} \end{aligned}$$

■

Contoh 6.37

12 lembar karton akan diwarnai sehingga 3 diantaranya berwarna hijau, 2 berwarna merah, 2 berwarna kuning, dan sisanya berwarna biru. Berapa jumlah cara pengecatan?

Penyelesaian:

Diketahui $n_1 = 3, n_2 = 2, n_3 = 2, n_4 = 5$, dan $n_1 + n_2 + n_3 + n_4 = 3 + 2 + 2 + 5 = 12$

Jumlah cara pengecatan = $P(12; 3, 2, 2, 5) =$

$$\frac{P(12,12)}{(3!)(2!)(2!)(5!)} = \frac{12!}{(3!)(2!)(2!)(5!)} = 166320 \text{ cara}$$

■

Contoh 6.38

12 buah lampu berwarna (4 merah, 3 putih, dan 5 biru) dipasang pada 18 buah soket dalam sebuah baris (sisanya 6 buah soket dibiarkan kosong). Berapa jumlah cara pengaturan lampu?

Penyelesaian:

Diketahui, $n = 18; n_1 = 4, n_2 = 3, n_3 = 5$, dan $n_4 = 6$ (*socket* kosong)

$$\text{Jumlah cara pengaturan lampu} = P(18; 4, 3, 5, 6) = \frac{18!}{(4!)(3!)(5!)(6!)} \text{ cara}$$

■

Contoh 6.39

Berapa banyak cara membagikan delapan buah buku berbeda kepada 3 orang mahasiswa, bila Billy mendapat empat buah buku, dan Andi serta Toni masing-masing memperoleh 2 buah buku.

Penyelesaian:

Diketahui $n = 8$, $n_1 = 4$, $n_2 = 2$, $n_3 = 2$, dan $n_1 + n_2 + n_3 = 4 + 2 + 2 = 8$

$$\text{Jadi, jumlah cara membagi seluruh buku} = \frac{8!}{(4!)(2!)(2!)} = 420 \text{ cara}$$

■

6.8 Kombinasi dengan Pengulangan

Tinjau kembali persoalan memasukkan bola ke dalam kotak. Misalkan terdapat r buah bola yang semua warnanya sama dan n buah kotak.

- Jika masing-masing kotak hanya boleh diisi paling banyak satu buah bola, maka jumlah cara memasukkan bola ke dalam kotak adalah $C(n, r)$.
- jika masing-masing kotak boleh lebih dari satu buah bola (tidak ada pembatasan jumlah bola), maka jumlah cara memasukkan bola ke dalam kotak adalah $C(n + r - 1, r)$.

$C(n + r - 1, r)$ adalah jumlah kombinasi yang membolehkan adanya pengulangan elemen, yaitu dari n buah objek kita akan mengambil r buah objek, dengan pengulangan diperbolehkan. Pembuktian rumus kombinasi dengan pengulangan tidak disertakan di sini.

Perhatikan bahwa $C(n + r - 1, r) = C(n + r - 1, n - 1)$.

Contoh 6.40

Pada persamaan $x_1 + x_2 + x_3 + x_4 = 12$, x_i adalah bilangan bulat ≥ 0 . Berapa jumlah kemungkinan solusinya?

Penyelesaian:

Analogikan 12 buah bola akan dimasukkan ke dalam 4 buah kotak (dalam hal ini, $n = 4$ dan $r = 12$). Bagilah keduabelas bola itu ke dalam tiap kotak. Sebuah kotak mungkin diisi 1 bola, atau 2 bola, atau ..., atau +12 bola, atau tidak diisi sama sekali, yang penting jumlah seluruh bola di dalam seluruh kotak tetap 12 buah. Misalnya,

Kotak 1 diisi 3 buah bola ($x_1 = 3$)
 Kotak 2 diisi 5 buah bola ($x_2 = 5$)
 Kotak 3 diisi 2 buah bola ($x_3 = 2$)
 Kotak 4 diisi 2 buah bola ($x_4 = 2$)
 $x_1 + x_2 + x_3 + x_4 = 3 + 5 + 2 + 2 = 12$

Banyak sekali jumlah susunan yang mungkin, namun seluruhnya ada $C(4 + 12 - 1, 12) = C(15, 12) = 455$ buah kemungkinan solusi. ■

Contoh 6.41

Tinjau kembali Contoh 6.41. Berapa banyak kemungkinan solusi persamaan tersebut bila disyaratkan $x_1 > 0$, $x_2 > 1$, $x_3 > 2$, dan $x_4 \geq 0$?

Penyelesaian:

Kita akan membagian 12 buah bola ke dalam kotak. Kotak pertama harus berisi paling sedikit 1 bola, kotak kedua paling sedikit 2 bola, kotak ketiga paling sedikit 3 bola, dan kotak keempat boleh kosong. Mula-mula masukkan 1 bola, 2 bola, dan 3 bola ke masing-masing kotak 1, 2, dan 3, sehingga tersisa 6 bola. Kemudian, masukkan 6 buah bola yang tersisa ini ke dalam empat kotak itu. Ini dapat dilakukan dalam sejumlah $C(6 + 4 - 1, 6) = C(9, 6) = 84$ cara. Jadi, ada 84 kemungkinan solusi persamaan. ■

Contoh 6.42

20 buah apel dan 15 buah jeruk dibagikan kepada 5 orang anak, tiap anak boleh mendapat lebih dari 1 buah apel atau jeruk, atau tidak sama sekali. Berapa jumlah cara pembagian yang dapat dilakukan?

Penyelesaian:

Pada persoalan ini, $n = 5$, $r_1 = 20$ (apel) dan $r_2 = 15$ (jeruk). Membagikan 20 buah apel kepada 5 orang anak dapat dilakukan dengan $C(5 + 20 - 1, 20)$ cara, dan membagikan 15 buah jeruk kepada 5 orang anak dapat dilakukan dengan $C(5 + 15 - 1, 15)$ cara. Karena setiap anak mendapat apel dan jeruk, maka jumlah cara pembagian kedua buah itu adalah

$$C(5 + 20 - 1, 20) \times C(5 + 15 - 1, 15) = C(24, 20) \times C(19, 15)$$

cara (nilai $C(24, 20)$ dan $C(19, 15)$ dapat anda selesaikan dengan mudah). ■

Contoh 6.43

Toko roti “Enak” menjual 8 jenis roti. Berapa jumlah cara mengambil 1 lusin roti (1 lusin = 12 buah).

Penyelesaian:

Analogikan 8 jenis roti sebagai 8 kotak. Kita akan mendistribusikan 12 roti itu ke dalam 8 kotak. Setiap kotak mungkin berisi lebih dari 1 buah roti. Di sini $n = 8$ dan $r = 12$, maka

Jumlah cara memilih 12 buah roti itu sama dengan jumlah cara memasukkan 12 buah roti ke dalam 8, yaitu sebanyak

$$C(8 + 12 - 1, 12) = C(19, 12)$$

cara. ■

Contoh 6.44

Andaikan terdapat kumpulan bola yang berwarna merah, biru, dan hijau. Jumlah bola dari masing-masing warna paling sedikit jumlahnya 8 buah.

- Berapa banyak cara memilih 8 buah bola (tanda ada batasan warna)?
- Berapa banyak cara memilih 8 buah bola jika paling sedikit 1 bola dari masing-masing warna terwakili?

Penyelesaian:

- $n = 3, r = 8$, maka $C(n + r - 1, r) = C(3 + 8 - 1, 8) = C(10, 8) = 45$
- Ambil terlebih dulu 1 bola dari masing-masing warna, kemudian ambil 5 bola sisanya. Jumlah cara seluruhnya adalah $C(3 + 5 - 1, 5) = C(7, 5) = 21$ cara. ■

Contoh 6.45

Tiga buah dadu dilempar bersamaan. Berapa banyaknya hasil berbeda yang mungkin?

Penyelesaian:

$$C(6 + 3 - 1, 3) = C(8, 3) = 56.$$
 ■

6.9 Koefisien Binomial

Teorema binomial memberikan cara untuk menjabarkan bentuk perpangkatan $(x + y)^n$, yang dalam hal ini, n adalah bilangan bulat positif. Cara ini digunakan sebagai alternatif bagi penggunaan segitiga Pascal, yaitu:

$$\begin{aligned}(x+y)^0 &= 1 && && && 1 \\(x+y)^1 &= x+y && && && 1 & 1 \\(x+y)^2 &= x^2 + 2xy + y^2 && && 1 & 2 & 1 \\(x+y)^3 &= x^3 + 3x^2y + 3xy^2 + y^3 && && 1 & 3 & 3 & 1 \\(x+y)^4 &= x^4 + 4x^3y + 6x^2y^2 + 4xy^3 + y^4 && && 1 & 4 & 6 & 4 & 1 \\(x+y)^5 &= x^5 + 5x^4y + 10x^3y^2 + 10x^2y^3 + 5xy^4 + y^5 && && 1 & 5 & 10 & 10 & 5 & 1\end{aligned}$$

Aturan untuk menjabarkan bentuk perpangkatan $(x + y)^n$ adalah:

- Suku pertama adalah x^n , sedangkan suku terakhir adalah y^n .

- Pada setiap suku berikutnya, pangkat x berkurang satu sedangkan pangkat y bertambah satu. Untuk setiap suku, jumlah pangkat x dan y adalah n .
- Koefisien untuk $x^{n-k}y^k$, yaitu suku ke- $(k+1)$, adalah $C(n, k)$. Bilangan $C(n, k)$ disebut **koefisien binomial**.

Dari aturan di atas dapat disimpulkan bahwa:

$$(x+y)^n = C(n, 0) x^n + C(n, 1) x^{n-1} y^1 + \dots + C(n, k) x^{n-k} y^k + \dots + C(n, n) y^n = \sum_{k=0}^n C(n, k) x^{n-k} y^k . \quad (6.6)$$

TEOREMA 6.1 (Teorema Binomial). Misalkan x dan y adalah peubah, dan n adalah bilangan bulat tak-negatif. Maka

$$(x+y)^n = \sum_{k=0}^n C(n, k) x^{n-k} y^k$$

Penggunaan teorema binomial diperlihatkan pada contoh-contoh berikut.

Contoh 6.46

Tentukan suku keempat dari penjabaran perpangkatan $(x-y)^5$.

Penyelesaian:

Perlu diperhatikan bahwa $(x-y)^5 = (x+(1-y))^5$.

Oleh sebab itu, suku keempat adalah: $C(5, 3) x^{5-3} (-y)^3 = -10x^2y^3$. ■

Contoh 6.47

Jabarkan $(3x-2)^3$.

Penyelesaian:

Jika didefinisikan $a = 3x$ dan $b = -2$, maka

$$\begin{aligned} (a+b)^3 &= C(3, 0) a^3 + C(3, 1) a^2 b^1 + C(3, 2) a^1 b^2 + C(3, 3) b^3 \\ &= 1 (3x)^3 + 3 (3x)^2 (-2) + 3 (3x) (-2)^2 + 1 (-2)^3 \\ &= 27x^3 - 54x^2 + 36x - 8 \end{aligned}$$

Contoh 6.48

Buktikan bahwa $\sum_{k=0}^n C(n, k) = 2^n$.

Penyelesaian:

Dari persamaan (6.6), ambil $x = y = 1$, sehingga

$$\begin{aligned} &\Leftrightarrow (x+y)^n = \sum_{k=0}^n C(n,k) x^{n-k} y^k \\ &\Leftrightarrow (1+1)^n = \sum_{k=0}^n C(n,k) 1^{n-k} 1^k = \sum_{k=0}^n C(n,k) \\ &\Leftrightarrow 2^n = \sum_{k=0}^n C(n,k) \end{aligned}$$

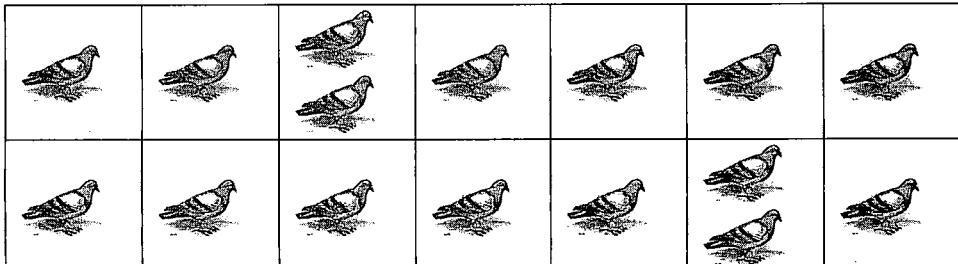
■

6.10 Prinsip Sarang Merpati

Misalkan kita mempunyai kandang burung merpati (*pigeon*) yang memiliki pintu masuk berupa lubang-ubang (*hole*). Satu lubang berarti satu sarang. Setiap sarang biasanya ditempati oleh seekor burung merpati. Misalkan merpati ada 16 ekor sedangkan kandang hanya mempunyai 14 buah sarang. Prinsip sarang merpati (*pigeonhole principle*) menyatakan bahwa paling sedikit terdapat satu sarang yang ditempati oleh dua ekor merpati (Gambar 6.4).

TEOREMA 6.2 (Prinsip Sarang Merpati). Jika $n + 1$ atau lebih objek ditempatkan di dalam n buah kotak, maka paling sedikit terdapat satu kotak yang berisi dua atau lebih objek.

Bukti: Misalkan tidak ada kotak yang berisi lebih dari dua objek. Maka, total jumlah objek paling banyak adalah n . Ini kontradiksi, karena jumlah objek paling sedikit $n + 1$.



Gambar 6.4 Kandang merpati dengan 14 buah sarang (*pigeonhole*) dan 16 ekor merpati.

Prinsip sarang merpati dikemukakan oleh G. Lejeune Dirichlet, seorang matematikawan Jerman, sehingga kadang-kadang dinamakan juga **Prinsip Kotak Dirichlet**, karena Dirichlet sering menggunakan prinsip ini dalam pekerjaannya.

Prinsip sarang merpati, jika diterapkan dengan baik, akan memberikan hanya objek-objek yang ada, dan bukan memberitahukan bagaimana mencari objek tersebut dan berapa banyak [JOH97]. Pada masalah sarang burung merpati, prinsip ini tidak memberitahukan di sarang merpati mana yang berisi lebih dari dua ekor merpati.

Contoh 6.49 sampai 6.51 berikut memperlihatkan penerapan prinsip sarang merpati.

Contoh 6.49

Dari 27 orang mahasiswa, paling sedikit terdapat dua orang yang namanya diawali dengan huruf yang sama, karena hanya ada 26 huruf dalam alfabet. Jika kita menganggap 27 huruf awal dari nama-nama mahasiswa sebagai merpati dan 26 huruf alfabet sebagai 26 buah lubang merpati, kita bisa menetapkan pemasangan 27 huruf awal nama ke 26 huruf alfabet seperti halnya pemasangan merpati ke sarang merpati. Menurut prinsip sarang merpati, beberapa huruf awal alfabet dipasangkan dengan paling sedikit dua huruf awal nama mahasiswa. ■

Contoh 6.50

Misalkan terdapat banyak bola merah, bola putih, dan bola biru di dalam sebuah kotak. Berapa paling sedikit jumlah bola yang diambil dari kotak (tanpa melihat ke dalam kotak) untuk menjamin bahwa sepasang bola yang berwarna sama terambil?

Penyelesaian:

Jika setiap warna dianggap sebagai sarang merpati, maka $n = 3$. Karena itu, jika orang mengambil paling sedikit $n + 1 = 4$ bola (merpati), maka dapat dipastikan sepasang bola yang berwarna sama ikut terambil. Jika hanya diambil 3 buah, maka ada kemungkinan ketiga bola itu berbeda warna satu sama lain. Jadi, 4 buah bola adalah jumlah minimum yang harus diambil dari dalam kotak untuk menjamin terambil sepasang bola yang berwarna sama. ■

Contoh 6.51

Misalkan sebuah turnamen basket diikuti oleh n buah tim yang dalam hal ini setiap tim bertanding dengan setiap tim lainnya dan setiap tim menang paling sedikit satu kali. Tunjukkan bahwa paling sedikit ada 2 tim yang mempunyai jumlah kemenangan yang sama.

Penyelesaian:

Jumlah kemenangan setiap tim paling sedikit 1 kali dan paling banyak $n - 1$ kali. Angka $n - 1$ berkoreponden dengan $n - 1$ buah sarang merpati untuk menampung n ekor merpati (tim basket). Jadi, paling sedikit ada 2 tim basket yang mempunyai jumlah kemenangan sama. ■

Prinsip sarang merpati dapat dirampatkan (*generalized*) sedemikian sehingga jumlah objek dapat merupakan kelipatan jumlah kotak. Misalnya kalau terdapat 20 sarang merpati dan 41 ekor merpati, maka terdapat satu buah sarang yang berisi lebih dari 2 ekor merpati. Hal ini dinyatakan di dalam teorema berikut:

TEOREMA 6.3 (Prinsip Sarang Merpati yang Dirampatkan). Jika M objek ditempatkan di dalam n buah kotak, maka paling sedikit terdapat satu kotak yang berisi minimal $\lceil M/n \rceil$ objek.

Bukti teorema 6.3 tidak diberikan di sini dan ditinggalkan kepada pembaca sebagai latihan. Teorema 6.3 juga menyatakan bahwa jika n buah kotak akan diisi dengan $M = nk + 1$ objek, yang dalam hal ini k adalah bilangan bulat positif, maka paling sedikit terdapat 1 kotak yang berisi minimal $k + 1$ objek.

Contoh 6.52

Di antara 50 orang mahasiswa, terdapat paling sedikit $\lceil 50/12 \rceil = 5$ orang yang lahir pada bulan yang sama. ■

Contoh 6.53

Tinjau kembali Contoh 6.50. Berapa paling sedikit jumlah bola yang harus diambil dari dalam kotak sehingga 3 pasang bola yang setiap pasangnya berwarna sama terambil?

Penyelesaian:

Tiga pasang bola yang setiap pasang berwarna sama berarti semuanya 6 buah bola. Pada masalah ini, n masih tetap sama dengan 3 (yaitu jumlah warna), dan kita perlu mengambil paling sedikit M buah bola untuk memastikan bahwa $\lceil M/3 \rceil = 6$ bola mengandung setiap pasang bola yang berwarna sama. Nilai $M = 3 \cdot 5 + 1 = 16$. Jika kita hanya mengambil 15 bola, maka mungkin saja hanya terambil 2 macam bola yang berwarna sama. Jadi, jumlah 16 buah bola adalah jumlah minimal yang perlu kita ambil dari dalam kotak untuk memastikan bahwa 3 pasang bola yang setiap pasang berwarna sama terambil. ■

6.11 Peluang Diskrit

Antara kombinatorial dan teori peluang (*probability*) sebenarnya terkait erat. Teori peluang banyak menggunakan konsep-konsep di dalam kombinatorial. Sayangnya, kedua bidang ini lahir dari tempat yang kurang patut, yaitu dari arena judi (*gambling games*) – salah satu kasusnya adalah menghitung peluang kemunculan nomor lotre. Meskipun begitu, aplikasi kombinatorial dan teori peluang saat ini meluas ke berbagai bidang ilmu lain maupun dalam kehidupan sehari-hari.

Di dalam upabab 6.1 sudah disebutkan bahwa kombinatorial didasarkan pada percobaan. Hasil percobaan (*outcomes*) diamati dan jumlah semua kemungkinannya

dihitung. Misalnya pada melempar dadu dengan 6 muka, hasil yang muncul untuk satu kali pelemparan ada 6 kemungkinan, yaitu muka 1, 2, 3, 4, 5, dan 6; pada kejadian mengambil 5 buah kartu remi dari 52 buah kartu, terdapat $C(52, 9) = 2.598.960$ kemungkinan kombinasi kartu; pada kejadian menjawab 10 buah pertanyaan pilihan berganda –tiap soal menyediakan 4 pilihan jawaban (a, b, c, d) – maka terdapat 4^{10} kemungkinan jawaban.

Himpunan semua kemungkinan hasil percobaan dinamakan ruang contoh (*sample space*) dari percobaan yang bersangkutan. Setiap hasil percobaan di dalam ruang contoh disebut **titik contoh** (*sample point*). Hasil-hasil percobaan tersebut bersifat saling terpisah (*mutually exclusive*). Dikatakan saling terpisah karena dari seluruh ruang contoh, hanya satu titik contoh yang muncul. Misalnya pada percobaan melempar dadu, hasil percobaan yang muncul hanya salah satu dari 6 muka dadu, tidak mungkin muncul dua muka atau lebih, atau tidak mungkin salah satu dari enam muka dadu tidak ada yang muncul.

Misalkan ruang contoh dilambangkan dengan S dan titik-titik contohnya dilambangkan dengan x_1, x_2, \dots , maka

$$S = \{x_1, x_2, \dots, x_i, \dots\}$$

menyatakan ruang contoh S yang terdiri dari titik-titik contoh x_1, x_2, \dots, x_i , dan seterusnya. Ruang contoh yang jumlah anggotanya terbatas disebut **ruang contoh diskrit** (*discrete sample space*). Peluang terjadinya sebuah titik contoh dinamakan **peluang diskrit** dan disimbolkan dengan $p(x_i)$.

DEFINISI 6.4. Misalkan x_i adalah sebuah titik contoh di dalam ruang contoh S . Peluang bagi x_i adalah ukuran kemungkinan terjadinya atau munculnya x_i di antara titik-titik contoh yang lain di dalam S .

Titik contoh yang mempunyai peluang lebih besar berarti kemungkinan terjadinya lebih besar pula, sedangkan titik contoh yang peluangnya lebih kecil berarti kemungkinan terjadinya juga lebih kecil.

Peluang diskrit mempunyai sifat sebagai berikut:

1. $0 \leq p(x_i) \leq 1$, yaitu nilai peluang adalah bilangan tidak negatif dan selalu lebih kecil atau sama dengan 1.
2. $\sum_{i=1}^{|S|} p(x_i) = 1$, yaitu jumlah peluang semua titik contoh di dalam ruang contoh S adalah 1.

Contoh 6.54

Pada pelemparan dadu, $S = \{1, 2, 3, 4, 5, 6\}$. Peluang munculnya setiap angka adalah sama yaitu $1/6$. ■

Contoh 6.55

Uang logam mempunyai dua buah muka, yaitu gambar (g) dan angka (a). Jika sebuah koin uang logam dilempar, maka peluang munculnya muka gambar = $1/2$, begitu juga peluang munculnya muka angka. Jika dua buah koin uang logam dilempar, maka ruang contohnya adalah $S = \{aa, gg, ag, ga\}$. Peluang setiap titik contoh adalah

$$p(aa) = p(gg) = p(ag) = p(ga) = \frac{1}{4}$$

Contoh 6.56

Sebuah koin yang mempunyai sisi A dan sisi B dilempar ke atas sebanyak empat kali. Berapakah peluang munculnya sisi A sebanyak tiga kali?

Penyelesaian:

Dari Contoh 6.30 kita sudah mengetahui bahwa jumlah kemungkinan munculnya sisi A sebanyak tiga kali adalah $C(4, 3) = 4$. Jumlah seluruh hasil percobaan adalah $2 \times 2 \times 2 \times 2 = 16$, sehingga peluang munculnya sisi A sebanyak tiga kali adalah $4/16 = 1/4$. ■

Kejadian (event) –disimbolkan dengan E – adalah himpunan bagian dari ruang contoh. Misalnya pada percobaan melempar dadu, kejadian munculnya angka ganjil adalah $E = \{1, 3, 5\}$, kejadian munculnya angka 1 adalah $E = \{1\}$. Kejadian yang hanya mengandung satu titik contoh disebut kejadian sederhana (*simple event*) dan kejadian yang mengandung lebih dari satu titik contoh disebut kejadian majemuk (*compound event*) [LIU85].

Suatu kejadian dikatakan terjadi jika salah satu dari titik contoh di dalam kejadian tersebut terjadi. Peluang terjadinya suatu kejadian didefinisikan sebagai berikut:

DEFINISI 6.5. Peluang kejadian E di dalam ruang contoh S adalah $p(E) = |E| / |S|$.

Peluang kejadian E juga dapat diartikan sebagai jumlah peluang semua titik contoh di dalam E . Jadi, kita dapat menuliskan bahwa

$$p(E) = \frac{|E|}{|S|} = \sum_{x_i \in E} p(x_i) \quad (6.7)$$

Contoh 6.57

Pada percobaan melempar dadu, $S = \{1, 2, 3, 4, 5, 6\}$. Kejadian munculnya angka ganjil adalah $E = \{1, 3, 5\}$. Di sini $|S| = 6$ dan $|E| = 3$. Kejadian munculnya angka ganjil adalah $3/6 = \frac{1}{2}$. Kita juga dapat menghitung peluang munculnya angka ganjil dengan melihat bahwa peluang setiap titik contoh di dalam E adalah $1/6$, sehingga $p(E) = 1/6 + 1/6 + 1/6 = 3/6 = \frac{1}{2}$. ■

Contoh 6.58

Dua buah dadu dilemparkan. Berapa peluang munculnya angka-angka dadu yang jumlahnya sama dengan 8?

Penyelesaian:

Jumlah hasil percobaan yang muncul adalah (gunakan kaidah perkalian) $6 \times 6 = 36$. Ruang contohnya adalah $S = \{(1,1), (1, 2), \dots, (1, 6), (2,1), (2, 2), \dots, (2,6), \dots, (6,1), (6,2), \dots, (6,6)\}$, semuanya ada 36 elemen. Kejadian munculnya jumlah angka sama dengan 8 adalah $E = \{(2,6), (3,5), (4,4), (5,3), (6,2)\}$. Peluang munculnya jumlah angka sama dengan 8 adalah $5/36$. ■

Contoh 6.59

Kartu remi (poker) seluruhnya 52 buah kartu. Keseluruhan kartu ini terdiri dari 13 jenis kartu, setiap jenis terdiri dari 4 buah kartu. Tiga belas jenis kartu tersebut adalah: 2, 3, 4, 5, 6, 7, 8, 9, 10, joker, ratu, raja, dan as. Setiap pemain remi mendapatkan 5 buah kartu. Berapa peluang dari 5 kartu tersebut mengandung 4 kartu dari jenis yang sama?

Penyelesaian:

Jumlah cara mengambil 5 kartu sembarang dari 52 buah kartu = $C(52, 5)$ (ini adalah jumlah titik contoh). Jumlah cara mengambil satu jenis kartu dari 13 jenis yang ada = $C(13, 1)$. Jumlah cara mengambil 4 kartu dari 4 kartu yang sejenis = $C(4, 4)$. Jumlah cara mengambil satu kartu lagi dari 48 kartu yang tersisa = $C(48, 1)$. Sehingga, peluang dari 5 kartu tersebut mengandung 4 kartu sejenis = $\frac{C(13,1) \times C(4,4) \times C(48,1)}{C(52,5)} = 0.000024$ ■

Contoh 6.60

(berkaitan dengan Contoh 6.59) Berapa peluang dari 5 kartu tersebut mengandung 4 kartu as?

Penyelesaian:

Pada soal ini, jenis kartu sudah ditentukan, yaitu kartu as, maka hanya ada satu cara mengambil jenis kartu as.

Jumlah cara mengambil 4 kartu dari 4 kartu as = $C(4, 4)$

Jumlah cara mengambil satu kartu lagi dari 48 kartu yang tersisa = $C(48, 1)$

Jumlah cara mengambil 5 kartu sembarang dari 52 buah kartu = $C(52, 5)$

Peluang dari 5 kartu tersebut mengandung 4 kartu as = $\frac{1 \times C(4,4) \times C(48,1)}{C(52,5)} = 0.0000185$ ■

Contoh 6.61

Sepuluh buah buku disusun di atas sebuah rak. Kesepuluh buku itu beragam topiknya, ada buku tentang fisika, buku kimia, buku biologi, buku matematika, dan buku sosiologi. Berapa peluang bahwa dari 10 buku itu tepat ada 2 buah buku untuk setiap topik?

Penyelesaian:

Misalkan susunan 10 buah buku tersebut seperti di bawah ini:

— — — — — — — — — —
1 2 3 4 5 6 7 8 9 10

Buku ke-1 ada 5 kemungkinan topik, buku ke-2 ada 5 kemungkinan topik, dan seterusnya, buku ke-10 ada 5 kemungkinan topik. Total semua kemungkinan topik buku adalah 5^{10} . Ini adalah jumlah titik contoh di dalam ruang contoh.

Banyaknya titik contoh dengan 2 buku untuk setiap topik adalah $\frac{10!}{2!2!2!2!2!}$ sehingga peluang dari 10 buku itu tepat ada 2 buah buku untuk setiap topik

$$= \frac{10!}{2!2!2!2!2!} = 0.0174$$

Konsep-konsep pada teori himpunan dapat diterapkan pada peluang diskrit. Misalkan diketahui dua buah himpunan A dan B adalah dua buah kejadian di dalam ruang contoh S :

1. Kejadian bahwa A dan B terjadi sekaligus berarti sama dengan munculnya salah satu titik contoh di dalam himpunan $A \cap B$. Peluang terjadinya kejadian A dan B adalah

$$p(A \cap B) = \sum_{x_i \in A \cap B} p(x_i) \quad (6.8)$$

2. Kejadian bahwa A atau B atau keduanya terjadi berarti sama dengan munculnya salah satu titik contoh di dalam $A \cup B$. Peluang terjadinya kejadian A atau B atau keduanya adalah

$$p(A \cup B) = \sum_{x_i \in A \cup B} p(x_i) \quad (6.9)$$

3. Kejadian bahwa A terjadi tetapi B tidak berarti sama dengan munculnya salah satu titik contoh di dalam $A - B$. Peluang terjadinya kejadian A tetapi B tidak adalah

$$p(A - B) = \sum_{x_i \in A - B} p(x_i) \quad (6.10)$$

4. Kejadian bahwa salah satu dari A dan B terjadi namun bukan keduanya berarti sama dengan munculnya salah satu titik contoh di dalam $A \oplus B$. Peluang terjadinya salah satu dari A dan B namun bukan keduanya adalah

$$p(A \oplus B) = \sum_{x_i \in A \oplus B} p(x_i) \quad (6.11)$$

5. (Komplemen) Peluang bahwa kejadian \bar{A} , komplemen dari kejadian A , terjadi adalah

$$p(\bar{A}) = 1 - p(A) \quad (6.12)$$

Persamaan (6.12) dapat dibuktikan sebagai berikut:

$$p(\bar{A}) = \frac{|S| - |A|}{|S|} = 1 - \frac{|A|}{|S|} = 1 - p(A) \quad \blacksquare$$

Dengan menggunakan prinsip inklusi-eksklusi kita juga dapat memperlihatkan bahwa

$$p(A \cup B) = p(A) + p(B) - p(A \cap B)$$

Bukti:

Prinsip inklusi-eksklusi untuk operasi gabungan dua buah himpunan menyatakan bahwa

$$|A \cup B| = |A| + |B| - |A \cap B|$$

Dalam hal ini,

$$\begin{aligned} p(A \cup B) &= \frac{|A \cup B|}{|S|} \\ &= \frac{|A| + |B| - |A \cap B|}{|S|} \\ &= \frac{|A|}{|S|} + \frac{|B|}{|S|} - \frac{|A \cap B|}{|S|} \\ &= p(A) + p(B) - p(A \cap B) \quad \blacksquare \end{aligned}$$

Contoh 6.62

Dari 100 orang mahasiswa ITB yang hadir dalam sebuah diskusi, 80 orang laki-laki dan 20 orang perempuan. Di antara mahasiswa pria terdapat 35 orang yang memakai jaket almamater (pja) dan 45 orang tidak memakai jaket tersebut ($ptja$), dan di antara mahasiswa wanita terdapat 8 orang yang memakai jaket alamamater (wja) dan 12 orang yang tidak memakainya ($wtja$). Kita ingin memilih salah seorang dari mahasiswa tersebut sebagai notulen. Maka ruang contohnya adalah $S = \{pja, ptja, wja, wtja\}$. Peluang setiap mahasiswa dari kategori terpilih sebagai notulen adalah

$$p(pja) = 35/100 = 0.35; p(ptja) = 45/100 = 0.45;$$
$$p(wja) = 8/100 = 0.08; p(wtja) = 12/100 = 0.12$$

Misalkan A adalah kejadian terpilihnya mahasiswa pria, dan B adalah kejadian terpilihnya mahasiswa(i) yang memakai jaket alamamater, maka

$$p(A) = 0.35 + 0.45 = 0.75$$
$$p(B) = 0.35 + 0.09 = 0.44$$

$A \cap B$ menyatakan kejadian terpilihnya mahasiswa pria yang memakai jaket almamater:

$$p(A \cap B) = 0.35$$

$A \cup B$ menyatakan kejadian terpilihnya mahasiswa pria atau mahasiswa(i) yang memakai jaket almamater:

$$p(A \cup B) = 0.35 + 0.45 + 0.08 = 0.88$$

$A \oplus B$ menyatakan kejadian terpilihnya mahasiswa pria yang tidak memakai jaket almamater atau mahasiswi yang memakai jaket:

$$p(A \oplus B) = 0.45 + 0.12 = 0.57$$

$A - B$ menyatakan kejadian terpilihnya mahasiswa pria tetapi tidak memakai jaket:

$$p(A - B) = 0.45$$
 ■

Contoh 6.63

Di antara 100 bilangan bulat positif pertama, berapa peluang memilih secara acak sebuah bilangan yang habis dibagi 3 atau 5?

Penyelesaian:

Misalkan,

- A menyatakan kejadian bilangan bulat yang dipilih habis dibagi 3,
 B menyatakan kejadian bilangan bulat yang dipilih habis dibagi 5,

$A \cap B$ menyatakan kejadian bilangan bulat yang dipilih habis dibagi 3 dan 5 (yaitu bilangan bulat yang habis dibagi oleh KPK – Kelipatan Persekutuan Terkecil – dari 3 dan

5, yaitu 15), maka $A \cup B$ menyatakan kejadian bilangan bulat yang dipilih habis dibagi 3 atau 5.

Terlebih dahulu kita harus menghitung

$$|A| = \lfloor 100/3 \rfloor = 33, \quad |B| = \lfloor 100/5 \rfloor = 20, \quad |A \cap B| = \lfloor 100/15 \rfloor = 6$$

untuk mendapatkan

$$p(A \cup B) = p(A) + p(B) - p(A \cap B) = 33/100 + 20/100 - 6/100 = 0.47$$

Jadi, peluang bilangan yang dipilih habis dibagi 3 atau 5 adalah 0.47. ■

Contoh 6.64

Dari 8 bit (atau 1 byte) yang dibangkitkan secara acak, berapa peluang bahwa byte tersebut tidak dimulai dengan ‘11’?

Penyelesaian:

Misalkan A menyatakan kejadian bahwa byte yang dibangkitkan dimulai dengan ‘11’. Maka, \bar{A} menyatakan kejadian bahwa byte yang dibangkitkan tidak dimulai dengan ‘11’. Jumlah byte yang dimulai dengan ‘11’ adalah $2^6 = 64$ buah karena 2 posisi pertama sudah diisi dengan ‘11’, sehingga kita cukup mengisi 6 posisi bit lainnya. Jadi, $|A| = 64$. Ruang contoh S adalah himpunan semua bit yang panjangnya 8, di sini $|S| = 2^8 = 256$. Maka, peluang byte yang dibangkitkan tidak dimulai dengan ‘11’ adalah

$$p(\bar{A}) = 1 - p(A) = 1 - \frac{|A|}{|S|} = 1 - 64/256 = 192/256$$

Contoh 6.65

[ROS99] Dari 8 bit (atau 1 byte) yang dibangkitkan secara acak, berapa peluang bahwa paling bit-bit ini mengandung sedikitnya satu buah bit 0?

Penyelesaian:

Misalkan A menyatakan kejadian bahwa byte yang dibangkitkan mengandung sedikitnya sebuah bit 0. Maka, \bar{A} menyatakan kejadian bahwa byte yang dibangkitkan semuanya mengandung bit 1. Ruang contoh S adalah himpunan semua bit yang panjangnya 8, di sini $|S| = 2^8 = 256$. Hanya ada satu buah byte yang semuanya mengandung bit 1, sehingga $p(\bar{A}) = 1/256$. Maka, peluang byte yang dibangkitkan mengandung sedikitnya satu buah bit 0 adalah

$$p(A) = 1 - p(\bar{A}) = 1 - \frac{|\bar{A}|}{|S|} = 1 - 1/256 = 255/256$$

6.12 Ragam Soal dan Penyelesaian

Kombinatorial kaya dengan ragam persoalan. Di dalam upabab 6.12 ini diberikan beragam soal tambahan agar pembaca dapat lebih memahami teori yang sudah dikemukakan sebelum ini.

Contoh 6.66

Dari 100.000 buah bilangan bulat positif pertama, berapa banyak bilangan yang mengandung tepat 1 buah angka 3, 1 buah angka 4, dan 1 buah angka 5?

Penyelesaian:

Bilangan 100.000 tidak memenuhi, karena tidak mengandung angka 4 dan 5. Jadi bilangan bulat yang akan ditinjau hanya terdiri dari 5 angka. Ada 5 cara untuk menempatkan angka 5, sehingga sisa tempat yang kosong tinggal 4 angka. Selanjutnya, ada 4 cara untuk menempatkan angka 4, sehingga sisa tempat kosong tinggal 3 angka. Terakhir, ada 3 cara untuk menempatkan angka 3, sehingga sisa tempat kosong tinggal 2 angka. Posisi selain angka 3, 4, dan 5 dapat diisi dengan angka lain dan boleh berulang. Jadi untuk kedua tempat yang masih kosong dapat diisi masing-masing dengan 7 angka. Banyaknya bilangan bulat yang dapat dibentuk sesuai dengan aturan tersebut adalah $5 \times 4 \times 3 \times 7 \times 7 = 2940$. ■

Contoh 6.67

Ada 10 soal di dalam ujian akhir *Matematika Diskrit*. Berapa banyak cara pemberian nilai (bilangan bulat) pada setiap soal jika jumlah nilai keseluruhan soal adalah 100 dan setiap soal mempunyai nilai paling sedikit 5. (Khusus untuk soal ini, nyatakan jawaban akhir anda dalam $C(a, b)$ saja, tidak perlu dihitung nilainya)

Penyelesaian:

Misalkan 10 soal digambarkan sebagai 10 buah kotak. Kita akan memasukkan 100 buah bola (nilai) yang sama warnanya ke dalam 10 kotak (soal) itu, setiap kotak (soal) berisi minimal 5. Ini adalah soal kombinasi dengan pengulangan. Mula-mula, masukkan 5 buah bola ke dalam setiap kotak, sehingga jumlah bola yang tersisa adalah $100 - 50 = 50$. Sisa 50 buah bola ini didistribusikan kembali ke sepuluh kotak tersebut, di sini $n = 10$ dan $r = 50$, sehingga banyaknya cara pemberian nilai bilangan bulat ke sepuluh soal tersebut adalah $C(10 + 50 - 1, 50) = C(59, 50)$. ■

Contoh 6.68

Berapa banyak *string* yang dapat dibentuk dari huruf-huruf kata “CONGRESS” sedemikian sehingga dua buah huruf “S” tidak terletak berdampingan?

Penyelesaian:

String “CONGRESS” disusun oleh 8 buah huruf, dan terjadi pengulangan dua kali untuk huruf “S”. Jika kedua huruf “S” boleh sembarang letaknya (tidak ada aturan khusus untuk huruf “S”), maka jumlah *string* berbeda yang dapat dibentuk adalah:

$$P(8; 1, 1, 1, 1, 1, 1, 1, 2) = \frac{8!}{1! \cdot 1! \cdot 1! \cdot 1! \cdot 1! \cdot 1! \cdot 2!} = \frac{8 \cdot 7 \cdot 6 \cdot 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1}{2!} = 8 \cdot 7 \cdot 6 \cdot 5 \cdot 4 \cdot 3 = 20160$$

Jika kedua huruf "S" harus berdampingan, maka jumlah *string* berbeda yang terjadi adalah sama dengan permutasi dari 7 huruf dari 7 huruf yang tersedia, dimana tidak ada karakter yang berulang yaitu:

$$P(7, 7) = 7! = 7 \cdot 6 \cdot 5 \cdot 4 \cdot 3 \cdot 2 = 5040$$

Jadi jumlah *string* berbeda yang dapat dibentuk dari huruf-huruf tersebut apabila dua huruf "S" tidak boleh berdampingan adalah

$$20160 - 5040 = 15120 \text{ buah.}$$

Contoh 6.69

Berapa banyak solusi bilangan bulat dari $x_1 + x_2 + x_3 = 10$ jika $0 \leq x_1 \leq 2$, $x_2 > 1$, dan $x_3 \geq 0$?

Penyelesaian:

Analogikan dengan membagi 10 buah bola yang identik ke dalam 3 buah kotak, sebutlah kotak x_1 , x_2 , dan kotak x_3 . Nilai x_1 ada 3 kemungkinan: 0, 1 dan 2. Untuk masing-masing nilai x_1 , kita rinci perhitungan untuk x_i lainnya:

- (i) Kasus $x_1 = 0$, persamaan menjadi $x_2 + x_3 = 10$. Isikan 2 ke dalam x_2 (karena $x_2 > 1$). Bagikan 8 buah bola sisa ke dalam x_2 dan x_3 , semuanya ada $C(2 + 8 - 1, 8) = C(9, 8)$ cara.
- (ii) Kasus $x_1 = 1$, persamaan menjadi $x_2 + x_3 = 9$. Isikan 2 ke dalam x_2 (karena $x_2 > 1$). Bagikan 7 buah bola sisa ke dalam x_2 dan x_3 , semuanya ada $C(2 + 7 - 1, 7) = C(8, 7)$ cara.
- (iii) Kasus $x_1 = 2$, persamaan menjadi $x_2 + x_3 = 8$. Isikan 2 ke dalam x_2 (karena $x_2 > 1$). Bagikan 6 buah bola sisa ke dalam x_2 dan x_3 , semuanya ada $C(2 + 6 - 1, 6) = C(7, 6)$ cara.

Jumlah solusi seluruhnya = $C(9, 8) + C(8, 7) + C(7, 6) = 9 + 8 + 7 = 24$ buah.

Contoh 6.70

Berapa banyak solusi bilangan bulat tak-negatif dari $x_1 + x_2 + x_3 = 11$ jika $x_1 > 1$, $x_2 \leq 4$, dan $x_3 = 1$?

Penyelesaian:

Soal ini serupa dengan Contoh 6.70 di atas. Isikan 1 bola ke dalam kotak x_3 karena nilai $x_3 = 1$, sehingga $x_1 + x_2 = 10$. Karena nilai x_1 minimum 2, masukkan 2 buah bola ke dalam x_1 , sehingga sisa bola yang belum dibagikan = $10 - 2 = 8$.

Perhatikan bahwa nilai x_2 maksimum 4. Kita juga dapat menganalisis kasus nilai-nilai x_2 , yaitu 0, 1, 2, 3, dan 4 seperti penyelesaian Contoh 6.54, namun ada cara lain yang lebih ringkas sebagai berikut:

- (i) Jika nilai $x_2 \geq 0$ (x_2 minimum 0), maka ada 8 nilai lagi yang harus didistribusikan ke x_1 dan x_2 , yang dalam hal ini $n = 2$ dan $r = 8$, sehingga $C(2 + 8 - 1, 8) = C(9, 8) = 9$
(ii) Jika nilai $x_2 \geq 5$ (x_2 minimum 5), maka ada $8 - 5 = 3$ nilai lagi yang harus didistribusikan ke x_1 dan x_2 , yang dalam hal ini $n = 2$ dan $r = 3$, sehingga $C(2 + 3 - 1, 3) = C(4, 3) = 4$

Jadi, banyaknya solusi bilangan bulat tak-negatif adalah $9 - 4 = 5$ buah. ■

Contoh 6.71

Perlihatkan bahwa $\sum_{k=0}^{2^k} C(n, k) = 3^n$

Penyelesaian:

$$(x + y)^n = \sum_{k=0}^n C(n, k) x^{n-k} y^k$$

Ambil $x = 2$ dan $y = 1$, sehingga

$$(2 + 1)^n = \sum_{k=0}^n C(n, k) 2^k 1^{n-k} = \sum_{k=0}^n C(n, k) 2^k$$

$$3^n = \sum_{k=0}^n C(n, k) 2^k$$
 ■

Contoh 6.72

Tersedia 6 huruf: a, b, c, d, e, f . Berapa jumlah pengurutan 3 huruf jika:

- (a) tidak ada huruf yang diulang;
(b) boleh ada huruf yang berulang;
(c) tidak boleh ada huruf yang diulang, tetapi huruf e harus ada;
(d) boleh ada huruf yang berulang, huruf e harus ada;

Penyelesaian:

- (a) $(6)(5)(4) = 120$ atau $P(6, 3) = 120$
(b) $(6)(6)(6) = 216$
(c) $3(5 \cdot 4) = 60$
(d) $(6)(6) + (5)(6) + (5)(5) = 91$ ■

Contoh 6.73

- (a) Berapa banyak bilangan genap 2-angka?
(b) Berapa banyak bilangan ganjil 2-angka dengan setiap angka berbeda?

Penyelesaian:

- (a) $(9)(5) = 45$
(b) $(8)(5) = 40$ ■

Contoh 6.74

Berapa banyak bilangan empat-angka yang tidak mengandung angka-angka yang berulang?

Penyelesaian:

$$(9)(9)(8)(7) = 4536$$

Contoh 6.75

Misalkan $|A| = m$ dan $|B| = n$. Berapa banyak fungsi yang dapat dibuat dari himpunan A ke himpunan B ?

Penyelesaian:

Ingatlah kembali definisi fungsi di dalam Bab 3 bahwa setiap elemen pada himpunan A harus mempunyai pemetaan ke satu dan hanya satu elemen di himpunan B . Elemen pertama di A mempunyai n kemungkinan peta di B , elemen kedua di A mempunyai n kemungkinan peta di B . Begitu seterusnya sehingga jumlah fungsi yang dapat dibuat dari A ke B (dengan menerapkan kaidah perkalian) adalah:

$$n \times n \times n \times \dots \times n \quad (\text{sebanyak } m \text{ kali}) = n^m \text{ buah.}$$

Contoh 6.76

Misalkan A dan B himpunan, $|A| = m$, $|B| = n$, dan $m \leq n$. Berapa banyak fungsi satu-ke-satu (*one-to-one*) yang dapat dibuat dari himpunan A ke himpunan B ?

Penyelesaian:

Pada fungsi satu-ke-satu, setiap elemen di himpunan A hanya dipetakan ke satu elemen di B dan tidak ada dua elemen himpunan A yang memiliki bayangan sama di B . Misalkan elemen-elemen himpunan A adalah a_1, a_2, \dots, a_m . Untuk a_1 ada n pilihan bayangan di B , untuk a_2 ada $n - 1$ pilihan bayangan di B (karena bayangan untuk a_1 tidak dapat digunakan lagi), ..., untuk a_k ada $n - k + 1$ pilihan bayangan di B , ..., untuk a_m ada sebanyak $n - m + 1$ pilihan bayangan. Dengan kaidah perkalian, maka akan terdapat sebanyak $n(n - 1)(n - 2) \dots (n - m + 1)$ fungsi satu-ke-satu dari himpunan A ke himpunan B .

Contoh 6.77

Sebuah mobil mempunyai 4 tempat duduk. Berapa banyak cara 3 orang didudukkan jika diandaikan satu orang harus duduk di kursi sopir?

Penyelesaian:

Ada tiga pilihan orang untuk menjadi sopir. Jika satu orang sudah duduk di tempat sopir, maka dua orang lagi akan didudukkan pada 3 posisi tempat duduk yang lain. Ini dilakukan dalam $P(3, 2)$ cara. Jumlah cara mendudukkan 3 orang seluruhnya adalah $3 \times P(3, 2) = 9$ cara.

Contoh 6.78

Tentukan banyak cara pengaturan agar 3 orang mahasiswa Jurusan Teknik Informatika (IF), 4 orang mahasiswa Teknik Kimia (TK), 4 orang mahasiswa Teknik Geologi (GL), dan 2 orang mahasiswa Farmasi (FA) dapat duduk dalam satu baris sehingga mereka dari departemen yang sama duduk berdampingan?

Penyelesaian:

Ada $4!$ atau $P(4, 4)$ cara untuk menyusun tempat duduk dari 4 departemen yang berbeda. Untuk setiap kasus, ada $3!$ cara untuk mengatur duduk mahasiswa IF, $4!$ cara untuk mengatur duduk mahasiswa TK, $4!$ cara untuk mengatur duduk mahasiswa GL, dan $2!$ cara untuk mengatur duduk mahasiswa FA.

Jumlah cara pengaturan duduk seluruhnya: $4! \times 3! \times 4! \times 4! \times 2!$ ■

Contoh 6.79

100 orang mahasiswa dikirim ke 5 negara, masing-masing negara 20 orang mahasiswa. Berapa banyak cara pengiriman mahasiswa?

Penyelesaian:

$$P(100; 20, 20, 20, 20, 20) = \frac{100!}{(20!)(20!)(20!)(20!)(20!)} \text{ cara}$$

Cara lain:

$$C(100, 20)C(80, 20)C(60, 20)C(40, 20)C(20, 20) = \frac{100!}{(20!)(20!)(20!)(20!)(20!)} \text{ cara.} \quad ■$$

Contoh 6.80

Berapa jumlah cara n wanita dan n pria didudukkan pada satu baris yang berisi $2n$ buah kursi jika pria dan wanita duduk berselang-seling.

Penyelesaian:

- Dudukkan n pria pada kursi bermotor ganjil \rightarrow ada $n!$ cara atau $P(n, n)$. Setelah pria duduk, wanita dapat didudukkan pada kursi bermotor genap dengan $n!$ cara. Jadi ada $(n!)^2$ cara pengaturan kursi.
- Dudukkan n pria pada kursi bermotor genap \rightarrow ada $n!$ cara atau $P(n, n)$. Setelah pria duduk, wanita dapat didudukkan pada kursi bermotor ganjil dengan $n!$ cara. Jadi ada $(n!)^2$ cara pengaturan kursi.

$$\text{Total cara pengaturan duduk} = (\text{i}) + (\text{ii}) = (n!)^2 + (n!)^2 = 2(n!)^2 \quad ■$$

Contoh 6.81

- Berapa banyak cara memasang 10 pesawat telepon yang identik di 12 kamar hotel?
- Berapa banyak cara memasang 10 pesawat telepon (4 berwarna merah, 3 putih, 3 hijau) di 12 kamar?

Penyelesaian:

(a) $C(12, 10)$

(b) $P(12; 4, 3, 3, 2) = \frac{12!}{(4!)(3!)(3!)(2!)}$

Cara lain: $C(12, 4)C(8, 3)C(5, 3)C(2, 2) = \frac{12!}{(4!)(3!)(3!)(2!)}$ ■

Contoh 6.82

Berapa jumlah permutasi huruf-huruf yang membentuk kata ‘ASBESTOS’?

Penyelesaian:

Cara 1: $P(8; 1, 3, 1, 1, 1, 1) = \frac{8!}{(1!)(3!)(1!)(1!)(1!)(1!)} = 6720$ buah.

Cara 2: huruf-huruf yang tidak berulang: *A, B, E, O, T*. Ada $P(8, 5)$ cara untuk menempatkan kelima huruf tersebut. Ada 1 cara untuk menempatkan 3 huruf *S* ke 3 posisi lainnya. Jumlah permutasi seluruhnya adalah

$$= P(8, 5) \times 1 = P(8, 5) = \frac{8!}{3!} = 6720 \text{ buah.} \quad \blacksquare$$

Contoh 6.83

Ada 5 orang mahasiswa jurusan Matematika dan 7 orang mahasiswa jurusan Informatika. Berapa banyak cara membentuk panitia yang terdiri dari 4 orang jika:

- (a) tidak ada batasan jurusan
- (a) semua anggota panitia harus dari jurusan Matematika
- (b) semua anggota panitia harus dari jurusan Informatika
- (c) semua anggota panitia harus dari jurusan yang sama
- (d) 2 orang mahasiswa per jurusan harus mewakili.

Penyelesaian:

- (a) $C(12, 4)$
- (b) $C(5,4) C(7,0)$
- (c) $C(7,4) C(5,0)$
- (d) $C(5,4) C(7,0) + C(7,4) C(5,0)$
- (e) $C(5,2) C(7,2)$

Contoh 6.84

Tujuh belas orang mahasiswa akan pergi ke sebuah pesta, dan terdapat 5 buah kendaraan untuk mereka gunakan. Jumlah tempat duduk yang kosong pada setiap kendaraan adalah 4, 3, 2, 5, dan 1, sehingga ada 2 orang mahasiswa yang tidak terangkut. Berapa jumlah cara mengangkut seluruh mahasiswa kecuali 2 orang ke pesta?

Penyelesaian:

$$C(17; 4, 3, 2, 5, 1, 2) = \frac{17!}{(4!)(3!)(2!)(5!)(1!)(2!)}$$

Contoh 6.85

Berapa banyak cara membagikan 5 buah kartu remi yang diambil dari tumpukan 52 buah kartu ke masing-masing dari 4 orang?

Penyelesaian:

$$C(52, 5) \times C(47, 5) \times C(42, 5) \times C(37, 5) = \\ \frac{52!}{(5!)(47!)} \cdot \frac{47!}{(5!)(42!)} \cdot \frac{42!}{(5!)(37!)} \cdot \frac{37!}{(5!)(32!)} = \frac{52!}{(5!)(5!)(5!)(5!)(32!)}$$

Contoh 6.86

Di perpustakaan Teknik Informatika terdapat 3 jenis buku: buku Algoritma dan Pemrograman, buku Matematika Diskrit, dan buku Basisdata. Perpustakaan memiliki paling sedikit 10 buah buku untuk masing-masing jenis. Berapa banyak cara memilih 10 buah buku?

Penyelesaian:

Diketahui $n = 3$, $r = 10$.

Jumlah cara memilih buku = $C(3 + 10 - 1, 10) = C(12, 10) = 66$

Contoh 6.87

Berapa banyak cara membentuk sebuah panitia yang beranggotakan 5 orang yang dipilih dari 7 orang pria dan 5 orang wanita, jika di dalam panitia tersebut paling sedikit beranggotakan 2 orang wanita?

Penyelesaian:

Jumlah wanita di dalam panitia: 2, 3, 4, atau 5 orang.

Pilih 2 orang dari 5 wanita → ada $C(5, 2)$ cara; sisanya 3 orang dipilih dari 7 pria
→ $C(7, 3)$ cara.

Pilih 3 orang dari 5 wanita → ada $C(5, 3)$ cara; sisanya 2 orang dipilih dari 7 pria
→ $C(7, 2)$ cara.

Pilih 4 orang dari 5 wanita → ada $C(5, 4)$ cara; sisanya 1 orang dipilih dari 7 pria
→ $C(7, 1)$ cara.

Pilih 5 orang dari 5 wanita → ada $C(5, 5)$ cara; sisanya 0 orang dipilih dari 7 pria
→ $C(7, 0)$ cara.

Jumlah cara pembentukan panitia seluruhnya =

$$C(5, 2)C(7, 3) + C(5, 3)C(7, 2) + C(5, 4)C(7, 1) + C(5, 5)C(7, 0)$$

Contoh 6.88

Dari sejumlah besar koin 25-an, 50-an, 100-an, dan 500-an, berapa banyak cara lima koin dapat diambil?

Penyelesaian:

Ini adalah kombinasi dengan membolehkan pengulangan, karena koin tertentu dapat terambil lebih dari sekali. Di sini $n = 4$, $r = 5$, sehingga seluruhnya ada $C(4 + 5 - 1, 5) = C(8, 5)$ cara untuk mengambil lima koin. ■

Contoh 6.89

Tentukan banyaknya cara agar 4 buku matematika, 3 buku sejarah, 3 buku kimia, dan 2 buku sosiologi dapat disusun dalam satu baris sedemikian sehingga (untuk masing-masing soal)

- (a) semua buku yang topiknya sama letaknya bersebelahan,
- (b) urutan buku dalam susunan bebas.

Penyelesaian:

(a) Soal ini mirip dengan Contoh 6.77. Ada $4!$ atau $P(4, 4)$ cara untuk menyusun buku-buku dari 4 kategori berbeda. Untuk setiap kategori buku, ada $4!$ cara untuk mengatur buku-buku matematika, $3!$ cara untuk mengatur buku-buku sejarah, $3!$ cara untuk mengatur buku-buku kimia, dan $2!$ cara untuk mengatur buku-buku sosiologi.

Jumlah cara pengaturan buku seluruhnya: $4! \times 4! \times 3! \times 3! \times 2! = 41.472$ cara.

(b) Jika masalahnya adalah seperti soal b, maka ini adalah masalah permutasi dengan pengulangan. Jumlah cara pengaturan buku adalah

$$P(12; 4, 3, 3, 2) = \frac{12!}{4! \times 3! \times 3! \times 2!} = 277.200 \text{ cara.}$$
 ■

Contoh 6.90

Panjang nomor telpon di kota Bandung adalah 7 angka ($XXX-XXXX$, yang dalam hal ini X merepresentasikan sembarang angka 0 sampai 9)). Tiga angka pertama menyatakan kode area. Kode area untuk wilayah Dago adalah 250, yaitu 250-XXXX. (a) Berapa jumlah nomor sambungan telepon yang dapat didefinisikan oleh PT Telkom? (b) Andaikan jumlah rumah atau kantor di wilayah Dago adalah 50.000 buah dan diasumsikan setiap rumah atau kantor meminta satu nomor sambungan telepon. Apakah panjang 7 angka untuk nomor telepon di kota Bandung masih mencukupi?

Penyelesaian:

- (a) Untuk nomor telepon 250-XXXX, X yang pertama dapat diisi dengan 10 cara (0 sampai 9), X yang kedua juga 10 cara, demikian pula untuk angka X ketiga dan keempat. Jumlah nomor telepon yang dapat didefinisikan adalah $10^4 = 10.000$.
- (b) Kita dapat melihat bahwa panjang 7 angka pada nomor telepon di kota Bandung tidak mencukupi untuk 100.000 permintaan. Perlu tambahan satu angka lagi sehingga

menjadi 8 angka, yaitu 250-XXXXX. Dengan tambahan satu angka lagi, maka jumlah nomor telepon yang dapat didefinisikan adalah $10^5 = 100.000$ buah (jumlah ini masih lebih besar dari 50.000). ■

Contoh 6.91

Tiga belas orang mempunyai nama depan Ali, Ahmad, dan Agus, serta nama belakang Faisal, Handoko, Hamidi, dan Rahman. Tunjukkan bahwa paling sedikit dua orang mempunyai nama depan dan nama belakang sama.

Penyelesaian:

Ada 3 nama depan dan 4 nama belakang, jadi terdapat $3 \times 4 = 12$ nama yang mungkin untuk 13 orang. Jika 13 orang dianggap sebagai merpati dan 12 nama-nama orang sebagai 12 buah sarang merpati, maka menurut prinsip sarang merpati paling sedikit ada 2 orang yang mempunyai nama depan dan nama belakang sama. ■

Soal Latihan

1. Jika suatu toko menjual 3 ukuran T-Shirt dengan 6 warna berbeda, dan setiap *T-Shirt* bisa bergambar naga, buaya, atau tidak bergambar sama sekali, berapa jenis *T-Shirt* yang dapat anda beli ?
2. Perusahaan pakaian membuat satu set pakaian yang dapat dikombinasikan, yang terdiri dari dua blus, dua pasang celana panjang, satu kemeja, dan satu *blazer*. Berapa kombinasi pakaian yang dapat dibuat dari set pakaian tersebut ? Jika pada set pakaian ditambahkan sebuah *sweater*, berapakah kombinasinya sekarang ? (Catatan: *blazer* harus dipakai di atas blus atau kemeja, atau tidak dipakai sama sekali. Tetapi *sweater* bisa langsung dipakai tanpa blus atau kemeja.)
3. Berapakah jumlah kata (terdiri dari 8 huruf) yang dapat dibentuk dari 26 huruf, tanpa memperhitungkan arti kata yang terbentuk. Buatlah untuk dua kemungkinan (boleh mengulang huruf atau tidak boleh mengulang huruf).
4. Enam orang melamar pekerjaan untuk 3 pekerjaan yang sama, yang masing-masing akan ditempatkan di Jakarta, Bogor, dan Bandung. Berapakah kemungkinan susunan orang yang diterima untuk menempati posisi tersebut ?
5. Suatu peubah (*variable*) di dalam bahasa pemrograman harus berupa sebuah huruf atau sebuah huruf diikuti dengan sebuah angka. Berapa banyak nama peubah yang dapat dibuat (tinjau dua buah kasus: huruf kapital dan huruf kecil dibedakan, dan huruf kapital dan huruf kecil tidak dibedakan)?
6. Dalam berapa banyak cara huruf-huruf *a, b, c, d, e, f* dapat disusun jika huruf *b* harus di sebelah kiri dan bersebelahan dengan huruf *e*?
7. (a) Misalkan pengulangan tidak dibolehkan. Berapa banyak bilangan empat-angka dapat dibentuk dari angka-angka 1, 2, 3, 5, 7, 8?
(b) Berapa banyak bilangan dalam (a) yang genap?
(c) Berapa banyak bilangan dalam (a) yang lebih kecil dari 4000?
8. Di dalam sebuah kelas terdapat 100 mahasiswa, 40 orang diantaranya laki-laki.
(a) Berapa banyak cara dapat dibentuk sebuah panitia 10-orang?
(b) Ulangi pertanyaan (a) jika banyaknya laki-laki harus sama dengan banyaknya perempuan.
(c) Ulangi pertanyaan (a) jika panitia itu harus terdiri dari enam laki-laki dan empat perempuan *atau* empat laki-laki dan enam perempuan?
9. Carilah jumlah himpunan bagian dari $A = \{a, b, c, d, e\}$? (Gunakan rumus kombinasi)

10. Berapakah jumlah himpunan bagian dari himpunan $B = \{1, 2, \dots, 10\}$ yang mempunyai anggota paling sedikit enam?
11. Seseorang mempunyai 10 kawan. Dalam berapa banyak cara ia dapat pergi makan ke restoran dengan dua atau lebih kawannya?
12. Berapa banyak permutasi bilangan yang dibentuk dari $\{1, 2, 3, 4, 5, 6, 7, 8\}$?
13. Berapa banyak *string* 5-huruf yang dapat dibentuk dari huruf-huruf $a, b, c, d,$ dan e sedemikian sehingga a tidak boleh diikuti langsung oleh b ?
14. Berapa banyak bilangan bulat positif empat-angka antara 1000 dan 9999 (termasuk 1000 dan 9999) yang habis dibagi 5 dan 7?
15. Berapa banyak jumlah bilangan bulat 5-angka yang mengandung tepat 1 buah angka 6? (Catatan: angka pertama tidak boleh 0)
16. Sebuah kelompok terdiri dari 7 orang wanita dan 4 orang pria. Berapa banyak perwakilan 4-orang yang dapat dibentuk dari kelompok itu jika paling sedikit harus ada 2 orang wanita di dalamnya?
17. Sebuah klub penggemar mobil VW terdiri atas 8 pria dan 6 wanita. Terdapat 1 pasang suami istri di antara anggota klub tersebut. Berapa banyak cara membentuk sebuah panitia yang terdiri atas 3 pria dan 3 wanita sedemikian sehingga memasukkan salah satu dari suami atau istri itu, tetapi tidak keduanya?
18. Sebuah klub mobil antik beranggotakan 6 orang pria dan 5 orang wanita. Mereka akan membentuk panitia yang terdiri dari lima orang. Berapa banyak jumlah panitia yang dapat dibentuk jika panitiannya terdiri dari paling sedikit satu pria dan satu wanita.
19. Tentukan banyaknya “kata” yang terbentuk dari huruf-huruf dalam kata “SELEBES” jika (masing-masing soal):
 - (c) setiap “kata” berawal dengan huruf E dan berakhir dengan E,
 - (b) pada setiap “kata”, tiga huruf E berdampingan satu sama lain.
20. Berapa banyak solusi bilangan bulat tak-negatif dari ketidaksamaan
$$x_1 + x_2 + x_3 \leq 10?$$
21. Dalam berapa banyak cara 22 buku yang berbeda dapat diberikan kepada 5 mahasiswa sehingga 2 diantara mereka memperoleh 5 buku dan 3 lainnya memperoleh 4 buku.
22. Dari sejumlah besar koin 25-an, 50-an, 100-an, dan 500-an, dalam berapa banyak cara lima oin diambil?

23. Berapa banyak bilangan 8-angka yang dapat dibentuk dari angka-angka 6, 3, 3, 5, 1, 1, 1, 5?
24. Palindrom adalah barisan karakter (huruf atau angka) yang bila dibaca dari depan atau dari belakang adalah sama. Contoh: KATAK, MALAM, 21477412, 36963. Untuk soal ini kita hanya meninjau palindrom yang dibentuk dari barisan angka. Berapa banyak bilangan palindrom 9-angka yang dapat dibentuk dari angka 0, 1, ..., 9 dengan ketentuan tidak boleh ada pengulangan angka pada setengah bagian (misalnya, 366191663 tidak dibenarkan karena 6 dipakai 2 kali)?
25. Berapa banyak bilangan bulat yang panjangnya 20 angka yang mengandung dua buah angka 0, empat buah angka 1, tiga buah angka 2, satu buah angka 3, dua buah angka 4, tiga buah angka 5, dua buah angka 7, dan tiga buah angka 9?
26. Dari sejumlah besar CD (*compact disc*) di dalam kotak yang berisi program-program aplikasi A, B, C, D , dan E , berapa banyak cara 10 CD dapat diambil?
27. Perlihatkan bahwa $C(n, k) = C(n, n-k)$.
28. Sebuah pesan kawat dibentuk dari rangkaian lima garis putus-putus (*dash*) dan tiga buah titik (*dot*). Berapa banyak pesan yang dapat dibentuk?
29. Lima belas pemain basket akan direkrut oleh tiga tim profesional di Bandung, Jakarta, dan Surabaya, sedemikian sehingga setiap tim akan merekrut lima pemain. Dalam berapa banyak cara ini dapat dilakukan?
30. Sebuah kardus berisi banyak bola berwarna merah, biru, dan ungu. Akan diambil 10 buah bola saja.
- Berapa banyak cara mengambil bola jika bola merah paling sedikit 5.
 - Berapa banyak cara mengambil bola jika bola merah paling banyak 5.
31. Berapa banyak cara $2n$ orang dapat dibagi menjadi n pasangan?
32. Jabarkan bentuk perpangkatan $(3x - 2y)^4$.
33. Tentukan suku keempat dari $(5 - 4x)^6$.
34. Buktikan bahwa $\sum_{k=0}^n (-1)^k C(n, k) = 0$.
35. Buktikan dengan induksi matematik teorema Binomial

$$(x + y)^n = \sum_{k=0}^n C(n, k)x^{n-k}y^k$$

36. Tunjukkan bahwa sembarang 6 kelas kuliah pasti terdapat dua kelas yang dijadwalkan pada hari yang sama, dengan asumsi tidak ada kuliah pada Hari Sabtu (akhir pekan).
37. Tunjukkan bahwa di antara kelompok dengan 5 buah bilangan bulat, terdapat dua buah bilangan dengan sisa yang sama jika dibagi dengan 4.
38. Sebuah kotak bola *bowling* berisi 10 bola berwarna merah dan 10 bola berwarna biru. Seorang pemain memilih bola secara acak tanpa melihat ke dalam kotak.
- Berapa banyak bola yang harus diambil untuk memastikan paling sedikit tiga bola berwarna sama?
 - Berapa banyak bola yang harus diambil untuk memastikan paling sedikit tiga bola berwarna biru?
39. Berapa peluang sebuah bilangan bulat yang dipilih secara acak dari 100 bilangan bulat positif pertama bernilai genap?
40. Berapa peluang dari 5 buah kartu remi yang dibagi tidak mengandung ratu satu buah pun?
41. Berapa peluang bahwa sebuah bilangan bulat yang tidak melebihi 100 habis dibagi 3?
42. Sebuah dadu dan sebuah koin uang logam dilempar bersamaan. Berapa peluang angka yang muncul adalah 3 dan muka koin yang muncul adalah gambar?
43. Tujuh kecelakaan mobil terjadi dalam seminggu. Berapa peluang bahwa semuanya terjadi pada hari yang sama?
44. Ada sepuluh pasang sepatu di dalam lemari. Jika delapan sepatu diambil secara acak, berapa peluang tidak ada sepasang sepatu yang terambil?
45. Sepuluh orang masuk lift pada lantai dasar sebuah gedung bertingkat 20. Berapa peluang mereka semua ke luar pada lantai/tingkat yang berbeda?

Lebih banyak lagi yang terdapat di langit dan di bumi, Horatio,
daripada yang engkau mimpiakan di dalam filsafatmu.
(Shakespeare dalam lakon "Hamlet")

BAB 7

Aljabar Boolean

Kita dapat menjadi berpengetahuan dengan pengetahuan orang lain,
tetapi kita tidak dapat menjadi bijaksana dengan
menggunakan kearifan orang lain.
(Micahel de Montaigne)

Aljabar Boolean, sebagai salah satu cabang matematika, pertama kali dikemukakan seorang matematikawan Inggris, George Boole, pada Tahun 1854. Boole melihat bahwa himpunan dan logika proposisi mempunyai sifat-sifat yang serupa (bandingkan Tabel 1.6 dengan Tabel 2.1). Dalam buku *The Laws of Thought*, Boole memaparkan aturan-aturan dasar logika (yang kemudian dikenal sebagai logika Boolean). Aturan dasar logika ini membentuk struktur matematika yang disebut **aljabar Boolean**. Pada tahun 1938, Claude Shannon memperlihatkan penggunaan aljabar Boolean untuk merancang rangkaian sirkuit yang menerima masukan 0 dan 1 dan menghasilkan keluaran juga 0 dan 1. Aljabar Boolean telah menjadi dasar teknologi komputer digital karena rangkaian elektronik di dalam komputer juga bekerja dengan mode operasi bit, 0 dan 1. Saat ini aljabar Boolean digunakan secara luas dalam perancangan rangkaian pensaklaran, rangkaian digital, dan rangkaian *IC* (*integrated circuit*) komputer. Kita akan memulai Bab 7 ini dengan konsep aljabar Boolean, penyederhanaan fungsi Boolean, dan aplikasinya dalam perancangan rangkaian logika.

7.1 Definisi Aljabar Boolean

Aljabar Boolean dapat didefinisikan secara abstrak dalam beberapa cara. Cara yang paling umum adalah dengan menspesifikasikan unsur-unsur pembentuknya dan operasi-operasi yang menyertainya, seperti pada Definisi 7.1 berikut.

DEFINISI 7.1. [LIP92] Misalkan B adalah himpunan yang didefinisikan pada dua operator biner, $+$ dan \cdot , dan sebuah operator uner, $'$. Misalkan 0 dan 1 adalah dua elemen yang berbeda dari B . Maka, tupel

$$\langle B, +, \cdot, ', 0, 1 \rangle$$

disebut **aljabar Boolean** jika untuk setiap $a, b, c \in B$ berlaku aksioma (sering dinamakan juga Postulat Huntington) berikut:

1. Identitas

- (i) $a + 0 = a$
- (ii) $a \cdot 1 = a$
- ⋮

2. Komutatif

- (i) $a + b = b + a$
- (ii) $a \cdot b = b \cdot a$

3. Distributif

- (i) $a \cdot (b + c) = (a \cdot b) + (a \cdot c)$
- (ii) $a + (b \cdot c) = (a + b) \cdot (a + c)$

4. Komplemen¹

Untuk setiap $a \in B$ terdapat elemen unik $a' \in B$ sehingga

- (i) $a + a' = 1$
- (ii) $a \cdot a' = 0$

Elemen 0 dan 1 adalah dua elemen unik yang di dalam B . 0 disebut elemen terkecil dan 1 disebut elemen terbesar. Kedua elemen unik dapat berbeda-beda pada beberapa aljabar Boolean (misalnya \emptyset dan U pada himpunan, F dan T pada proposisi), namun secara umum kita tetap menggunakan 0 dan 1 sebagai dua buah elemen unik yang berbeda. Elemen 0 disebut elemen *zero*, sedangkan elemen 1 disebut elemen *unit*. Operator $+$ disebut operator penjumlahan, \cdot disebut operator perkalian, dan $'$ disebut operator komplemen.

Terdapat perbedaan antara aljabar Boolean dengan aljabar biasa untuk aritmetika bilangan riil:

¹ Di dalam Bab Aljabar Boolean ini, kita menggunakan notasi tanda petik tunggal ($'$) untuk menyatakan komplemen. Sebagian buku menggunakan notasi garis atas ($\bar{}$) untuk menyatakan komplemen.

1. Hukum distributif yang pertama, $a \cdot (b + c) = (a \cdot b) + (a \cdot c)$, sudah dikenal di dalam aljabar biasa, tetapi hukum distributif yang kedua, $a + (b \cdot c) = (a + b) \cdot (a + c)$, benar untuk aljabar Boolean, tetapi tidak benar untuk aljabar biasa.
2. Aljabar Boolean tidak memiliki kebalikan perkalian (*multiplicative inverse*) dan kebalikan penjumlahan; karena itu, tidak ada operasi pembagian dan pengurangan di dalam aljabar Boolean.
3. Aksioma nomor 4 mendefinisikan operator yang dinamakan *komplemen* yang tidak tersedia pada aljabar biasa.
4. Aljabar biasa memperlakukan himpunan bilangan riil dengan elemen yang tidak berhingga banyaknya. Sedangkan aljabar Boolean memperlakukan himpunan elemen B yang sampai sekarang belum didefinisikan, tetapi pada aljabar Boolean dua-nilai (yang dijelaskan di upabab 7.2), B didefinisikan sebagai himpunan dengan hanya dua nilai, 0 dan 1.

Hal lain yang penting adalah membedakan elemen himpunan dan peubah (*variable*) pada sistem aljabar. Sebagai contoh, pada aljabar biasa, elemen himpunan bilangan riil adalah angka, sedangkan peubahnya seperti a , b , c , dan sebagainya. Dengan cara yang sama pada aljabar Boolean, orang mendefinisikan elemen-elemen himpunan dan peubah seperti x , y , z sebagai simbol-simbol yang merepresentasikan elemen.

Berhubung elemen-elemen B tidak didefinisikan nilainya (kita bebas menentukan anggota-anggota B), maka untuk mempunyai sebuah aljabar Boolean, orang harus memperlihatkan [MAN82]:

1. elemen-elemen himpunan B ,
2. kaidah/aturan operasi untuk dua operator biner dan operator uner,
3. himpunan B , bersama-sama dengan dua operator tersebut, memenuhi keempat aksioma di atas.

Jika ketiga persyaratan di atas dipenuhi, maka aljabar yang didefinisikan dapat dikatakan sebagai aljabar Boolean.

Contoh 7.1

[LIP92] Misalkan $B = \{1, 2, 5, 7, 10, 14, 35, 70\}$ adalah pembagi dari 70. Tunjukkan cara membentuk B menjadi sebuah aljabar Boolean.

Penyelesaian:

Elemen-elemen himpunan B sudah didefinisikan. Sekarang kita tentukan kaidah operasi untuk operator $+$, \cdot , dan $'$. Misalkan kita definisikan

$$\begin{aligned} a + b &= \text{KPK}(a, b) = \text{kelipatan persekutuan terkecil} \\ a \cdot b &= \text{PBB}(a, b) = \text{pembagi bersama terbesar} \\ a' &= 70/a \end{aligned}$$

Maka sekarang kita tunjukkan apakah B bersama-sama dengan kedua operator biner dan operator uner memenuhi empat aksioma yang disebutkan di dalam Definisi 7.1.

1. Identitas: 1 adalah elemen identitas untuk operasi penjumlahan (1 sebagai elemen *zero*) dan 70 adalah elemen identitas untuk operasi perkalian (70 sebagai elemen *unit*), karena
 - (i) $a + 1 = \text{KPK}(a, 1) = a$
 - (ii) $a \cdot 70 = \text{PBB}(a, 70) = a$
2. Komutatif: jelas berlaku karena
 - (i) $a + b = b + a = \text{KPK}(a, b)$
 - (ii) $a \cdot b = b \cdot a = \text{PBB}(a, b)$
3. Distributif: jelas berlaku karena (ditunjukkan dengan contoh)
 - (i) $10 \cdot (5 + 7) = \text{PBB}(10, \text{KPK}(5, 7)) = \text{PBB}(10, 35) = 5$
 $(10 \cdot 5) + (10 \cdot 7) = \text{KPK}(\text{PBB}(10, 5), \text{PBB}(10, 7)) = \text{KPK}(5, 1) = 5$
 - (ii) $10 + (5 \cdot 7) = \text{KPK}(10, \text{PBB}(5, 7)) = \text{KPK}(10, 1) = 10$
 $(10 + 5) \cdot (10 + 7) = \text{PBB}(\text{KPK}(10, 5), \text{KPK}(10, 7)) = \text{PBB}(10, 70) = 10$
4. Komplemen: jelas berlaku karena
 - (i) $a + a' = \text{KPK}(a, 70/a) = 70$
 - (ii) $a \cdot a' = \text{PBB}(a, 70/a) = 1$

Oleh karena semua aksioma di dalam Definisi 7.1 dipenuhi, maka $B = \{1, 2, 5, 7, 10, 14, 35, 70\}$ yang didefinisikan pada operator biner $+$ dan \cdot dan operator komplemen ‘ adalah sebuah aljabar Boolean. Dengan kata lain, $\langle B, +, \cdot, ', 1, 70 \rangle$ adalah sebuah aljabar Boolean.

Definisi 7.1 mengingatkan kita pada aljabar himpunan (Bab 3) dan aljabar proposisi (Bab 1), karena terdapat kemiripan sifat antara kedua aljabar tersebut dengan aljabar Boolean. Hal ini tidak mengherankan karena aljabar himpunan dan aljabar proposisi sebenarnya juga merupakan aljabar Boolean. Dengan kata lain, aljabar himpunan dan aljabar proposisi adalah himpunan bagian (*subset*) dari aljabar Boolean.

Pada aljabar himpunan, B beranggotakan semua himpunan bagian dari himpunan semesta U . Dua elemen unik berbeda dari B adalah \emptyset dan U , dua operator binernya adalah \cup dan \cap , operator unernya adalah komplemen ($\bar{}$), dan semua aksioma yang terdapat pada Definisi 7.1 dipenuhi oleh semua anggota B . Dengan kata lain $\langle B, \cup, \cap, \bar{}, \emptyset, U \rangle$ adalah aljabar Boolean.

Sedangkan pada aljabar proposisi, B beranggotakan semua proposisi dengan n peubah. Dua elemen unik berbeda dari B adalah T dan F , operator binernya adalah \vee dan \wedge , operator unernya adalah \sim , dan semua aksioma yang terdapat pada Definisi 7.1 dipenuhi oleh semua anggota B . Dengan kata lain $\langle B, \vee, \wedge, \sim, F, T \rangle$ adalah aljabar Boolean.

7.2 Aljabar Boolean Dua-Nilai

Mengingat B tidak ditentukan anggota-anggotanya, maka kita dapat membentuk sejumlah tidak berhingga aljabar Boolean. Pada aljabar Boolean berhingga, banyaknya anggota B terbatas, tetapi paling sedikit beranggotakan dua buah elemen karena –menurut Definisi 7.1– di dalam B harus terdapat dua elemen yang berbeda.

Aljabar Boolean yang terkenal dan memiliki terapan yang luas adalah aljabar Boolean dua-nilai (*two-valued Boolean algebra*). Aljabar Boolean dua-nilai didefinisikan pada sebuah himpunan B dengan dua buah elemen 0 dan 1 (sering dinamakan *bit* – singkatan dari *binary digit*), yaitu $B = \{0, 1\}$, operator biner, $+$ dan \cdot , operator uner, $'$. Kaidah untuk operator biner dan operator uner ditunjukkan pada Tabel 7.1, 7.2, dan 7.3 di bawah ini.

Tabel 7.1

a	b	$a \cdot b$
0	0	0
0	1	0
1	0	0
1	1	1

Tabel 7.2

a	b	$a + b$
0	0	0
0	1	1
1	0	1
1	1	1

Tabel 7.3

a	a'
0	1
1	0

Kita harus memperlihatkan bahwa keempat aksioma di dalam Definisi 7.1 terpenuhi pada himpunan $B = \{0, 1\}$ dengan dua operator biner dan satu operator uner yang didefinisikan di atas:

1. Identitas: jelas berlaku karena dari tabel dapat kita lihat bahwa:

- (i) $0 + 1 = 1 + 0 = 1$
- (ii) $0 \cdot 1 = 0 \cdot 1 = 0$

yang memenuhi elemen identitas 0 dan 1 seperti yang didefinisikan pada postulat Huntington.

2. Komutatif: jelas berlaku dengan melihat simetri tabel operator biner.

3. Distributif:

- (i) $a \cdot (b + c) = (a \cdot b) + (a \cdot c)$ dapat ditunjukkan benar dari tabel operator biner di atas dengan membentuk tabel kebenaran untuk semua nilai yang mungkin dari a , b , dan c (Tabel 7.4). Oleh karena nilai-nilai pada kolom $a \cdot (b + c)$ sama dengan nilai-nilai pada kolom $(a \cdot b) + (a \cdot c)$, maka kesamaan $a \cdot (b + c) = (a \cdot b) + (a \cdot c)$ adalah benar.
- (ii) Hukum distributif $a + (b \cdot c) = (a + b) \cdot (a + c)$ dapat ditunjukkan benar dengan membuat tabel kebenaran dengan cara yang sama seperti (i).

Tabel 7.4

a	b	c	$b + c$	$a \cdot (b + c)$	$a \cdot b$	$a \cdot c$	$(a \cdot b) + (a \cdot c)$
0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	1	0	0	0	0
0	1	1	1	0	0	0	0
1	0	0	0	0	0	0	0
1	0	1	1	1	0	1	1
1	1	0	1	1	1	0	1
1	1	1	1	1	1	1	1

4. Komplemen: jelas berlaku karena Tabel 7.4 memperlihatkan bahwa:

- (i) $a + a' = 1$, karena $0 + 0' = 0 + 1 = 1$ dan $1 + 1' = 1 + 0 = 1$
- (ii) $a \cdot a' = 0$, karena $0 \cdot 0' = 0 \cdot 1 = 0$ dan $1 \cdot 1' = 1 \cdot 0 = 0$

Karena keempat aksioma terpenuhi dipenuhi, maka terbukti bahwa $B = \{0, 1\}$ bersama-sama dengan operator biner $+$ dan \cdot operator komplemen $'$ merupakan aljabar Boolean.

Aljabar Boolean dua-nilai mempunyai terapan yang sangat luas dalam bidang elektronika, khususnya pada perancangan sirkuit di dalam komputer. Beberapa terapan lainnya juga ditemukan di bidang teknik sipil, teknik mesin, dan sebagainya.

Untuk selanjutnya, jika disebut aljabar Boolean, maka aljabar Boolean yang dimaksudkan di sini adalah aljabar Boolean dua-nilai.

7.3 Ekspresi Boolean

Pada aljabar Boolean dua-nilai, $B = \{0, 1\}$. Kedua elemen B ini seringkali disebut elemen biner atau *bit* (singkatan *binary bit*). Peubah (*variable*) x disebut **peubah Boolean** atau peubah biner jika nilainya hanya dari B . Ekspresi Boolean dibentuk dari elemen-elemen B dan/atau peubah-peubah yang dapat dikombinasikan satu sama lain dengan operator $+$, \cdot , dan $'$. Secara formal, ekspresi Boolean didefinisikan secara rekursif oleh Definisi 7.2 berikut [LIU85].

DEFINISI 7.2. Misalkan $(B, +, \cdot, ', 0, 1)$ adalah sebuah aljabar Boolean. Suatu ekspresi Boolean dalam $(B, +, \cdot, ',)$ adalah:

- (i) Setiap elemen di dalam B ,
- (ii) setiap peubah;
- (iii) jika e_1 dan e_2 adalah ekspresi Boolean, maka $e_1 + e_2$, $e_1 \cdot e_2$, e_1' adalah ekspresi Boolean

Jadi, menurut Definisi 7.2 di atas, setiap ekspresi di bawah ini,

- 0
- 1
- a
- b
- c
- $a + b$
- $a \cdot b$
- $a' \cdot (b + c)$
- $a' \cdot b' + a \cdot b \cdot c' + b'$, dan sebagainya

adalah ekspresi Boolean. Ekspresi Boolean yang mengandung n peubah dinamakan ekspresi Boolean bagi n peubah.

Mengevaluasi ekspresi Boolean artinya memberikan nilai pada peubah-peubah di dalam ekspresi tersebut dengan elemen-elemen di dalam B . Hasil dari evaluasi adalah sebuah nilai yang merupakan salah satu dari anggota B . Pada aljabar Boolean dua-nilai, $B = \{0, 1\}$ sehingga peubah-peubah di dalam ekspresi Boolean dapat diberi nilai 0 atau 1, begitu juga hasil evaluasi ekspresi Boolean adalah 0 atau 1. Sebagai contoh, pada ekspresi Boolean dengan 3 peubah berikut:

$$a' \cdot (b + c)$$

jika $a = 0$, $b = 1$, dan $c = 0$, maka hasil evaluasi ekspresi tersebut adalah

$$0' \cdot (1 + 0) = 1 \cdot 1 = 1$$

Dua ekspresi Boolean dikatakan **ekivalen** jika keduanya mempunyai nilai yang sama untuk setiap pemberian nilai-nilai kepada n peubah. Misalnya, ekspresi pada aksioma distributif yang pertama,

$$a \cdot (b + c)$$

ekivalen dengan

$$(a \cdot b) + (a \cdot c)$$

sehingga kita dapat menuliskan kedua ekspresi tersebut sebagai kesamaan (*identity*):

$$a \cdot (b + c) = (a \cdot b) + (a \cdot c)$$

Kekivalenan dua buah ekspresi dapat ditunjukkan dengan tabel kebenaran. Jika semua nilai ekspresi ruas kiri untuk semua kemungkinan nilai-nilai peubah sama dengan nilai ekspresi pada ruas kanan kesamaan, maka kedua ekspresi tersebut dikatakan ekivalen. Untuk aksioma distributif, hal ini sudah ditunjukkan pada Tabel 7.4.

Contoh 7.2

Perlihatkan bahwa $a + a'b = a + b$.

Penyelesaian:

Tabel kebenaran untuk kesamaan $a + a'b = a + b$ ditunjukkan pada Tabel 7.5. Karena nilai-nilai pada kolom $a + a'b$ sama dengan nilai-nilai pada kolom $a + b$, maka $a + a'b = a + b$. ■

Tabel 7.5

a	b	a'	$a'b$	$a + a'b$	$a + b$
0	0	1	0	0	0
0	1	1	1	1	1
1	0	0	0	1	1
1	1	0	0	1	1

Dalam penulisan ekspresi Boolean selanjutnya, kita menggunakan perjanjian berikut: selain tanda kurung, operator ' \cdot ' mempunyai prioritas lebih tinggi daripada operator $+$ dan \cdot . Jadi, sebagai contoh,

$a + b \cdot c$ berarti $a + (b \cdot c)$, bukan $(a + b) \cdot c$

$a \cdot b'$ berarti $a \cdot (b')$, bukan $(a \cdot b)'$

Untuk menyederhanakan penulisan, kita boleh tidak menuliskan notasi \cdot pada operasi perkalian. Jadi, kita lebih sering menuliskan $a \cdot b$ sebagai ab saja. Namun, kadang-kadang kita merasa perlu menuliskan notasi \cdot untuk maksud menekankan operasi perkalian, misalnya kita menuliskan $a \cdot 0$ ketimbang $a0$. Dengan cara penyederhanaan ini, maka aksioma distributif pada nomor 4 ditulis sebagai berikut:

- (i) $a(b + c) = ab + ac$
- (ii) $a + bc = (a + b)(a + c)$

Dengan cara seperti di atas, tentunya bila $a + b \cdot c$ ditulis sebagai $a + bc$ maka pengertian prioritas operator menjadi lebih jelas.

7.4 Prinsip Dualitas

Di dalam aljabar Boolean banyak ditemukan kesamaan (*identity*) yang dapat diperoleh dari kesamaan lainnya, misalnya pada dua aksioma distributif yang sudah disebutkan di dalam Definisi 7.1:

- (i) $a(b + c) = ab + ac$
- (ii) $a + bc = (a + b)(a + c)$

Aksioma yang kedua diperoleh dari aksioma pertama dengan cara mengganti · dengan + dan mengganti + dengan ·. Prinsip ini dikenal dengan **prinsip dualitas**, prinsip yang juga kita temukan di dalam teori himpunan maupun logika. Definisi prinsip dualitas di dalam aljabar Boolean adalah sebagai berikut ini:

DEFINISI 7.3. Misalkan S adalah kesamaan (*identity*) di dalam aljabar Boolean yang melibatkan operator +, ·, dan komplemen, maka jika pernyataan S^* diperoleh dari S dengan cara mengganti

- dengan +
- + dengan ·
- 0 dengan 1
- 1 dengan 0

dan membiarkan operator komplemen tetap apa adanya, maka kesamaan S^* juga benar. S^* disebut sebagai *dual* dari S . ■

Contoh 7.3

Tentukan dual dari

- (i) $a + 0 = a$
- (ii) $(a \cdot 1)(0 + a') = 0$
- (iii) $a(a' + b) = ab$
- (iv) $(a + b)(b + c) = ac + b$
- (v) $(a + 1)(a + 0) = a$

Penyelesaian:

- (i) $a \cdot 1 = a$
 - (ii) $(a + 0) + (1 \cdot a') = 1$
 - (iii) $a + a'b = a + b$
 - (iv) $ab + bc = (a + c)b$
 - (v) $(a \cdot 0) + (a \cdot 1) = a$
-

7.5 Hukum-hukum Aljabar Boolean

Ada banyak hukum di dalam aljabar Boolean. Beberapa literatur bervariasi dalam mengungkapkan jumlah hukum pada aljabar Boolean, tetapi hukum-hukum yang paling penting ditampilkan di dalam Tabel 7.5. Catatlah bahwa beberapa dari hukum yang terdapat di dalam tabel tersebut sudah disebutkan di dalam Definisi 7.1. Perhatikan juga bahwa terdapat kemiripan antara hukum-hukum aljabar

Boolean dengan hukum-hukum aljabar himpunan dan hukum-hukum aljabar proposisi (hal ini sudah kita jelaskan sebelumnya bahwa baik aljabar himpunan maupun aljabar proposisi keduanya juga merupakan aljabar Boolean).

Tabel 7.5 Hukum-hukum Aljabar Boolean

1. Hukum identitas: (i) $a + 0 = a$ (ii) $a \cdot 1 = a$	2. Hukum idempoten: (i) $a + a = a$ (ii) $a \cdot a = a$
3. Hukum komplementen: (i) $a + a' = 1$ (ii) $aa' = 0$	4. Hukum dominansi: (i) $a \cdot 0 = 0$ (ii) $a + 1 = 1$
5. Hukum involusi: (i) $(a')' = a$	6. Hukum penyerapan: (i) $a + ab = a$ (ii) $a(a + b) = a$
7. Hukum komutatif: (i) $a + b = b + a$ (ii) $ab = ba$	8. Hukum asosiatif: (i) $a + (b + c) = (a + b) + c$ (ii) $a(bc) = (ab)c$
9. Hukum distributif: (i) $a + (bc) = (a + b)(a + c)$ (ii) $a(bc) = ab + ac$	10. Hukum De Morgan: (i) $(a + b)' = a'b'$ (ii) $(ab)' = a' + b'$
11. Hukum 0/1 (i) $0' = 1$ (ii) $1' = 0$	

Kita dapat memperoleh hukum-hukum aljabar Boolean dari hukum-hukum aljabar Himpunan pada Tabel 2.1 (atau dari hukum-hukum aljabar proposisi pada Tabel 1.6) dengan cara mempertukarkan

$$\begin{array}{ll}
 \cup \text{ dengan } +, & \text{atau} \quad \vee \text{ dengan } + \\
 \cap \text{ dengan } ., & \text{atau} \quad \wedge \text{ dengan } . \\
 U \text{ dengan } 1, & \text{atau} \quad T \text{ dengan } 1 \\
 \emptyset \text{ dengan } 0, & \text{atau} \quad F \text{ dengan } 0.
 \end{array}$$

Perhatikanlah bahwa hukum yang ke-(ii) dari setiap hukum di atas merupakan dual dari hukum yang ke-(i). Sebagai contoh,

Hukum komutatif:
dualnya: $a + b = b + a$
 $ab = ba$

Hukum asosiatif:
dualnya: $a + (b + c) = (a + b) + c$
 $a(bc) = (ab)c$

Hukum distributif:
dualnya: $a(b + c) = ab + ac$
 $a + bc = (a + b)(a + c)$

Beberapa hukum aljabar Boolean di atas akan dibuktikan di bawah ini, sedangkan bukti untuk hukum lainnya diserahkan kepada anda sebagai latihan.

Bukti:

$$\begin{aligned} (2i) \quad a + a &= (a + a) (1) && (\text{Hukum Identitas}) \\ &= (a + a) (a + a') && (\text{Hukum Komplemen}) \\ &= a + aa' && (\text{Hukum Distributif}) \\ &= a + 0 && (\text{Hukum Komplemen}) \\ &= a && (\text{Hukum Identitas}) \end{aligned}$$

$$\begin{aligned} (2ii) \quad aa &= aa + 0 && (\text{Hukum Identitas}) \\ &= a a + a a' && (\text{Hukum Komplemen}) \\ &= a (a + a') && (\text{Hukum Distributif}) \\ &= a \cdot 1 && (\text{Hukum Komplemen}) \\ &= a && (\text{Hukum Identitas}) \end{aligned}$$

(2ii adalah dual dari 2i)

$$\begin{aligned} (4i) \quad a + 1 &= a + (a + a') && (\text{Hukum Komplemen}) \\ &= (a + a) + a' && (\text{Hukum Asosiatif}) \\ &= a + a' && (\text{Hukum Idempoten}) \\ &= 1 && (\text{Hukum Komplemen}) \end{aligned}$$

$$\begin{aligned} (4ii) \quad a \cdot 0 &= a(aa') && (\text{Hukum Komplemen}) \\ &= (aa) a' && (\text{Hukum Asosiatif}) \\ &= aa' && (\text{Hukum Idempoten}) \\ &= 0 && (\text{Hukum Komplemen}) \end{aligned}$$

(4ii adalah dual dari 4i)

$$\begin{aligned} (6i) \quad a + ab &= a \cdot 1 + a \cdot b && (\text{Hukum Identitas}) \\ &= a (1 + b) && (\text{Hukum Distributif}) \\ &= a \cdot 1 && (\text{Hukum Dominansi}) \\ &= a && (\text{Hukum Identitas}) \end{aligned}$$

$$\begin{aligned} (6ii) \quad a(a + b) &= (a + 0)(a + b) && (\text{Hukum Identitas}) \\ &= a + (0 \cdot b) && (\text{Hukum Distributif}) \end{aligned}$$

$$= a + 0 \quad (\text{Hukum Dominansi})$$

$$= a \quad (\text{Hukum Idenntitas})$$

(6ii) adalah dual dari (6i)

$$(10i) (ab)' = a' + b'$$

diketahui: $(ab)(ab)' = 0$

perlihatkan: $(ab)(a' + b') = 0$

Bukti: $(ab)(a' + b') = ab a' + abb' = 0 \cdot b + a \cdot 0 = 0 + 0 = 0$

(Hukum Distributif)
(Hukum Komplemen)
(Hukum Dominansi)
(Hukum Identitas) ■

$$(10ii) (a + b)' = a' \cdot b'$$

diketahui: $(a + b) + (a + b)' = 1$

perlihatkan: $(a + b) + (a'b') = 1$

Bukti: $(a + b) + (a'b') = (a + b + a')(a + b + b') = (1 + b) + (a + 1) = (1 + 1) = 1$

(Distributif)
(Komplemen)
(Dominansi)
(Identitas) ■

(10ii adalah dual dari 4a)

Contoh 7.3

Buktikan bahwa untuk sembarang elemen a dan b dari aljabar Boolean maka kesamaaan berikut

$$a + a'b = a + b \quad \text{dan} \quad a(a' + b) = ab$$

adalah benar.

Penyelesaian:

$$\begin{aligned} (i) \quad a + a'b &= (a + ab) + a'b && (\text{Hukum Penyerapan}) \\ &= a + (ab + a'b) && (\text{Hukum Asosiatif}) \\ &= a + (a + a')b && (\text{Hukum Distributif}) \\ &= a + 1 \cdot b && (\text{Hukum Komplemen}) \\ &= a + b && (\text{Hukum Identitas}) \end{aligned}$$

$$\begin{aligned} (ii) \quad a(a' + b) &= a a' + ab && (\text{Hukum Distributif}) \\ &= 0 + ab && (\text{Hukum Komplemen}) \\ &= ab && (\text{Hukum Identitas}) \end{aligned}$$

atau, dapat juga dibuktikan melalui dualitas dari (i) sebagai berikut:

$$\begin{aligned}
 a(a' + b) &= a(a + b)(a' + b') \\
 &= a \{ (a + b)(a' + b') \} \\
 &= a \{ (a a') + b \} \\
 &= a(0 + b) \\
 &= ab
 \end{aligned}$$

Kesamaan aljabar Boolean (termasuk hukum-hukum aljabar Boolean) juga dapat diperlihatkan benar dengan menggunakan tabel kebenaran (seperti yang sudah djelaskan pada upabab 7.3).

7.6 Fungsi Boolean

DEFINISI 7.3. **Fungsi Boolean** (disebut juga fungsi biner) adalah pemetaan dari B^n ke B melalui ekspresi Boolean, kita menuliskannya sebagai

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

yang dalam hal ini B^n adalah himpunan yang beranggotakan pasangan terurut ganda- n (*ordered n-tuple*) di dalam daerah asal B .

Misalkan ekspresi Boolean dengan n peubah adalah $E(x_1, x_2, \dots, x_n)$. Menurut Definisi 7.3 ini, setiap pemberian nilai-nilai kepada peubah x_1, x_2, \dots, x_n merupakan suatu pasangan terurut ganda- n di dalam daerah asal B^n dan nilai ekspresi tersebut adalah bayangannya di dalam daerah hasil B [LIU85]. Dengan kata lain, setiap ekspresi Boolean tidak lain merupakan fungsi Boolean. Misalkan sebuah fungsi Boolean adalah $f(x, y, z) = xyz + x'y + y'z$. Fungsi f memetakan nilai-nilai pasangan terurut ganda-3 (x, y, z) ke himpunan $\{0, 1\}$. Contoh pasangan terurut ganda-3 misalnya $(1, 0, 1)$ yang berarti $x = 1, y = 0$, dan $z = 1$ sehingga $f(1, 0, 1) = 1 \cdot 0 \cdot 1 + 1' \cdot 0 + 0' \cdot 1 = 0 + 0 + 1 = 1$.

Contoh 7.4

Contoh-contoh fungsi Boolean:

1. $f(x) = x$
2. $f(x, y) = x'y + xy' + y'$
3. $f(x, y) = x'y'$
4. $f(x, y) = (x + y)'$
5. $f(x, y, z) = xyz'$

Setiap peubah di dalam fungsi Boolean, termasuk dalam bentuk komplemennya, disebut **literal**. Fungsi $h(x, y, z) = xyz'$ pada contoh di atas terdiri dari 3 buah literal, yaitu x, y , dan z' . Fungsi tersebut berharga 1 jika $x = 1, y = 1, z = 0$, sebab

$$h(1, 1, 0) = 1 \cdot 1 \cdot 0' = (1 \cdot 1) \cdot 1 = 1 \cdot 1 = 1$$

dan berharga 0 untuk harga x, y , dan z lainnya.

Selain secara aljabar, fungsi Boolean juga dapat dinyatakan dengan tabel kebenaran dan dengan rangkaian logika (yang terakhir ini akan dijelaskan di dalam upabab 7.10). Tabel kebenaran berisi nilai-nilai fungsi untuk semua kombinasi nilai-nilai peubahnya.

Jika fungsi Boolean dinyatakan dengan tabel kebenaran, maka untuk fungsi Boolean dengan n buah peubah, kombinasi dari nilai peubah-peubahnya adalah sebanyak 2^n . Ini berarti terdapat 2^n baris yang berbeda di dalam tabel kebenaran tersebut. Misalkan $n = 3$, maka akan terdapat $2^3 = 8$ baris tabel. Cara yang praktis membuat semua kombinasi tersebut adalah sebagai berikut:

1. Untuk peubah pertama, isi 4 baris pertama pada kolom pertama dengan sebuah 0 dan 4 baris selanjutnya dengan sebuah 1 berturut-turut.
2. Untuk peubah kedua, isi 2 baris pertama pada kolom kedua dengan 0 dan 2 baris berikutnya dengan 1, 2 baris berikutnya dengan 0 lagi, dan 2 baris terakhir dengan 1.
3. Untuk peubah ketiga, isi kolom ketiga secara berselang-seling dengan 0 dan 1 mulai baris pertama sampai baris terakhir.

Contoh 7.5

Diketahui fungsi Boolean $f(x, y, z) = xy z'$, nyatakan h dalam tabel kebenaran.

Penyelesaian:

Nilai-nilai fungsi Boolean diperlihatkan pada Tabel 7.6.

Tabel 7.6

x	y	z	$f(x, y, z) = xy z'$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

Fungsi Boolean tidak selalu unik pada representasi ekspresinya. Artinya, dua buah fungsi yang ekspresi Booleannya berbeda dapat menyatakan dua buah fungsi yang sama. Dengan kata lain, dua buah fungsi sama jika kedua ekspresi Booleannya ekivalen. Misalkan f dan g adalah ekspresi dari suatu fungsi Boolean. Fungsi f dan g dikatakan merupakan fungsi yang sama jika keduanya memiliki nilai yang sama pada tabel kebenaran untuk setiap kombinasi peubah-peubahnya. Sebagai contoh, fungsi

$$f(x, y, z) = x'y'z + x'yz + xy' \text{ dan } g(x, y, z) = x'z + x'y'$$

adalah dua buah fungsi Boolean yang sama. Kesamaan ini dapat dilihat pada Tabel 7.7.

Tabel 7.7

x	y	z	$x'y'z + x'yz + xy'$	$x'z + x'y'$
0	0	0	0	0
0	0	1	1	1
0	1	0	0	0
0	1	1	1	1
1	0	0	1	1
1	0	1	1	1
1	1	0	0	0
1	1	1	0	0

Jika sebuah fungsi Boolean tidak unik dalam representasi ekspresinya, dapatkah kita menemukan ekspresi Boolean lainnya yang menspesifikasikan fungsi yang sama? Jawabnya: dapat, yaitu dengan melakukan manipulasi aljabar terhadap ekspresi Boolean. Yang dimaksud dengan memanipulasi atau menyederhanakan ekspresi Boolean adalah menggunakan hukum-hukum aljabar Boolean untuk menghasilkan bentuk yang ekivalen. Perhatikanlah bahwa

$$\begin{aligned} f(x, y, z) &= x'y'z + x'yz + xy' \\ &= x'z(y' + y) + x'y' \quad (\text{Hukum distributif}) \\ &= x'z \cdot 1 + x'y' \quad (\text{Hukum komplemen}) \\ &= x'z \cdot 1 + x'y' \quad (\text{Hukum identitas}) \end{aligned}$$

Manipulasi aljabar pada ekspresi Boolean akan dibahas lebih lanjut pada upa-bab **Penyederhanaan Fungsi Boolean**.

7.7 Penjumlahan dan Perkalian Dua Fungsi

Misalkan f dan g adalah dua buah fungsi Boolean dengan n peubah, maka penjumlahan $f + g$ didefinisikan sebagai

$$(f + g)(x_1 + x_2 + \dots + x_n) = f(x_1 + x_2 + \dots + x_n) + g(x_1 + x_2 + \dots + x_n)$$

sedangkan perkalian $f \cdot g$ didefinisikan sebagai

$$(f \cdot g)(x_1 + x_2 + \dots + x_n) = f(x_1 + x_2 + \dots + x_n) g(x_1 + x_2 + \dots + x_n)$$

Contoh 7.6

Misalkan $f(x, y) = xy' + y$ dan $g(x, y) = x' + y'$ maka

$$h(x, y) = f + g = xy' + y + x' + y'$$

yang bila disederhanakan lebih lanjut menjadi

$$h(x, y) = xy' + x' + (y + y') = xy' + x' + 1 = xy' + x'$$

dan

$$i(x, y) = f \cdot g = (xy' + y)(x' + y')$$

7.8 Komplemen Fungsi Boolean

Bila sebuah fungsi Boolean dikomplemenkan, kita memperoleh fungsi komplemen. Fungsi komplemen berguna pada saat kita melakukan penyederhanaan fungsi Boolean. Fungsi komplemen dari suatu fungsi f , yaitu f' dapat dicari dua cara berikut:

1. Cara pertama: menggunakan hukum De Morgan

Hukum De Morgan untuk dua buah peubah, x_1 dan x_2 , adalah

$$(i) (x_1 + x_2)' = x_1' x_2' \quad \text{dan dualnya: (ii)} (x_1 \cdot x_2)' = x_1' + x_2'$$

Hukum De Morgan untuk tiga buah peubah, x_1, x_2 , dan x_3 , adalah

$$\begin{aligned} (i) (x_1 + x_2 + x_3)' &= (x_1 + y)', \text{ yang dalam hal ini } y = x_2 + x_3 \\ &= x_1' y' \\ &= x_1' (x_2 + x_3)' \\ &= x_1' x_2' x_3' \end{aligned}$$

dan dualnya adalah

$$(ii) (x_1 \cdot x_2 \cdot x_3)' = x_1' + x_2' + x_3'$$

Hukum De Morgan untuk n buah peubah, x_1, x_2, \dots, x_n , adalah

$$(i) (x_1 + x_2 + \dots + x_n)' = x_1' x_2' \dots x_n'$$

dan dualnya adalah

$$(ii) (x_1 \cdot x_2 \cdot \dots \cdot x_n)' = x_1' + x_2' + \dots + x_n'$$

- Cara 2: $f(x, y, z) = x(yz + yz')$
- Dual dari ekspresi Booleannya: $x' + (y + z)(y' + z')$
- Komplementkan tiap literal dari dual: $f'(x, y, z) = x + (y + z)(y' + z')$

$$= x + (y + z)(y' + z')$$

$$= x + (yz)(y'z')$$

$$= x + (yz' + y'z)$$

$$f'(x, y, z) = (x(yz' + y'z))'$$

$$= x(yz' + y'z)$$

Pembelajaran:

Cariyah komplemen dari fungsi $f(x, y, z) = x(yz + yz')$

Contoh 7.9

- Jadi, $f'(x, y, z) = x' + (y + z)(y' + z')$
- $x' + (y + z)(y' + z') = f'$

Komplementkan tiap literal dari dual di atas menjadi

$$x + (y + z)(y' + z')$$

Misalkan $f(x, y, z) = x(yz' + yz)$, maka dual dari ekspresi Booleannya adalah

Contoh 7.8

2. Cara kedua: menggunakan prinsip dualitas.
- Tentukan dual dari ekspresi Boolean yang merepresentasikan f . Lalu komplementkan setiap literal di dalam dual tersebut. Bentuk akhir yang diperoleh menyatakan fungsi komplemen

- Misalkan $f(x, y, z) = x(yz' + yz)$, maka fungsi komplemenya adalah
- $f'(x, y, z) = (x(yz' + yz))'$
- $= x' + (y + z)(y' + z')$
- $= x' + (y'z)(yz')$
- $= x' + (y'z + yz')$

Misalkan $f(x, y, z) = x(yz' + yz)$, maka fungsi komplemenya adalah

Contoh 7.7

7.9 Bentuk Kanonik

Ekspresi Boolean yang menspesifikasikan suatu fungsi dapat disajikan dalam dua bentuk berbeda. Pertama, sebagai penjumlahan dari hasil kali dan kedua sebagai perkalian dari hasil jumlah. Misalnya,

$$f(x, y, z) = x'y'z + xy'z' + xyz$$

dan

$$g(x, y, z) = (x + y + z)(x + y' + z)(x + y' + z')(x' + y + z')(x' + y' + z)$$

adalah dua buah fungsi yang sama (dapat ditunjukkan dari tabel kebenarannya). Fungsi yang pertama, f , muncul dalam bentuk penjumlahan dari hasil kali, sedangkan fungsi yang kedua, g , muncul dalam bentuk perkalian dari hasil jumlah. Perhatikan juga bahwa setiap suku (*term*) di dalam ekspresi mengandung literal yang lengkap dalam peubah x , y , dan z , baik peubahnya tanpa komplemen maupun dengan komplemen. Ada dua macam bentuk *term*, yaitu *minterm* (hasil kali) dan *maxterm* (hasil jumlah).

Suku-suku di dalam ekspresi Boolean dengan n peubah x_1, x_2, \dots, x_n dikatakan *minterm* jika ia muncul dalam bentuk

$$\tilde{x}_1 \tilde{x}_2 \dots \tilde{x}_n$$

dan dikatakan *maxterm* jika ia muncul dalam bentuk

$$\tilde{x}_1 + \tilde{x}_2 + \dots + \tilde{x}_n$$

yang dalam hal ini \tilde{x}_i menyatakan literal x_i atau x_i' . Perhatikan bahwa sebuah *minterm* atau *maxterm* harus mengandung literal yang lengkap. Sebagai contoh, $x'y'z$, $xy'z'$, dan xyz pada fungsi f di atas adalah tiga buah *minterm*, dan $x + y + z$, $x + y' + z$, $x + y' + z'$, $x' + y + z'$, dan $x' + y' + z$ pada fungsi g di atas adalah empat buah *maxterm*; setiap *term* mengandung literal lengkap dalam peubah x , y , dan z .

Ekspresi Boolean yang dinyatakan sebagai penjumlahan dari satu atau lebih *minterm* atau perkalian dari satu atau lebih *maxterm* disebut dalam **bentuk kanonik**.

Jadi, ada dua macam bentuk kanonik:

1. Penjumlahan dari hasil kali (*sum-of-product* atau SOP)
2. Perkalian dari hasil jumlah (*product-of-sum* atau POS)

Fungsi

$$f(x, y, z) = x'y'z + xy'z' + xyz$$

dikatakan dalam bentuk SOP dan fungsi

$$g(x, y, z) = (x + y + z)(x + y' + z)(x + y' + z')(x' + y + z')(x' + y' + z)$$

dikatakan dalam bentuk POS. Nama lain untuk SOP adalah **bentuk normal disjungtif** (*disjunctive normal form*) dan nama lain POS adalah **bentuk normal konjungtif** (*conjunctive normal form*).

Cara membentuk *minterm* dan *maxterm* dari tabel kebenaran ditunjukkan pada Tabel 7.8 (dua peubah) dan 7.9 (tiga peubah). Untuk *minterm*, setiap peubah yang bernilai 0 dinyatakan dalam bentuk komplemen, sedangkan peubah yang bernilai 1 dinyatakan tanpa komplemen. Sebaliknya, untuk *maxterm*, setiap peubah yang bernilai 0 dinyatakan tanpa komplemen, sedangkan peubah yang bernilai 1 dinyatakan dalam bentuk komplemen.

Kadang-kadang *minterm* dilambangkan sebagai huruf *m* kecil berindeks. Indeks menyatakan nilai desimal dari *string* biner yang merepresentasikan *term*. Misalnya pada *term* dengan 2 peubah *x* dan *y*, indeks 0 pada m_0 menyatakan nilai desimal dari 00 ($x = 0$ dan $y = 0$), indeks 1 pada m_1 menyatakan nilai desimal dari 01 ($x = 0$ dan $y = 1$), dan seterusnya. Jadi, untuk *minterm* dari 3 peubah (*x*, *y*, dan *z*), jika ditulis m_6 maka ini berarti *minterm* xyz' karena 6 (desimal) = 110 (biner); di sini $x = 1$, $y = 1$ dan $z = 0$. Pada kasus yang terakhir, peubah *x* dan *y* dinyatakan tanpa komplemen sedangkan peubah *z* dinyatakan dengan komplemen karena bernilai 0, sehingga ditulis xyz' .

Maxterm dilambangkan sebagai huruf *M* besar berindeks. Indeks menyatakan nilai desimal dari *string* biner yang merepresentasikan $x + y$. Misalnya pada *term* dengan 2 peubah *x* dan *y*, indeks 0 pada M_0 menyatakan nilai desimal dari 00 ($x = 0$ dan $y = 0$), indeks 1 pada M_1 menyatakan nilai desimal dari 01 ($x = 0$ dan $y = 1$), dan seterusnya. Jadi, untuk *maxterm* dari 3 peubah (*x*, *y*, dan *z*), jika ditulis M_6 maka ini berarti *minterm* $x' + y' + z$ karena 6 (desimal) = 110 (biner); di sini $x = 1$, $y = 1$ dan $z = 0$. Peubah *x* dan *y* dinyatakan dengan komplemen sedangkan peubah *z* dinyatakan tanpa komplemen karena bernilai 0, sehingga ditulis $x' + y' + z$.

Tabel 7.8

<i>x</i>	<i>y</i>	<i>Minterm</i>		<i>Maxterm</i>	
		Suku	Lambang	Suku	Lambang
0	0	$x'y'$	m_0	$x + y$	M_0
0	1	$x'y$	m_1	$x + y'$	M_1
1	0	xy'	m_2	$x' + y$	M_2
1	1	xy	m_3	$x' + y'$	M_3

Tabel 7.9

x	y	z	Minterm		Maxterm	
			Suku	Lambang	Suku	Lambang
0	0	0	$x'y'z'$	m_0	$x + y + z$	M_0
0	0	1	$x'y'z$	m_1	$x + y + z'$	M_1
0	1	0	$x'y z'$	m_2	$x + y' + z$	M_2
0	1	1	$x'y z$	m_3	$x + y' + z'$	M_3
1	0	0	$x y'z'$	m_4	$x' + y + z$	M_4
1	0	1	$x y'z$	m_5	$x' + y + z'$	M_5
1	1	0	$x y z'$	m_6	$x' + y' + z$	M_6
1	1	1	$x y z$	m_7	$x' + y' + z'$	M_7

Jika diberikan sebuah tabel kebenaran, kita dapat membentuk fungsi Boolean dalam bentuk kanonik (SOP atau POS) dari tabel tersebut dengan cara mengambil *minterm* atau *maxterm* dari setiap nilai fungsi yang bernilai 1 (untuk SOP) atau 0 (untuk POS).

Untuk membentuk fungsi dalam bentuk SOP, tinjau kombinasi nilai-nilai peubah yang memberikan nilai fungsi sama dengan 1. Misalkan kombinasi nilai-nilai peubah yang memberikan nilai fungsi sama dengan 1 adalah 001, 100, dan 111, maka bentuk SOP fungsi tersebut adalah

$$f(x, y, z) = x'y'z + xy'z' + xyz.$$

Untuk membentuk fungsi dalam bentuk SOP, tinjau kombinasi nilai-nilai peubah yang memberikan nilai fungsi sama dengan 0. Misalkan kombinasi nilai-nilai peubah yang memberikan nilai fungsi sama dengan adalah 000, 010, 101, dan 110, maka bentuk POS fungsi tersebut adalah

$$f(x, y, z) = (x + y + z)(x + y' + z)(x' + y + z')(x' + y' + z).$$

Contoh 7.10

Tinjau fungsi Boolean yang dinyatakan oleh Tabel 7.10 di bawah ini. Nyatakan fungsi tersebut dalam bentuk kanonik SOP dan POS.

Penyelesaian:

(a) SOP

Kombinasi nilai-nilai peubah yang menghasilkan nilai fungsi sama dengan 1 adalah 001, 100, dan 111, maka fungsi Booleannya dalam bentuk kanonik SOP adalah

$$f(x, y, z) = x'y'z + xy'z' + xyz$$

atau (dengan menggunakan lambang *minterm*),

$$f(x, y, z) = m_1 + m_4 + m_7 = \sum (1, 4, 7)$$

(b) POS

Kombinasi nilai-nilai peubah yang menghasilkan nilai fungsi sama dengan 0 adalah 000, 010, 011, 101, dan 110, maka fungsi Booleannya dalam bentuk kanonik POS adalah

$$f(x, y, z) = (x + y + z)(x + y' + z)(x + y' + z')(x' + y + z')(x' + y' + z)$$

atau dalam bentuk lain,

$$f(x, y, z) = M_0 M_2 M_3 M_5 M_6 = \prod(0, 2, 3, 5, 6)$$

■

Tabel 7.10

x	y	z	f(x, y, z)
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Catatan:

Notasi \sum dan \prod berguna untuk mempersingkat penulisan ekspresi dalam bentuk SOP dan POS.

Fungsi Boolean yang tidak berbentuk kanonik dapat dinyatakan dalam bentuk kanonik dengan cara melengkapi literal yang belum terdapat di dalam setiap sukunya. Hal ini ditunjukkan pada Contoh 7.11 dan 7.12 berikut ini.

Contoh 7.11

Nyatakan fungsi Boolean $f(x, y, z) = x + y'z$ dalam bentuk kanonik SOP dan POS.

Penyelesaian:

(a) SOP

Kita harus melengkapi terlebih dahulu literal untuk setiap suku agar jumlahnya sama.

$$\begin{aligned}x &= x(y + y') \\&= xy + xy' \\&= xy(z + z') + xy'(z + z') \\&= xyz + xyz' + xy'z + xy'z' \\y'z &= y'z(x + x') \\&= xy'z + x'y'z\end{aligned}$$

■ atau $f(x, y, z) = M_0 M_2 M_4 M_5 = \prod(0, 2, 4, 5)$

Jadi, $f(x, y, z) = (x + y + z)(z + y + x)(z + y + x)(x + y + z)$

$$\begin{aligned} (z + y + x)(z + y + x) &= z + y + x \\ (z + y + x)(z + y + x) &= z + x \\ x + y + z + zz &= (x + y + z)(x + y + z) \end{aligned}$$

Lengkap! literal unitik setiap suku agar jumlahnya sama:

$$\begin{aligned} (z + y)(z + x)(x + y) &= \\ (z + y)(z + x)(x + y)(x + x) &= \\ (xy + z)(xy + z) &= \\ f(x, y, z) &= xy + xz \end{aligned}$$

Penyelesaian:

Nyatakan fungsi Boolean $f(x, y, z) = xy + xz$ dalam bentuk kanonik POS.

Contoh 7.12

■ atau $f(x, y, z) = M_0 M_2 M_3 = \prod(0, 2, 3)$

Jadi, $f(x, y, z) = (x + y + z)(x + y + z)(z + y + x)(z + y + x)$

$$(z + y + x)(z + y + x) = z + x$$

$$(z + y + x)(z + y + x)(z + y + z) = x + y + z + zz$$

Kita harus mengekapi terlebih dahulu literal pada setiap suku agar jumlahnya sama:

$$\begin{aligned} (z + x)(z + y) &= \\ f(x, y, z) &= x + y \\ (b) \text{ POS} & \end{aligned}$$

atau $f(x, y, z) = m_1 + m_4 + m_5 + m_6 + m_7 = \Sigma(1, 4, 5, 6, 7)$

Jadi $f(x, y, z) = x + y + z$

7.10 Konversi Antar Bentuk Kanonik

Fungsi Boolean dalam bentuk kanonik SOP dapat ditransformasi ke bentuk kanonik POS, demikian pula sebaliknya. Misalkan f adalah fungsi Boolean dalam bentuk SOP dengan tiga peubah:

$$f(x, y, z) = \Sigma (1, 4, 5, 6, 7)$$

dan f' adalah fungsi komplemen dari f ,

$$f'(x, y, z) = \Sigma (0, 2, 3) = m_0 + m_2 + m_3$$

Dengan menggunakan hukum De Morgan, kita dapat memperoleh fungsi f dalam bentuk POS:

$$\begin{aligned} f'(x, y, z) &= (f'(x, y, z))' = (m_0 + m_2 + m_3)' \\ &= m_0' \cdot m_2' \cdot m_3' \\ &= (x'y'z')' (x'y'z')' (x'y'z')' \\ &= (x + y + z) (x + y' + z) (x + y' + z') \\ &= M_0 M_2 M_3 \\ &= \prod (0, 2, 3) \end{aligned}$$

Jadi, $f(x, y, z) = \Sigma (1, 4, 5, 6, 7) = \prod (0, 2, 3)$.

Kesimpulan:

$$m_j' = M_j$$

Contoh 7.13

Nyatakan $f(x, y, z) = \prod (0, 2, 4, 5)$ dan $g(w, x, y, z) = \Sigma(1, 2, 5, 6, 10, 15)$ dalam bentuk SOP.

Penyelesaian:

Fungsi f dikonversi ke SOP dengan mengambil nilai-nilai antara 0 sampai 7 selain 0, 2, 4, dan 5, menjadi:

$$f(x, y, z) = \Sigma (1, 3, 6, 7)$$

Fungsi g dikonversi ke POS dengan mengambil nilai-nilai antara 0 sampai 15 selain 1, 2, 5, 6, 10, dan 15, menjadi:

$$g(w, x, y, z) = \prod (0, 3, 4, 7, 8, 9, 11, 12, 13, 14)$$

■

baku POS.

Cara lain untuk mengekspresikan fungsi Boolean adalah bentuk baku (standard). Padahal bentuk ini, suku-suku yang mempunyai adaptasi mengandung satu, dua atau sejumlah literal. Dua tipe bentuk baku adalah bentuk baku SOP dan bentuk baku POS.

Dua bentuk kanonik adalah bentuk normal (x) atau dalam bentuk komplementnya (\bar{x}). Dari tabel kebenaran, Bentuk ini umumnya sangat jarang muncul, karena setiap suku (term) di dalam bentuk kanonik harus mengandung literal lengkap, baik dalam bentuk normal (x) atau dalam bentuk komplementnya (\bar{x}).

7.11 Bentuk Baku

$$f(x, y, z) = M_3 = x + \bar{y} + z$$

(b) POS

$$\begin{aligned} \text{atau } f(x, y, z) &= m_0 + m_1 + m_2 + m_4 + m_5 + m_7 \\ &= \bar{x}\bar{y}z + \bar{x}\bar{y}\bar{z} + \bar{x}y\bar{z} + \bar{x}yz + xy\bar{z} + xyz \\ &= (\bar{x} + \bar{y} + z)(\bar{x} + \bar{y} + \bar{z})(\bar{x} + y + \bar{z}) \\ &= (\bar{x} + z + \bar{y})(x + z + \bar{y})(x + y + z) \\ f(x, y, z) &= \bar{y}x + \bar{y}xz + \bar{y}xy + xyz \end{aligned}$$

(a) SOP

Penyelisihan:

Carilah bentuk kanonik SOP dan POS dari $f(x, y, z) = \bar{y} + \bar{xy} + x\bar{yz}$

Contoh 7.15

$$f(x, y) = M_2 M_3 = (x + \bar{y})(x + \bar{y})$$

(b) POS

$$\begin{aligned} \text{atau } f(x, y) &= m_0 + m_1 \\ f(x, y) &= x = \bar{x}(y + \bar{y}) = \bar{x}y + x\bar{y} \end{aligned}$$

(a) SOP

Penyelisihan:

Carilah bentuk kanonik, SOP dan POS, dari fungsi Boolean $f(x, y) = x$,

Contoh 7.14

Contohnya,

$$f(x, y, z) = y' + xy + x'yz \quad (\text{bentuk baku SOP})$$

$$f(x, y, z) = x(y' + z)(x' + y + z') \quad (\text{bentuk baku POS})$$

Perbedaan antara bentuk kanonik dan bentuk baku adalah, pada bentuk kanonik, setiap *term* harus mengandung literal lengkap, sedangkan pada bentuk baku setiap *term* tidak harus mengandung literal lengkap.

7.12 Aplikasi Aljabar Boolean

Aljabar Boolean memiliki aplikasi yang luas dalam bidang keteknikan, antara lain di bidang jaringan pensaklaran dan rangkaian digital. Masing-masing aplikasi dibahas di bawah ini.

7.12.1 Jaringan Pensaklaran (*Switching Network*)

Saklar adalah objek yang mempunyai dua buah status: buka dan tutup. Kita dapat mengasosiasikan setiap peubah di dalam fungsi Boolean sebagai saklar dalam sebuah saluran yang dialiri listrik, air, gas, informasi atau benda lain yang mengalir. Secara fisik, saklar ini dapat berupa keran di dalam pipa hidrolik, transistor atau dioda dalam rangkaian listrik, *dispatcher* pada alat rumah tangga, atau sembarang alat lain yang dapat melewatkannya atau menghambat aliran.

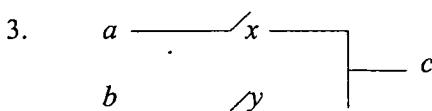
Tiga bentuk saklar paling sederhana:

1. $a - \overbrace{\hspace{1cm}}^x - b$

Keluaran b ada jika dan hanya jika saklar x ditutup $\Rightarrow x$

2. $a - \overbrace{\hspace{1cm}}^x - \overbrace{\hspace{1cm}}^y - b$

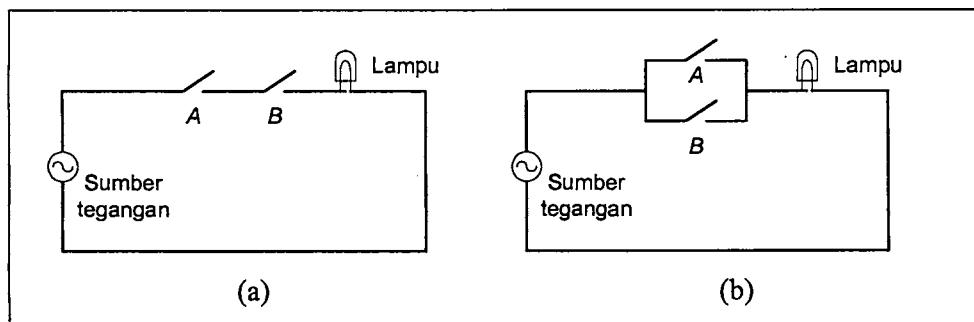
Keluaran b ada jika dan hanya jika x dan y keduanya ditutup $\Rightarrow xy$



Keluaran c ada jika dan hanya jika x atau y ditutup $\Rightarrow x + y$

Catatan: jika saklar menyatakan keran di dalam sistem perpipaan, maka kata “ditutup” diganti dengan “dibuka”. Jadi, pada keran yang terhubung secara serial (nomor 2 di atas), keluaran b hanya ada jika dan hanya jika keran x dan y keduanya dibuka. Begitu juga pada keran yang terhubung secara paralel, keluaran b hanya ada jika salah satu keran x atau y dibuka.

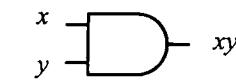
Gambar 7.1(a) memperlihatkan contoh rangkaian pensaklaran pada rangkaian listrik dalam hubungan seri. A dan B adalah saklar yang terpasang secara serial. Lampu hanya menyala jika A dan B ditutup (*closed*). Dalam ekspresi Boolean, hubungan serial ini dinyatakan sebagai AB . Sedangkan Gambar 7.1(b) memperlihatkan contoh rangkaian pensaklaran pada rangkaian listrik dalam hubungan paralel. A dan B adalah saklar yang terpasang secara paralel. Lampu hanya menyala jika salah satu dari A atau B ditutup. Dalam ekspresi Boolean, hubungan paralel ini dinyatakan sebagai $A + B$.



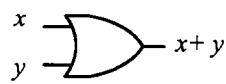
Gambar 7.1 (a) Dua saklar dalam hubungan AND, (b) dua saklar dalam hubungan OR

7.12.2 Sirkuit Elektronik

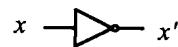
Komputer dan peralatan elektronik lain dibuat dari sejumlah rangkaian atau sirkuit (*circuit*). Sirkuit menerima masukan dan keluaran berupa pulsa-pulsa listrik yang dapat dipandang sebagai 0 atau 1. Aljabar Boolean digunakan untuk memodelkan sirkuit elektronik. Elemen dasar dari sirkuit adalah gerbang (*gate*). Sirkuit elektronik dimodelkan dengan sejumlah **gerbang logika** (*logic gate*). Setiap gerbang mengimplementasikan sebuah operasi Boolean. Ada tiga macam gerbang dasar: AND, OR, dan NOT (lihat Gambar 7.2). Sirkuit yang dibentuk oleh kombinasi beberapa gerbang logika disebut **sirkuit logika**. Secara fisik, sirkuit logika diimplementasikan sebagai rangkaian listrik spesifik, tetapi rangkaian listrik di luar pembahasan buku ini. Sirkuit yang kita bahas di dalam buku ini memberikan keluaran yang bergantung pada masukannya. Rangkaian tidak menyimpan keadaan saat ini, dengan kata lain tidak memiliki kemampuan memori. Sirkuit semacam ini disebut juga **sirkuit kombinasi** (*combinational circuit*).



Gerbang AND dua-masukan



Gerbang OR dua-masukan



Gerbang NOT (*inverter*)

Gambar 7.2 Tiga gerbang logika dasar: AND, OR, dan NOT

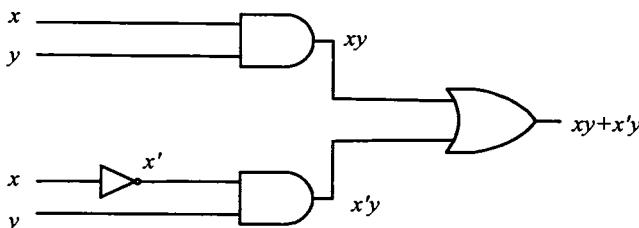
Contoh 7.17

Nyatakan fungsi $f(x, y, z) = xy + x'y$ ke dalam rangkaian logika.

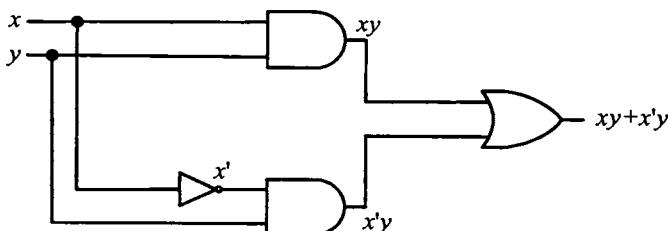
Penyelesaian:

Ada beberapa cara penggambaran rangkaian logika. Di bawah ini disajikan tiga cara yang lazim dipakai.

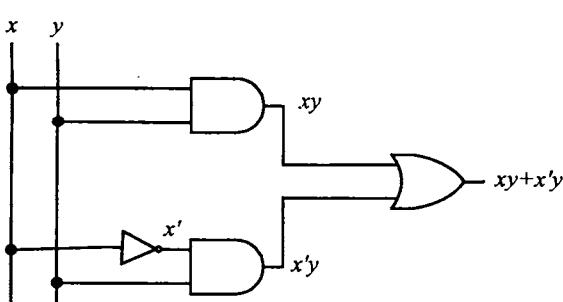
(a) Cara pertama



(b) Cara kedua

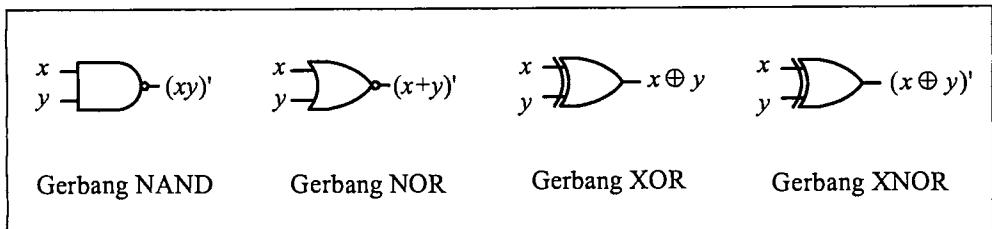


(c) Cara ketiga



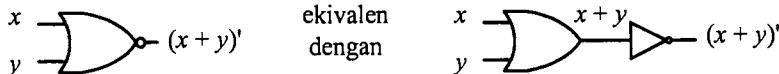
■

Selain gerbang dasar AND, OR, dan NOT, masih terdapat gerbang logika turunan, yaitu NAND, NOR, XOR (*Exclusive OR*), dan XNOR yang diturunkan dari tiga gerbang dasar tersebut (Gambar 7.3).

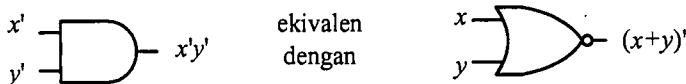


Gambar 7.3 Empat gerbang turunan

Keempat gerbang di atas merupakan kombinasi dari gerbang-gerbang dasar, misalnya gerbang NOR disusun oleh kombinasi gerbang OR dan gerbang NOT:



Selain itu, dengan menggunakan hukum De Morgan, kita juga dapat membuat gerbang logika yang ekivalen dengan gerbang NOR dan NAND di atas:



7.13 Penyederhanaan Fungsi Boolean

Fungsi Boolean seringkali mengandung operasi-operasi yang tidak perlu, literal atau suku-suku yang berlebihan. Oleh karena itu, kita dapat menyederhanakan fungsi Boolean lebih lanjut. Menyederhanakan fungsi Boolean artinya mencari bentuk fungsi lain yang ekivalen tetapi dengan jumlah literal atau operasi yang lebih sedikit. Penyederhanaan fungsi Boolean disebut juga **minimisasi fungsi**.

Contohnya, $f(x, y) = x'y + xy' + y'$ dapat disederhanakan menjadi $f(x, y) = x' + y'$.

Dipandang dari segi aplikasi aljabar Boolean, fungsi Boolean yang lebih sederhana berarti rangkaian logikanya juga lebih sederhana (menggunakan jumlah gerbang logika lebih sedikit).

Ada tiga metode yang dapat digunakan untuk menyederhanakan fungsi Boolean:

1. Secara aljabar, menggunakan hukum-hukum aljabar Boolean.
2. Metode Peta Karnaugh.
3. Metode Quine-McCluskey (metode tabulasi)

Masing-masing metode di atas akan kita bahas lebih mendalam di dalam upabab-upabab berikut ini.

7.13.1 Penyederhanaan Fungsi Boolean Secara Aljabar

Jumlah literal di dalam sebuah fungsi Boolean dapat diminimumkan dengan trik manipulasi aljabar. Sayangnya, tidak ada aturan khusus yang harus diikuti yang akan menjamin menuju ke jawaban akhir. Metode yang tersedia adalah prosedur yang *cut-and-try* yang memanfaatkan postulat, hukum-hukum dasar, dan metode manipulasi lain yang sudah dikenal.

Contoh 7.18

Sederhanakan fungsi-fungsi Boolean berikut ini:

- (a) $f(x, y) = x + x'y$
- (b) $f(x, y) = x(x' + y)$
- (c) $f(x, y, z) = x'y'z + x'yz + xy'$
- (d) $f(x, y, z) = xz' + y'z + xyz'$
- (e) $f(x, y, z) = (x + z')(y' + z)(x + y + z')$

Penyelesaian:

- (a)
$$\begin{aligned} f(x, y) &= x + x'y \\ &= (x + x')(x + y) \quad (\text{Hukum distributif}) \\ &= 1 \cdot (x + y) \quad (\text{Hukum komplemen}) \\ &= x + y \quad (\text{Hukum identitas}) \end{aligned}$$
- (b)
$$\begin{aligned} f(x, y) &= x(x' + y) \\ &= xx' + xy \quad (\text{Hukum distributif}) \\ &= 0 + xy \quad (\text{Hukum komplemen}) \\ &= xy \quad (\text{Hukum identitas}) \end{aligned}$$
- (c)
$$\begin{aligned} f(x, y, z) &= x'y'z + x'yz + xy' \\ &= x'z(y' + y) + xy' \quad (\text{Hukum distributif}) \\ &= x'z \cdot 1 + xz' \quad (\text{Hukum komplemen}) \\ &= x'z + xz' \quad (\text{Hukum identitas}) \end{aligned}$$
- (d)
$$\begin{aligned} f(x, y, z) &= xz' + y'z + xyz' \\ &= xz' \cdot 1 + y'z + xyz' \quad (\text{Hukum identitas}) \\ &= xz'(1 + y) + y'z \quad (\text{Hukum distributif}) \end{aligned}$$

$$\begin{aligned}
 &= xz' \cdot 1 + y'z \\
 &= xz' + y'z
 \end{aligned}
 \quad \begin{array}{l} (\text{Hukum dominansi}) \\ (\text{Hukum identitas}) \end{array}$$

- (e) $f(x, y, z) = (x + z')(y' + z)(x + y + z') = (x + z')(y' + z)$ dengan dualitas dari fungsi nomor (d). ■

7.13.2 Metode Peta Karnaugh

Metode Peta Karnaugh (atau *K-map*) merupakan metode grafis untuk menyederhanakan fungsi Boolean. Metode ini ditemukan oleh Maurice Karnaugh pada tahun 1953. Peta Karnaugh adalah sebuah diagram/peta yang terbentuk dari kotak-kotak (berbentuk bujursangkar) yang bersisian. Tiap kotak merepresentasikan sebuah *minterm*. Tiap kotak dikatakan bertetangga jika *minterm-minterm* yang merepresentasikannya berbeda hanya 1 buah literal.

Peta Karnaugh dapat dibentuk dari fungsi Boolean yang dispesifikasikan dengan dengan ekspresi Boolean maupun fungsi yang direpresentasikan dengan tabel kebenaran. Di bawah ini kita hanya membicarakan peta Karnaugh untuk fungsi dengan 2, 3, dan 4 buah peubah.

Peta Karnaugh dengan Dua Peubah

Misalkan dua peubah di dalam fungsi Boolean adalah x dan y . Baris pada peta Karnaugh untuk peubah x dan kolom untuk peubah y . Baris pertama diidentifikasi nilai 0 (menyatakan x'), sedangkan baris kedua dengan 1 (menyatakan x). Kolom pertama diidentifikasi nilai 0 (menyatakan y'), sedangkan kolom kedua dengan 1 (menyatakan y). Setiap kotak merepresentasikan *minterm* dari kombinasi baris dan kolom yang bersesuaian. Di bawah ini diberikan tiga cara yang lazim digunakan sejumlah literatur dalam menggambarkan peta Karnaugh untuk dua peubah. Kita akan lebih sering menggunakan cara penyajian nomor dua.

m_0	m_1
m_2	m_3

Penyajian 1

		y	
		0	1
x	0	$x'y'$	$x'y$
	1	xy'	xy

Penyajian 2

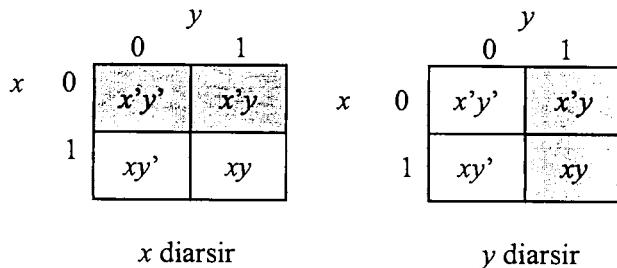
x'	y'	y
x	xy'	xy

Penyajian 3

Perhatikanlah bahwa dua kotak yang bertetangga hanya berbeda satu literal. Kotak $x'y'$ dan $x'y$ misalnya, hanya berbeda pada literal kedua (y' dan y), sedangkan literal pertama sama (yaitu x). Jika *minterm* pada setiap kotak

direpresentasikan dengan *string* biner, maka dua kotak yang bertetangga hanya berbeda 1 bit (contohnya 00 dan 01 pada kedua kotak tersebut hanya berbeda satu bit, yaitu pada bit kedua).

Peta Karnaugh dapat dianggap sebagai diagram Venn, yang dalam hal ini x diwakili oleh titik-titik pada baris pertama, dan y diwakili titik-titik pada kolom kedua, seperti ditunjukkan pada gambar di bawah ini.:



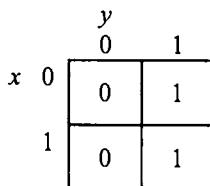
Contoh 7.19

Gambarkan peta Karnaugh untuk $f(x, y) = xy + x'y$

Penyelesaian:

Peubah tanpa komplemen dinyatakan sebagai 1 dan peubah dengan komplemen dinyatakan sebagai 0, sehingga xy dinyatakan sebagai 11 dan $x'y$ dinyatakan sebagai 01. Kotak-kotak yang merepresentasikan *minterm* 11 dan 01 diisi dengan 1, sedangkan kotak-kotak yang tidak terpakai diisi dengan 0.

Hasil pemetaan:



Contoh 7.20

Diberikan fungsi Boolean yang direpresentasikan dengan tabel kebenaran (Tabel 7.11). Petakan fungsi tersebut ke peta Karnaugh.

Tabel 7.11

x	y	$f(x, y)$
0	0	0
0	1	0
1	0	1
1	1	1

Penyelesaian:

Tinjau hanya nilai fungsi yang memberikan 1. Fungsi Boolean yang merepresentasikan tabel kebenaran adalah $f(x, y) = xy' + xy$. Tempatkan 1 di dalam kotak di peta Karnaugh untuk kombinasi nilai x dan y yang bersesuaian (dalam hal ini 10 dan 11)

		y	
		0	1
x	0	0	0
	1	1	1

■

Peta Karnaugh dengan Tiga Peubah

Untuk fungsi Boolean dengan tiga peubah (misalkan x , y , dan z), jumlah kotak di dalam peta Karnaugh meningkat menjadi $2^3 = 8$. Baris pada peta Karnaugh untuk peubah x dan kolom untuk peubah yz . Baris pertama diidentifikasi nilai 0 (menyatakan x'), sedangkan baris kedua dengan 1 (menyatakan x). Kolom pertama diidentifikasi nilai 00 (menyatakan $x'y'$), kolom kedua diidentifikasi nilai 01 (menyatakan xy'), kolom ketiga diidentifikasi nilai 11 (menyatakan xy), sedangkan kolom keempat diidentifikasi nilai 10 (menyatakan xy'). Perhatikanlah bahwa antara satu kolom dengan kolom berikutnya hanya berbeda satu bit. Setiap kotak merepresentasikan *minterm* dari kombinasi baris dan kolom yang bersesuaian.

			yz	
			00	01
x	0	$x'y'z'$	$x'y'z$	$x'yz$
	1	$xy'z'$	$xy'z$	xyz

m_0	m_1	m_3	m_2	
m_4	m_5	m_7	m_6	

Perhatikan urutan dari m_i -nya. Urutan disusun sedemikian rupa sehingga setiap dua kotak yang bertetangga hanya berbeda satu bit.

Contoh 7.21

Gambarkan peta Karnaugh untuk $f(x, y, z) = x'yz' + xyz' + xyz$

Penyelesaian:

- $x'yz' \rightarrow$ dalam bentuk biner : 010
- $xyz' \rightarrow$ dalam bentuk biner: 110
- $xyz \rightarrow$ dalam bentuk biner: 111

Kotak-kotak yang merepresentasikan *minterm* 010, 110, dan 111 diisi dengan 1, sedangkan kotak-kotak yang tidak terpakai diisi dengan 0.

		yz			
		00	01	11	10
x	0	0	0	0	1
	1	0	0	1	1



Contoh 7.22

Gambarkan peta Karnaugh untuk $f(x, y, z) = xz' + y$

Penyelesaian:

Kita pilah untuk masing-masing suku:

(i) xz' :

$x \rightarrow$ semua kotak pada baris ke-2

$z' \rightarrow$ semua kotak pada kolom ke-1 dan kolom ke-4

Irisan dari baris kedua dengan kolom pertama dan kolom keempat menyatakan menyatakan *minterm-minterm* yang merepresentasikan xz' . Isikan kotak-kotak yang beririsan dengan 1, seperti ditunjukkan pada gambar di bawah ini:

		yz			
		00	01	11	10
x	0				
	1	1			1

$xz' = xy'z' + xyz'$

(ii) y :

$y \rightarrow$ semua kotak pada kolom ke-3 dan kolom ke-4

Isikan kotak-kotak pada kolom 3 dan kolom 4 dengan 1, seperti ditunjukkan pada gambar di bawah ini:

		yz			
		00	01	11	10
x	0			1	1
	1			1	1

y

- (iii) Gabungkan (i) dan (ii) dan isikan kotak-kotak yang kosong dengan 0.
 Peta Karnaugh yang merepresentasikan $f(x, y, z) = xz' + y$ adalah:

		yz				
		00	01	11	10	
x		0	0	0	1	1
		1	1	0	1	1
$xz' + y$						

Contoh 7.23

Diberikan fungsi Boolean dengan tiga buah peubah yang direpresentasikan dengan tabel kebenaran (Tabel 7.12). Petakan fungsi tersebut ke peta Karnaugh.

Tabel 7.12

x	y	z	$f(x, y, z)$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

Penyelesaian:

Tinjau hanya nilai fungsi yang memberikan 1. Fungsi Boolean yang merepresentasikan tabel kebenaran adalah $f(x, y) = x'y'z + xy'z' + xy'z + xyz$. Tempatkan 1 di dalam kotak di peta Karnaugh untuk kombinasi nilai x dan y yang bersesuaian (dalam hal ini 001, 100, 101, dan 111):

		yz				
		00	01	11	10	
x		0	0	1	0	0
		1	1	1	1	0
$x'y'z + xy'z' + xy'z + xyz$						

Peta Karnaugh dengan Empat Peubah

Misalkan empat peubah di dalam fungsi Boolean adalah w, x, y , dan z . Jumlah kotak di dalam peta Karnaugh meningkat menjadi $2^4 = 16$. Baris pada peta Karnaugh untuk peubah wx dan kolom untuk peubah yz . Baris pertama diidentifikasi nilai

00 (menyatakan $w'x'$), baris kedua dengan 01 (menyatakan $w'x$), baris ketiga dengan 11 (menyatakan wx) dan baris keempat dengan 10 (menyatakan wx'). Kolom pertama diidentifikasi nilai 00 (menyatakan $y'z'$), kolom kedua diidentifikasi nilai 01 (menyatakan yz'), kolom ketiga diidentifikasi nilai 11 (menyatakan yz), sedangkan kolom keempat diidentifikasi nilai 10 (menyatakan $y'z$). Perhatikanlah bahwa antara satu kolom dengan kolom berikutnya hanya berbeda satu bit. Setiap kotak merepresentasikan *minterm* dari kombinasi baris dan kolom yang bersesuaian.

				yz	00	01	11	10	
m_0	m_1	m_3	m_2	wx	00	$w'x'y'z'$	$w'x'y'z$	$w'x'yz$	$w'x'yz'$
m_4	m_5	m_7	m_6	01	$w'xy'z'$	$w'xy'z$	$w'xyz$	$w'xyz'$	
m_{12}	m_{13}	m_{15}	m_{14}	11	$wxy'z'$	$wxy'z$	$wxyz$	$wxyz'$	
m_8	m_9	m_{11}	m_{10}	10	$wx'y'z'$	$wx'y'z$	$wx'yz$	$wx'yz'$	

Perhatikan urutan dari m_i -nya. Urutan disusun sedemikian rupa sehingga setiap dua kotak yang bertetangga hanya berbeda satu bit.

Contoh 7.24

Diberikan fungsi Boolean yang direpresentasikan dengan tabel kebenaran (Tabel 7.13). Petakan tabel tersebut ke peta Karnaugh.

Tabel 7.13

w	x	y	z	$f(w, x, y, z)$
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	0

Penyelesaian:

Tinjau hanya nilai fungsi yang memberikan 1. Fungsi Boolean yang merepresentasikan tabel kebenaran adalah $f(w, x, y, z) = w'x'y'z + w'xyz' + w'xyz + wxyz'$.

Hasil pemetaan tabel ke peta Karnaugh:

		yz		00	01	11	10
		00	0	1	0	1	
		01	0	0	1	1	
		11	0	0	0		1
		10	0	0	0	0	

Teknik Minimisasi Fungsi Boolean dengan Peta Karnaugh

Penggunaan Peta Karnaugh dalam penyederhanaan fungsi Boolean dilakukan dengan cara menggabungkan kotak-kotak yang bernilai 1 dan saling bersisian. Kelompok kotak yang bernilai 1 dapat membentuk pasangan (dua), kuad (empat), dan oktet (delapan). Tinjau masing-masing kelompok sebagai berikut melalui contoh:

Contoh 7.25 (Pasangan)

Diberikan peta Karnaugh yang merepresentasikan fungsi Boolean dengan 2 buah *term*:

		yz		00	01	11	10
		00	0	0	0	0	
		01	0	0	0	0	
		11	0	0	(1)	(1)	
		10	0	0	0	0	

Sebelum disederhanakan, dua buah *term* di atas menyatakan $f(w, x, y, z) = wxyz + wxyz'$

Cara penyederhanaan: Kelompokkan (misal dengan cara melingkari) dua buah 1 yang bertetangga. Perhatikan bahwa *minterm* pada kotak-kotak yang dilingkari ($wxyz$ dan $wxyz'$) memiliki literal yang sama, yaitu w , x , dan y (ditandai dengan angka 11 dan 1 yang dicetak tebal), sehingga

$$f(w, x, y, z) = wxy$$

Bukti secara aljabar:

$$\begin{aligned}f(w, x, y, z) &= wxyz + wxyz' \\&= wxy(z + z') \\&= wxy(1) \\&= wxy\end{aligned}$$

■

Contoh 7.26 (Kuad)

Diberikan peta Karnaugh yang merepresentasikan fungsi Boolean dengan 4 buah *term*

		yz	00	01	11	10
		wx	00	0	0	0
		01	0	0	0	0
		11	1	1	1	1
		10	0	0	0	0

Sebelum disederhanakan, empat buah *term* di dalam lingkaran kuad di atas menyatakan fungsi: $f(w, x, y, z) = wxy'z' + wxy'z + wxyz + wxzy'$

Cara penyederhanaan: perhatikan bahwa suku-suku pada kotak yang dilingkari memiliki literal yang sama, yaitu w dan x (ditandai dengan angka 11 yang dicetak tebal), sehingga

$$f(w, x, y, z) = wx$$

Bukti secara aljabar (sebuah kuad dapat dianggap sebagai 2 buah pasangan, lihat gambar di bawah):

$$\begin{aligned}f(w, x, y, z) &= wxy' + wxzy' \\&= wx(z' + z) \\&= wx(1) \\&= wx\end{aligned}$$

		yz	00	01	11	10
		wx	00	0	0	0
		01	0	0	0	0
		11	1	1	1	1
		10	0	0	0	0

■

Contoh bentuk kuad yang lain adalah seperti peta Karnaugh berikut:

$wx \backslash yz$	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	0	0
10	1	1	0	0

Sebelum disederhanakan, empat buah *term* di dalam lingkaran kuad di atas menyatakan fungsi: $f(w, x, y, z) = wxy'z' + wxy'z + wx'y'z' + wx'y'z$

Cara penyederhanaan: perhatikan bahwa suku-suku pada kotak yang dilingkari memiliki literal yang sama, yaitu w dan y' (ditandai dengan angka 1 dan 0 yang dicetak tebal), sehingga

$$f(w, x, y, z) = wy'$$

Contoh 7.27 (Oktet)

Diberikan peta Karnaugh sebagai berikut

$wx \backslash yz$	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	1	1
10	1	1	1	1

Sebelum disederhanakan, 8 buah *term* di dalam lingkaran oktet di atas menyatakan fungsi $f(w, x, y, z) = wxy'z' + wxy'z + wxyz' + wxy'z + wx'y'z' + wx'y'z + wx'yz + wx'yz'$.

Cara penyederhanaan: perhatikan bahwa suku-suku pada kotak yang dilingkari memiliki literal yang sama, yaitu w (ditandai dengan angka 1 yang dicetak tebal), sehingga

$$f(w, x, y, z) = w$$

Bukti secara aljabar (sebuah *oktet* dapat dianggap sebagai 2 buah *quad*, lihat gambar di bawah):

$$\begin{aligned} f(w, x, y, z) &= wy' + wy \\ &= w(y' + y) \\ &= w \end{aligned}$$

$wx \backslash yz$	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	1	1
10	1	1	1	1

■

Contoh 7.28

Andaikan suatu tabel kebenaran telah diterjemahkan ke dalam Peta Karnaugh di bawah ini. Sederhanakan fungsi Boolean yang bersesuaian sesederhana mungkin.

$wx \backslash yz$	00	01	11	10
00	0	(1)	(1)	(1)
01	0	0	0	1
11	(1)	(1)	0	1
10	(1)	(1)	0	(1)

■

Penyelesaian:

Kelompokkan 1 yang bertetangga sebanyak mungkin (dimulai dengan mencari oktet sebanyak-banyaknya terlebih dahulu, kemudian kuad, dan terakhir pasangan). Hasil pengelompokan diperlihatkan pada peta Karnaugh di atas. Terdapat sebuah oktet, sebuah kuad, dan sebuah pasangan.

Fungsi Boolean hasil penyederhanaan adalah $f(w, x, y, z) = wy' + yz' + w'x'z$

Untuk mendapatkan fungsi yang sesederhana mungkin, kita boleh membuat suatu pengelompokan 1 saling beririsan (*overlapping*) dengan pengelompokan 1 lainnya. Prinsip yang dipakai adalah: carilah pengelompokan 1 sebanyak mungkin! Contoh 7.29 memperlihatkan hal ini.

Contoh 7.29

Minimisasi fungsi Boolean yang bersesuaian dengan peta Karnaugh di bawah ini.

	$wx \backslash yz$	00	01	11	10
00	0	1	1	0	
01	0	1	1	1	1
11	0	1	1	1	1
10	1	1	1	0	

Penyelesaian:

Pengelompokan 1 menghasilkan sebuah oktet, sebuah kuad, dan sebuah pasangan yang beririsan. Fungsi Boolean hasil penyederhanaan adalah $f(w, x, y, z) = z + xy + wx'y'$.

Jika pengelompokan 1 pada Contoh 7.29 adalah seperti di bawah ini:

	$wx \backslash yz$	00	01	11	10
00	0	1	1	0	
01	0	1	1		1
11	0	1	1		1
10	1	1	1	0	

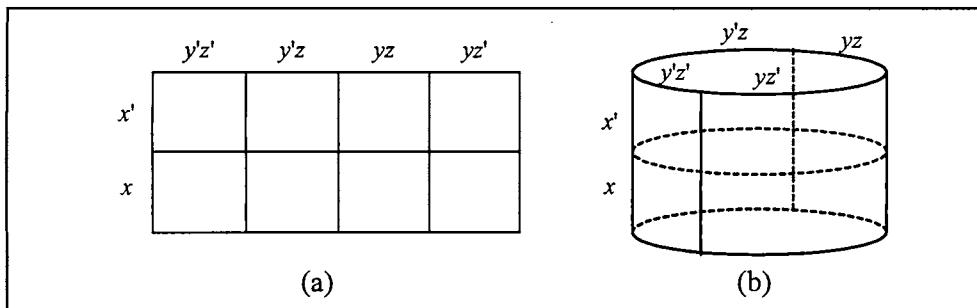
maka fungsi Boolean hasil penyederhanaan adalah

$$f(w, x, y, z) = y + xyz' + wx'y'z' \quad (\text{jumlah literal} = 8)$$

yang ternyata masih belum sederhana dibandingkan $f(w, x, y, z) = z + xy + wx'y'$ (jumlah literal = 6). ■

Penggulungan

Ada kemungkinan kotak-kotak yang bernilai 1 terdapat di sisi pinggir yang berseberangan (sisi kiri dengan sisi kanan, sisi atas dengan sisi bawah). Kotak-kotak yang terletak pada sisi berseberangan sebenarnya merupakan kotak yang bertetangga juga. Anggaplah peta Karnaugh seperti sebuah kertas yang ujung-ujungnya dapat ditautkan dengan cara menggulung. Dengan menautkan sisi kiri dengan sisi kanan dari Peta Karnaugh dengan tiga peubah, diperoleh bentuk gulungan seperti pada Gambar 7.4.



Gambar 7.4. (a) Peta Karnaugh "normal" dengan 3 peubah

(b) Peta Karnaugh dengan sisi kiri dan sisi kanan ditautkan (seperti digulung).

Contoh 7.30 (Penggulungan)

Sederhanakan fungsi Boolean $f(x, y, z) = x'y'z + xy'z' + xyz + xyz'$.

Penyelesaian:

		$y'z$	00	01	11	10
		x	0	0	1	0
y	0	0	0	1	0	
	1	1	0	1	1	

Jika penyederhanaan fungsi tanpa penggulungan, maka sebuah satuan di sisi kiri dan sebuah satuan di sisi kanan adalah sebuah *minterm* terpisah, sehingga hasil penyederhanaan adalah

$$f(x, y, z) = yz + xy'z' + xyz'$$

Namun, jika kita menggulung peta Karnaugh dengan menautkan sisi kiri dengan sisi kanan, kita peroleh pengelompokan 1 seperti pada peta di atas, sehingga fungsi Boolean hasil penyederhanaan adalah

$$f(x, y, z) = yz + xz'$$

Hasil terakhir ini dapat dibuktikan benar dengan memanipulasi hasil penyederhanaan pertama sbb:

$$\begin{aligned}f(x, y, z) &= yz + xy'z' + xyz' \\&= yz + xz' (y' + y) \quad (\text{Hukum distributif}) \\&= yz + xz' \cdot 1 \quad (\text{Hukum komplemen}) \\&= yz + xz'\end{aligned}$$

■

Kelompok Berlebihan

Jika pengelompokan 1 tidak dilakukan secara hati-hati, maka ada kemungkinan kita membuat kelompok 1 yang tidak perlu. Pengelompokan yang berlebihan (redundan) ini menghasilkan fungsi Boolean dengan *term* yang tidak perlu. Hal ini ditunjukkan pada Contoh 7.31 berikut ini.

Contoh 7.31 (Kelompok Berlebihan)

Sederhanakan fungsi Boolean yang bersesuaian dengan Peta Karnaugh di bawah ini.

		yz	00	01	11	10
		wx	00	01	11	10
00	01	00	0	0	0	0
		01	0	1	0	0
11	10	00	1	1	1	0
		10	0	0	1	0

Penyelesaian:

Jika pengelompokannya adalah seperti Peta Karnaugh di atas, maka fungsi hasil penyederhanaan adalah

$$f(w, x, y, z) = xy'z + wxz + wyz \rightarrow \text{masih belum sederhana.}$$

Kita dapat membuat fungsi yang lebih sederhana lagi dengan memperhatikan bahwa kelompok yang ketiga (lingkaran horizontal) merupakan kelompok yang berlebihan, karena ia meningkatkan jumlah suku (*term*). Kelompok ketiga ini memuat dua buah 1 yang sebenarnya sudah termasuk ke dalam kelompok lain. Kelompok berlebihan dapat dihilangkan dari Peta Karnaugh menjadi:

$wx \backslash yz$	00	01	11	10
00	0	0	0	0
01	0	1	0	0
11	0	1	1	0
10	0	0	1	0

maka fungsi Boolean hasil penyederhanaan adalah

$$f(w, x, y, z) = xy'z + wyz$$

yang ternyata lebih sederhana dibandingkan dengan jawaban pertama karena mengandung jumlah literal dan operasi biner lebih sedikit. ■

Ketidakunikian Fungsi Hasil Penyederhanaan

Metode peta Karnaugh menghasilkan fungsi Boolean yang lebih sederhana. Fungsi yang lebih sederhana mempunyai jumlah literal dan jumlah *term* yang lebih sedikit daripada fungsi asalnya. Namun, hasil penyederhanaan dengan peta Karnaugh tidak selalu unik. Artinya, mungkin terdapat beberapa bentuk fungsi minimasi yang berbeda meskipun jumlah literal dan jumlah *term*-nya sama. Hal ini diberikan pada contoh peta Karnaugh di bawah ini.

Kemungkinan pengelompokan I:					Kemungkinan pengelompokan II:				
$wx \backslash yz$	00	01	11	10	$wx \backslash yz$	00	01	11	10
00	0	0	(1)	(1)	00	0	0	(1)	(1)
01	0	(1)	0	0	01	0	(1)	0	0
11	(1)	0	(1)	(1)	11	(1)	0	(1)	(1)
10	(1)	(1)	(1)	0	10	(1)	(1)	(1)	0

Fungsi minimasi: $f(w, x, y, z) = w'x'y + w'xy'z + wy'z' + wxy + wx'z$

Fungsi minimasi: $f(w, x, y, z) = w'x'y + w'xy'z + wy'z' + wxy + wx'z$

Kemungkinan pengelompokan III:

$wx \backslash yz$	00	01	11	10
00	0	0	(1)	(1)
01	0	(1)	0	0
11	(1)	0	(1)	(1)
10	(1)	(1)	(1)	0

Fungsi minimasi: $f(w, x, y, z) = w'x'y + w'xy'z + wxz' + wxy + x'yz$

7.13.3 Contoh-contoh Penyederhanaan Fungsi Boolean

Di bawah ini diberikan beberapa contoh penyederhanaan fungsi dengan metode peta Karnaugh, agar pembaca dapat lebih memahami penggunaan metode tersebut. Kunci penyelesaian minimisasi fungsi dengan peta Karnaugh terletak pada cara kita mengelompokkan kotak-kotak yang berisi 1.

Contoh 7.32

Tentukan bentuk sederhana dari fungsi Boolean yang merepresentasikan tabel kebenaran (Tabel 7.14) dalam bentuk baku SOP dan bentuk baku POS.

Tabel 7.14

$x \backslash y$	0	1	$z \backslash$	$f(x, y, z)$
0	0	0	0	0
0	0	0	1	1
0	1	0	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	1	0	0	1
1	1	1	1	0

Penyelesaian:

(a) Bentuk baku SOP: kelompokkan 1

$x \backslash yz$	00	01	11	10
0	0	(1)	(1)	0
1	(1)	0	0	(1)

Fungsi minimasi: $f(x, y, z) = x'z + xz'$

(b) Bentuk baku POS: kelompokkan 0

	yz x\y	00	01	11	10
0	0	1	1	0	
1	1	0	0	1	

Fungsi minimasi: $f(x, y, z) = (x' + z')(x + z)$

Contoh 7.33

Minimisasi fungsi Boolean $f(x, y, z) = x'z + x'y + xy'z + yz$

Penyelesaian:

	yz x\y	00	01	11	10
0	0	1	1	1	1
1	0	1	1	1	0

Fungsi minimasi: $f(x, y, z) = z + x'y$

Contoh 7.34

Minimisasi fungsi Boolean $f(x, y, z) = \Sigma(0, 2, 4, 5, 6)$

Penyelesaian:

Peta Karnaugh untuk fungsi tersebut adalah:

	yz x\y	00	01	11	10
0	1	0	0	1	
1	1	1	0	1	

Hasil penyederhanaan: $f(x, y, z) = z' + xy'$

Contoh 7.35

Sederhanakan fungsi Boolean yang bersesuaian dengan peta Karnaugh berikut ini.

Penyelesaian:

		00	01	11	10
		00	0	0	0
		01	0	0	1
		11	(1)	(1)	(1)
		10	0	(1)	(1)

Hasil penyederhanaan: $f(w, x, y, z) = wx + wz + wy + xyz$

Contoh 7.36

Minimisasi fungsi Boolean $f(w, x, y, z) = w'x'y' + x'yz' + w'xyz' + wx'y'$

Penyelesaian:

Peta Karnaugh untuk fungsi tersebut adalah:

		00	01	11	10	
		00	(1)	(1)	0	(1)
		01	0	0	0	(1)
		11	0	0	0	0
		10	(1)	(1)	0	(1)

Hasil penyederhanaan: $f(w, x, y, z) = x'y' + x'z' + w'yz'$

Contoh 7.37

Minimisasi fungsi Boolean $f(w, x, y, z) = \Sigma (0,1,2,4,5,6,8,9,12,13,14)$

Penyelesaian:

Peta Karnaugh untuk fungsi tersebut adalah:

wx	00	01	11	10
yz	00	01	11	10
00	1	1	0	1
01	1	1	0	1
11	1	1	0	1
10	1	1	0	0

Hasil penyederhanaan: $f(w, x, y, z) = y' + w'z' + xz'$ ■

Contoh 7.38

Sederhanakan fungsi Boolean berikut: $f(w, x, y, z) = \Sigma (0,1,2,5,8,9,10)$ ke dalam bentuk baku POS.

Penyelesaian:

Untuk memperoleh POS, kelompokkan kotak-kotak yang berelemen 0 dengan cara yang sama seperti pengelompokan 1.

wx	00	01	11	10
yz	00	01	11	10
00	1	1	0	1
01	0	1	0	0
11	0	0	0	0
10	1	1	0	0

Hasil penyederhanaan: $f(w, x, y, z) = (w' + x')(y' + z)(x' + z)'$ ■

Contoh 7.39

Sederhanakan fungsi $f(w,x,y,z) = (w+x')(w+x+y)(w'+x'+y')(w'+x+y+z')$ dengan menggunakan Peta Karnaugh. Hasil penyederhanaan dalam bentuk baku SOP dan POS.

Penyelesaian:

Peta Karnaugh untuk fungsi tersebut adalah:

	00	01	11	10
00	0	0	1	1
01	0	0	0	0
11	1	1	0	0
10	1	0	1	1

Pengelompokan 1 dinyatakan dengan garis penuh, sedangkan pengelompokan nol dinyatakan dengan garis putus-putus.

Hasil penyederhanaan:

$$\text{SOP: } f(w, x, y, z) = x'y + wxy' + wy'z' \quad (\text{garis penuh})$$

$$\text{POS: } f(w, x, y, z) = (x' + y')(w + y)(x + y + z') \quad (\text{garis putus-putus})$$

Contoh 7.40

Sederhanakan fungsi $f(x, y, z, t) = xy' + xyz + x'y'z' + x'yzt'$

Penyelesaian:

Pengelompokan yang berlebihan:

	00	01	11	10
00	1	1	0	0
01	0	0	0	1
11	0	0	1	1
10	1	1	1	1

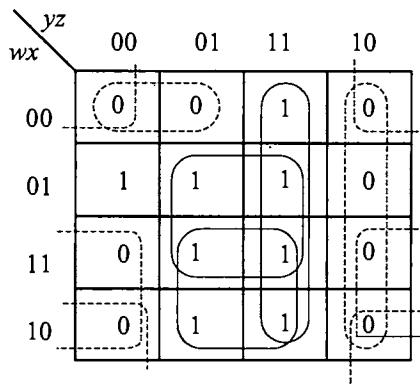
Pengelompokan yang benar:

	00	01	11	10
00	1	1	0	0
01	0	0	0	1
11	0	0	1	1
10	1	1	1	1

Fungsi minimasi: $f(x, y, z, t) = y'z' + xz + yzt'$

Contoh 7.41

Minimasi fungsi yang telah dipetakan ke peta Karnaugh di bawah ini dalam bentuk baku SOP dan bentuk baku POS.



Penyelesaian:

(lihat peta di atas)

$$\text{SOP: } f(w, x, y, z) = yz + wz + xz + w'xy' \quad (\text{garis penuh})$$

$$\text{POS: } f(w, x, y, z) = (y' + z)(w' + z)(x + z)(w + x + y) \quad (\text{garis putus-putus})$$

7.13.4 Peta Karnaugh untuk Lima Peubah

Peta Karnaugh untuk lima peubah dibuat dengan anggapan ada dua buah peta empat peubah yang disambungkan. Demikian juga untuk enam peubah dianggap ada dua buah peta empat peubah yang disambungkan. Setiap ‘sub-peta’ ditandai dengan garis ganda di tengah-tengahnya. Dua kotak dianggap bertetangga jika secara fisik berdekatan **dan** merupakan pencerminan terhadap garis ganda.

Contoh peta lima peubah:

	000	001	011	010	110	111	101	100
00	m_0	m_1	m_3	m_2	m_6	m_7	m_5	m_4
01	m_8	m_9	m_{11}	m_{10}	m_{14}	m_{15}	m_{13}	m_{12}
11	m_{24}	m_{25}	m_{27}	m_{26}	m_{30}	m_{31}	m_{29}	m_{28}
10	m_{16}	m_{17}	m_{19}	m_{18}	m_{22}	m_{23}	m_{21}	m_{20}

↑
Garis pencerminan

Pada peta Karnaugh di atas, m_{31} akan berdekatan dengan m_{30} , m_{15} , m_{29} , m_{23} , dan m_{27} .

Contoh 7.42

(Contoh penggunaan peta 5 peubah) Carilah fungsi sederhana dari

$$f(v, w, x, y, z) = \Sigma (0, 2, 4, 6, 9, 11, 13, 15, 17, 21, 25, 27, 29, 31)$$

Penyelesaian:

Peta Karnaugh dari fungsi tersebut adalah:

\backslash	$x\bar{y}z$	000	001	110	010	110	111	101	100
$v\bar{w}$	1	0	0	1	1	0	0	1	
00									
01	0	1	1	0	0	1	1	0	
11	0	1	1	0	0	1	1	0	
10	0	1	0	0	0	0	1	0	

Fungsi minimasi: $f(v, w, x, y, z) = wz + v'w'z' + vy'z$ ■

7.13.5 Keadaan *Don't Care*

Keadaan *don't care* adalah kondisi nilai peubah yang tidak diperhitungkan oleh fungsinya. Artinya nilai 1 atau 0 dari peubah *don't care* tidak berpengaruh pada hasil fungsi tersebut. Contohnya pada perancangan rangkaian digital untuk memperagakan angka desimal dari 0 sampai 9. Jumlah bit yang diperlukan untuk merepresentasikan angka desimal 0 sampai 9 adalah empat bit. Jadi, rangkaian hanya menerima masukan 4-bit, namun tidak semua kombinasi 4-bit biner yang dipakai. Dari Tabel 7.15 dapat dilihat bahwa kombinasi nilai-nilai

- 1010 (nilai desimal = 10),
- 1011 (nilai desimal = 11),
- 1100 (nilai desimal = 12),
- 1101 (nilai desimal = 13),
- 1110 (nilai desimal = 14),
- 1111 (nilai desimal = 15)

tidak digunakan untuk merepresentasikan angka desimal (diabaikan) karena nilai desimalnya melebihi jumlah angka desimal yang ada (10 buah, yaitu 0 sampai 9). Dalam hal ini, kita tidak mempedulikan nilai fungsi untuk keenam kombinasi 4-bit yang terakhir (nilai fungsi dinyatakan dengan X).

Tabel 7.15

w	x	y	z	Desimal
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	X
1	0	1	1	X
1	1	0	0	X
1	1	0	1	X
1	1	1	0	X
1	1	1	1	X

Dalam menyederhanakan Peta Karnaugh yang mengandung keadaan *don't care*, ada dua hal penting yang dijadikan pegangan. Pertama, kita anggap semua nilai *don't care* (disimbolkan dengan X) sama dengan 1 dan kemudian membentuk kelompok sebesar mungkin yang melibatkan angka 1 termasuk tanda X tersebut. Kedua, semua nilai X yang tidak termasuk dalam kelompok tersebut kita anggap bernilai 0. Dengan cara ini, keadaan-keadaan X telah dimanfaatkan semaksimal mungkin; dan kita boleh melakukannya secara bebas sebab keadaan *don't care* dapat diperlakukan sebagai 0 atau 1, terserah pada kita. Contoh 7.42 mengilustrasikan penyederhanaan Pea Karnaugh yang mengandung keadaan *don't care*.

Contoh 7.43

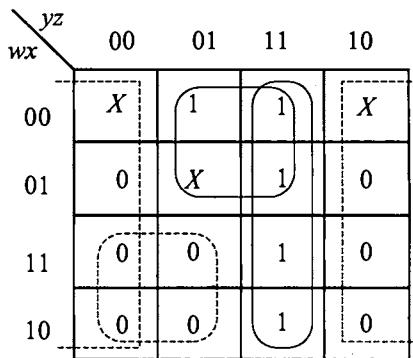
Minimisasi fungsi Boolean berikut (hasil penyederhanaan dalam bentuk baku SOP dan bentuk baku POS):

$$f(w, x, y, z) = \Sigma (1, 3, 7, 11, 15)$$

dengan kondisi *don't care* adalah $d(w, x, y, z) = \Sigma (0, 2, 5)$

Penyelesaian:

Peta Karnaugh dari fungsi tersebut adalah:



Hasil penyederhanaan dalam bentuk SOP

$$f(w, x, y, z) = yz + w'z \quad (\text{SOP}) \quad (\text{berdasarkan kelompok garis penuh})$$

dan hasil penyederhanaan dalam bentuk baku POS adalah

$$f(w, x, y, z) = z(w' + y) \quad (\text{POS}) \quad (\text{berdasarkan kelompok garis putus-putus})$$

Contoh 7.44

Diberikan Tabel 7.16 yang merepresentasikan sebuah fungsi Boolean, f . Minimisasi fungsi f sesederhana mungkin.

Tabel 7.16

w	x	y	z	$f(w, x, y, z)$
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	X
1	0	0	1	X
1	0	1	0	X
1	0	1	1	X
1	1	0	0	X
1	1	0	1	X
1	1	1	0	X
1	1	1	1	X

Penyelesaian:

Peta Karnaugh dari fungsi tersebut adalah:

$wx \backslash yz$	00	01	11	10
00	1	0	1	0
01	1	1	1	0
11	X	X	X	X
10	X	0	X	X

Hasil penyederhanaan: $f(w, x, y, z) = xz + y'z' + yz$ ■

7.14 Penyederhanaan Rangkaian Logika

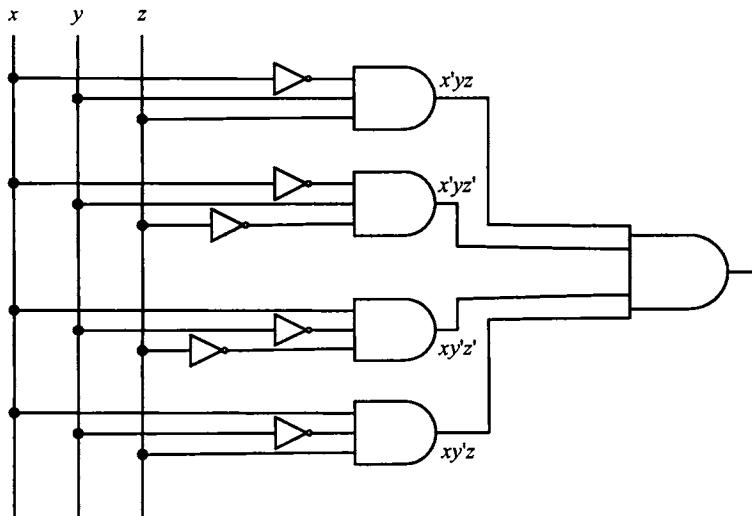
Teknik minimisasi fungsi Boolean dengan Peta Karnaugh mempunyai terapan yang sangat penting dalam menyederhanakan rangkaian logika. Penyederhanaan rangkaian dapat mengurangi jumlah gerbang logika yang digunakan, bahkan dapat mengurangi jumlah kawat masukan. Contoh-contoh di bawah ini memberikan ilustrasi penyederhanaan rangkaian logika.

Contoh 7.45

Minimisasi fungsi Boolean $f(x, y, z) = x'yz + x'y'z' + xy'z' + xy'z$. Gambarkan rangkaian logikanya.

Penyelesaian:

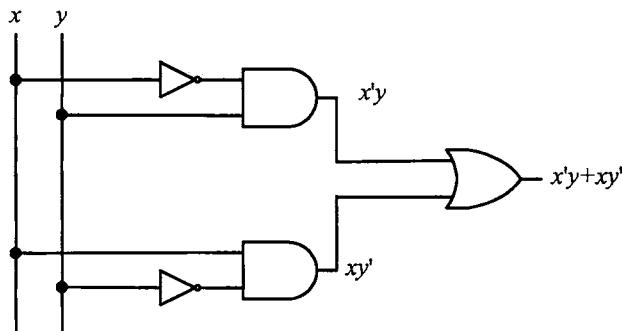
Rangkaian logika fungsi $f(x, y, z)$ sebelum diminimisasikan adalah seperti di bawah ini:



Minimisasi dengan Peta Karnaugh adalah sebagai berikut:

		yz	00	01	11	10
		x	0	1	1	1
y	z	0	1	0	(1)	(1)
		1	(1)	(1)	0	0

Fungsi Boolean hasil minimisasi adalah $f(x, y, z) = x'y + xy'$, dan rangkaian logikanya ditunjukkan berikut ini.



Rangkaian logika hasil minimisasi fungsi hanya membutuhkan dua buah kawat masukan, x dan y , dua buah gerbang AND, sebuah gerbang NOT, dan sebuah gerbang OR. ■

7.15 Metode Quine-McCluskey

Metode peta Karnaugh hanya cocok digunakan jika fungsi Boolean mempunyai jumlah peubah paling banyak 6 buah. Jika jumlah peubah yang terlibat pada suatu fungsi Boolean lebih dari 6 buah maka penggunaan peta Karnaugh menjadi semakin rumit, sebab ukuran peta bertambah besar. Selain itu, metode peta Karnaugh lebih sulit diprogram dengan komputer karena diperlukan pengamatan visual untuk mengidentifikasi *minterm-minterm* yang akan dikelompokkan. Untuk itu diperlukan metode penyederhanaan yang lain yang dapat diprogram dan dapat digunakan untuk fungsi Boolean dengan sembarang jumlah peubah. Metode alternatif tersebut adalah metode Quine-McCluskey yang dikembangkan oleh W.V. Quine dan E.J. McCluskey pada tahun 1950.

Langkah-langkah metode Quine-McCluskey untuk menyederhanakan ekspresi Boolean dalam bentuk SOP adalah sebagai berikut:

1. Nyatakan tiap *minterm* dalam n peubah menjadi *string* bit yang panjangnya n , yang dalam hal ini peubah komplemen dinyatakan dengan ‘0’, peubah yang bukan komplemen dengan ‘1’.
2. Kelompokkan tiap *minterm* berdasarkan jumlah ‘1’ yang dimilikinya.
3. Kombinasikan *minterm* dalam n peubah dengan kelompok lain yang jumlah ‘1’-nya berbeda satu, sehingga diperoleh **bentuk prima** (*prime-implicant*) yang terdiri dari $n - 1$ peubah. *Minterm* yang dikombinasikan diberi tanda “ \checkmark ”.
4. Kombinasikan *minterm* dalam $n - 1$ peubah dengan kelompok lain yang jumlah ‘1’-nya berbeda satu, sehingga diperoleh bentuk prima yang terdiri dari $n - 2$ peubah.
5. Teruskan langkah 4 sampai diperoleh bentuk prima yang sesederhana mungkin.
6. Ambil semua bentuk prima yang tidak bertanda “ \checkmark ”. Buatlah tabel baru yang memperlihatkan *minterm* dari ekspresi Boolean semula yang dicakup oleh bentuk prima tersebut (tandai dengan “ \times ”). Setiap *minterm* harus dicakup oleh paling sedikit satu buah bentuk prima.
7. Pilih bentuk prima yang memiliki jumlah literal paling sedikit namun mencakup sebanyak mungkin *minterm* dari ekspresi Boolean semula. Hal ini dapat dilakukan dengan cara berikut:
 - a. Tandai kolom-kolom yang mempunyai satu buah tanda “ \times ” dengan tanda “ $*$ ”, lalu beri tanda “ \checkmark ” di sebelah kiri bentuk prima yang berasosiasi dengan tanda “ $*$ ” tersebut. Bentuk prima ini telah dipilih untuk fungsi Boolean sederhana.
 - b. Untuk setiap bentuk prima yang telah ditandai dengan “ \checkmark ”, beri tanda *minterm* yang dicakup oleh bentuk prima tersebut dengan tanda “ \checkmark ” (di baris bawah setelah “ $*$ ”).
 - c. Periksa apakah masih ada *minterm* yang belum dicakup oleh bentuk prima terpilih. Jika ada, pilih dari bentuk prima yang tersisa yang mencakup sebanyak mungkin *minterm* tersebut. Beri tanda “ \checkmark ” bentuk prima yang dipilih itu serta *minterm* yang dicakupnya.
 - d. Ulangi langkah c sampai seluruh *minterm* sudah dicakup oleh semua bentuk prima.

Metode Quine McCluskey biasanya digunakan untuk menyederhanakan fungsi Boolean yang ekspresinya dalam bentuk SOP, namun metode ini dapat dimodifikasi sehingga juga dapat digunakan untuk ekspresi dalam bentuk POS.

Contoh 7.46 dan Contoh 7.47 berikut mengilustrasikan penggunaan metode McCluskey untuk menyederhanakan fungsi Boolean dalam bentuk SOP.

Contoh 7.46

Sederhanakan fungsi Boolean $f(w, x, y, z) = \Sigma (0, 1, 2, 8, 10, 11, 14, 15)$.

Penyelesaian:

(i) Langkah 1 sampai 5:

(a)		(b)		(c)	
term	w x y z	term	w x y z	term	w x y z
0	0 0 0 0 ✓	0,1	0 0 0 -	0,2,8,10	- 0 - 0
		0,2	0 0 - 0 ✓	0,8,2,10	- 0 - 0
1	0 0 0 1 ✓	0,8	- 0 0 0 ✓		
2	0 0 1 0 ✓			10,11,14,15	1 - 1 -
8	1 0 0 0 ✓	2,10	- 0 1 0 ✓	10,14,11,15	1 - 1 -
		8,10	1 0 - 0 ✓		
10	1 0 1 0 ✓				
		10,11	1 0 1 - ✓		
11	1 0 1 1 ✓	10,14	1 - 1 0 ✓		
14	1 1 1 0 ✓				
		11,15	1 - 1 1 ✓		
15	1 1 1 1 ✓	14,15	1 1 1 - ✓		

(ii) Langkah 6 dan 7:

Bentuk prima	minterm							
	0	1	2	8	10	11	14	15
✓ 0,1	x	x						
✓ 0,2,8,10	x		x	x	x			
✓ 10,11,14,15				x	x	x	x	
	*	*	*	*	*	*	*	*
	✓	✓	✓	✓	✓	✓	✓	✓

Bentuk prima yang terpilih adalah:

- 0,1 yang bersesuaian dengan term $w'x'y$
 0, 2, 8, 10 yang bersesuaian dengan term $x'z'$
 10, 11, 14, 15 yang bersesuaian dengan term wy

Semua bentuk prima di atas sudah mencakup semua minterm dari fungsi Boolean semula. Dengan demikian, fungsi Boolean hasil penyederhanaan adalah $f(w, x, y, z) = w'x'y' + x'z' + wy$. ■

Contoh 7.46 di atas kurang begitu bagus dalam memberikan ilustrasi metode Quine-McCluskey. Contoh 7.47 di bawah ini dapat memberikan gambaran metode untuk kasus yang lebih umum.

Contoh 7.47

Sederhanakan fungsi Boolean $f(w, x, y, z) = \Sigma (1, 4, 6, 7, 8, 9, 10, 11, 15)$

Penyelesaian:

(i) Langkah 1 sampai 5:

(a)					(b)					(c)				
term	w	x	y	z	term	w	x	y	z	term	w	x	y	z
1	0	0	0	1	✓	1,9	-	0	0	1	8,9,10,11	1	0	--
4	0	1	0	0	✓	4,6	0	1	-	0	8,10,9,11	1	0	--
8	1	0	0	0	✓	8,9	1	0	0	-	✓			
						8,10	1	0	-	0	✓			
6	0	1	1	0	✓									
9	1	0	0	1	✓	6,7	0	1	1	-				
10	1	0	1	0	✓	9,11	1	0	-	1	✓			
						10,11	1	0	1	-	✓			
7	0	1	1	1	✓									
11	1	0	1	1	✓	7,15	-	1	1	1				
						11,15	1	-	1	1				
15	1	1	1	1	✓									

(ii) Langkah 6 dan 7

Bentuk prima	minterm								
	1	4	6	7	8	9	10	11	15
✓ 1,9	✗					✗			
✓ 4,6		✗	✗						
6,7			✗	✗					
7,15				✗					✗
11,15							✗	✗	
✓ 8,9,10,11					✗	✗	✗	✗	
	*	*		*	*				
	✓	✓	✓	✓	✓	✓	✓	✓	

Sampai tahap ini, masih ada dua *minterm* yang belum tercakup dalam bentuk prima terpilih, yaitu 7 dan 15. Bentuk prima yang tersisa (tidak terpilih) adalah (6,7), (7,15), dan (11, 15). Dari ketiga kandidat ini, kita pilih bentuk prima (7,15) karena bentuk prima ini mencakup *minterm* 7 dan 15 sekaligus.

Bentuk prima	minterm								
	1	4	6	7	8	9	10	11	15
✓ 1,9		✗				✗			
✓ 4,6			✗	✗					
6,7				✗	✗				
✓ 7,15					✗				✗
11,15							✗	✗	
✓ 8,9,10,11					✗	✗	✗	✗	
	*	*	*	*	*	*	*	*	
	✓	✓	✓	✓	✓	✓	✓	✓	✓

Sekarang, semua *minterm* sudah tercakup dalam bentuk prima terpilih. Bentuk prima yang terpilih adalah:

- | | | |
|-----------|------------------------------|---------|
| 1,9 | yang bersesuaian dengan term | $x'y'z$ |
| 4,6 | yang bersesuaian dengan term | $w'xz'$ |
| 7,15 | yang bersesuaian dengan term | xyz |
| 8,9,10,11 | yang bersesuaian dengan term | wx' |

Dengan demikian, fungsi Boolean hasil penyederhanaan adalah $f(w, x, y, z) = x'y'z + w'xz' + xyz + wx'$. ■

7.16 Ragam Soal dan Penyelesaian

Contoh 7.48

Majority gate merupakan sebuah rangkaian digital yang keluarannya sama dengan 1 jika mayoritas masukannya bernilai 1. Keluaran sama dengan 0 jika tidak memenuhi hal tersebut di atas. Dengan bantuan tabel kebenaran, carilah fungsi Boolean yang diimplementasikan dengan 3-input *majority gate*. Sederhanakan fungsinya, lalu gambarkan rangkaian logikanya.

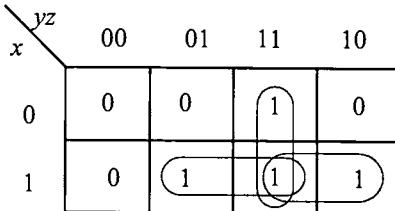
Penyelesaian:

Tabel kebenaran *majority gate* dari tiga peubah diperlihatkan pada Tabel 7.17.

Tabel 7.17

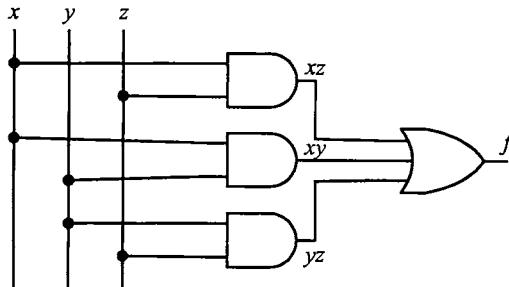
x	y	z	$f(x, y, z)$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Peta Karnaugh dari tabel tersebut dan rangkaian logikanya ditunjukkan berikut ini:



$$f(x, y, z) = xz + xy + yz$$

Rangkaian logika:

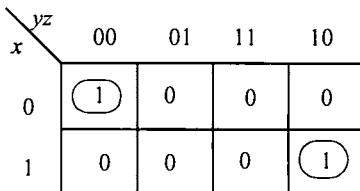


Contoh 7.49

Implementasikan fungsi $f(x, y, z) = \Sigma (0, 6)$ dan hanya dengan gerbang NAND saja.

Penyelesaian:

Peta Karnaugh dari fungsi tersebut adalah:



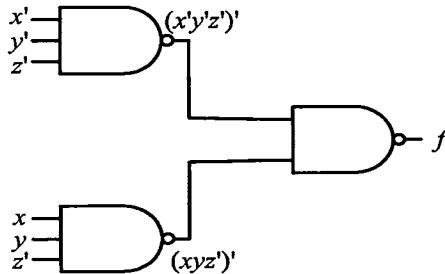
Fungsi hasil penyederhanaan: $f(x, y, z) = x'y'z' + xyz'$. Dengan menggunakan hukum De Morgan, fungsi f dapat ditulis sebagai

$$f(x, y, z) = ((x'y'z')'(xyz'))'$$

(periksalah bahwa $((x'y'z')'(xyz'))'$ dapat dikembalikan menjadi $x'y'z' + xyz'$, yaitu

$$\begin{aligned} ((x'y'z')'(xyz'))' &= ((x+y+z)(x'+y'+z))' \\ &= (x+y+z)' + (x'+y'+z)' \\ &= x'y'z' + xyz' \end{aligned}$$

Rangkaian logika untuk $f(x, y, z) = ((x'y'z')'(xyz'))'$ dengan hanya menggunakan gerbang NAND adalah seperti di bawah ini:



Contoh 7.50

Gunakan Peta Karnaugh untuk merancang rangkaian logika yang dapat menentukan apakah sebuah angka desimal yang direpresentasikan dalam bit biner merupakan bilangan genap atau bukan (yaitu, memberikan nilai 1 jika genap dan 0 jika tidak).

Penyelesaian:

Angka desimal: 0 .. 9 (direpresentasikan dalam 4 bit biner, misalkan $a_0a_1a_2a_3$). Fungsi $f(a_0, a_1, a_2, a_3)$ bernilai 1 jika representasi desimal dari $a_0a_1a_2a_3$ menyatakan bilangan genap, dan bernilai 0 jika tidak (Tabel 7.18).

Peta Karnaugh:

a_2a_3	00	01	11	10
00	1	0	0	1
01	1	0	0	1
11	X	X	X	X
10	1	0	X	X

Fungsi hasil penyederhanaan: $f(a_0, a_1, a_2, a_3) = a_3'$

Rangkaian logikanya:



Tabel 7.18

a_0	a_1	a_2	a_3	Desimal	$f(a_0, a_1, a_2, a_3)$
0	0	0	0	0	1
0	0	0	1	1	0
0	0	1	0	2	1
0	0	1	1	3	0
0	1	0	0	4	1
0	1	0	1	5	0
0	1	1	0	6	1
0	1	1	1	7	0
1	0	0	0	8	1
1	0	0	1	9	0
1	0	1	0	10	X
1	0	1	1	11	X
1	1	0	0	12	X
1	1	0	1	13	X
1	1	1	0	14	X
1	1	1	1	15	X

Contoh 7.51

Sebuah instruksi dalam sebuah program adalah

```
if A > B then writeln(A) else writeln(B);
```

Nilai A dan B yang dibandingkan masing-masing panjangnya dua bit (misalkan a_1a_2 dan b_1b_2).

- Buatlah rangkaian logika (yang sudah disederhanakan tentunya) yang menghasilkan keluaran 1 jika $A > B$ atau 0 jika tidak.
- Gambarkan kembali rangkaian logikanya jika hanya menggunakan gerbang $NAND$ saja (petunjuk: gunakan hukum de Morgan)

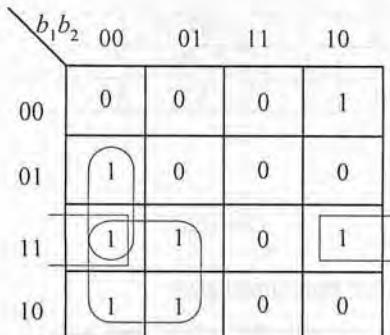
Penyelesaian:

- Tabel kebenaran (Tabel 7.19) menyatakan bahwa fungsi f bernilai 1 jika A (a_1a_2) lebih besar dari B (b_1b_2) dan bernilai 0 jika tidak.

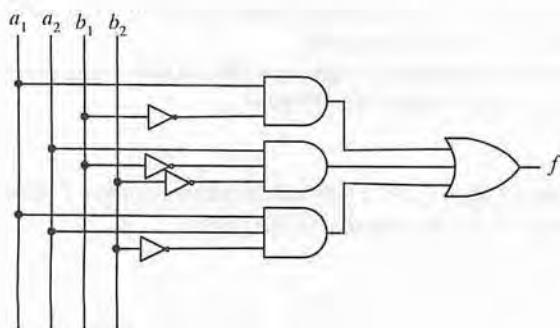
Tabel 7.19

Desimal		Biner				$f(a_1, a_2, b_1, b_2)$
A	B	a_1	a_2	b_1	b_2	
0	0	0	0	0	0	0
0	1	0	0	0	1	0
0	2	0	0	1	0	0
0	3	0	0	1	1	0
1	0	0	1	0	0	1
1	1	0	1	0	1	0
1	2	0	1	1	0	0
1	3	0	1	1	1	0
2	0	1	0	0	0	1
2	1	1	0	0	1	1
2	2	1	0	1	0	0
2	3	1	0	1	1	0
3	0	1	1	0	0	1
3	1	1	1	0	1	1
3	2	1	1	1	0	1
3	3	1	1	1	1	0

Peta Karnaugh:

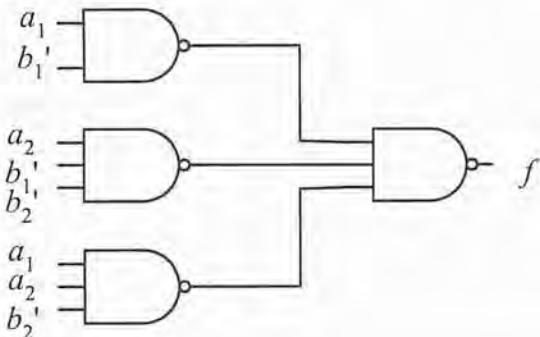
Fungsi Boolean: $f(a_1, a_2, b_1, b_2) = a_1 b_1' + a_2 b_1' b_2' + a_1 a_2 b_2'$

Rangkaian logikanya:



$$(b) f(a_1, a_2, b_1, b_2) = a_1 b_1' + a_2 b_1' b_2' + a_1 a_2 b_2' \\ = ((a_1 b_1')' (a_2 b_1' b_2')' (a_1 a_2 b_2')')' \quad (\text{hukum De Morgan})$$

Rangkaian logika:



Contoh 7.52

Berbagai sistem digital menggunakan kode *binary coded decimal (BCD)*. Misalkan 4-bit masukan *BCD* adalah $wxyz$ dan kode *Excess-3* adalah $f_1f_2f_3f_4$. Diberikan Tabel 7.20 untuk konversi *BCD* ke kode *Excess-3* sebagai berikut:

Tabel 7.20

Desimal	Masukan BCD				Keluaran kode Excess-3			
	w	x	y	z	$f_1(w, x, y, z)$	$f_2(w, x, y, z)$	$f_3(w, x, y, z)$	$f_4(w, x, y, z)$
0	0	0	0	0	0	0	1	1
1	0	0	0	1	0	1	0	0
2	0	0	1	0	0	1	0	1
3	0	0	1	1	0	1	1	0
4	0	1	0	0	0	1	1	1
5	0	1	0	1	1	0	0	0
6	0	1	1	0	1	0	0	1
7	0	1	1	1	1	0	1	0
8	1	0	0	0	1	0	1	1
9	1	0	0	1	1	1	0	0
10	1	0	1	0	X	X	X	X
11	1	0	1	1	X	X	X	X
12	1	1	0	0	X	X	X	X
13	1	1	0	1	X	X	X	X
14	1	1	1	0	X	X	X	X
15	1	1	1	1	X	X	X	X

Masing-masing Peta Karnaugh dari fungsi f_1, f_2, f_3 , dan f_4 adalah sebagai berikut:

(a) $f_1(w, x, y, z)$

		yz	00	01	11	10
		00	0	0	0	0
		01	0	1	1	1
		11	X	X	X	X
		10	1	1	X	X

Fungsi minimasi: $f_1(w, x, y, z) = w + xz + xy = w + x(y + z)$

(b) $f_2(w, x, y, z)$

		yz	00	01	11	10
		00	0	1	1	1
		01	1	0	0	0
		11	X	X	X	X
		10	0	1	X	X

Fungsi minimasi: $f_2(w, x, y, z) = xy'z' + x'z + x'y = xy'z' + x'(y + z)$

(c) $f_3(w, x, y, z)$

		yz	00	01	11	10
		00	1	0	1	0
		01	1	0	1	0
		11	X	X	X	X
		10	X	0	X	X

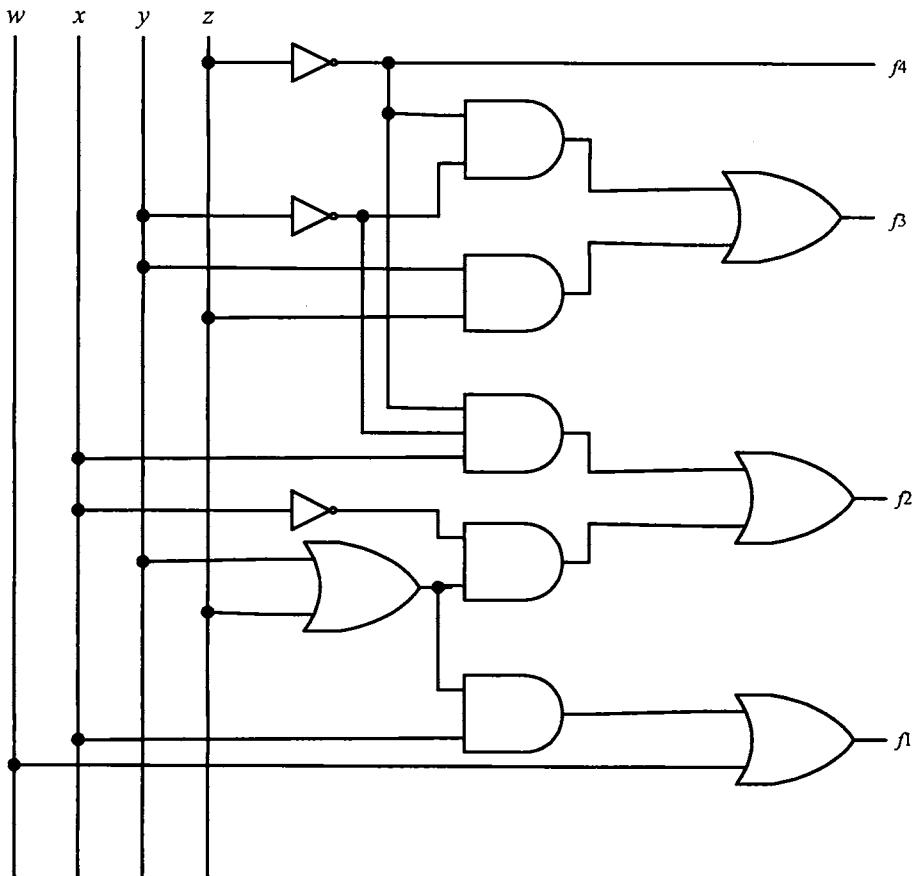
Fungsi minimasi: $f_3(w, x, y, z) = y'z' + yz$

(d) $f_4(w, x, y, z)$

yz	00	01	11	10
00	1	0	0	1
01	1	0	0	1
11	X	X	X	X
10	1	0	X	X

Fungsi minimasi: $f_4(w, x, y, z) = z'$

Rangkaian logikanya:



Contoh 7.53

Di dalam unit aritmetika komputer (*Arithmetic Logical Unit – ALU*) terdapat rangkaian penjumlahah (*adder*). Salah satu jenis rangkaian penjumlahah adalah penjumlahah-paruh (*half adder*). Rangkaian ini menjumlahkan 2 bit masukan dengan keluarannya adalah *SUM* (jumlah) dan *CARRY* (pindahan).

Tabel kebenaran dari penjumlahah-paruh ditunjukkan pada Tabel 7.21.

Tabel 7.21

x	y	SUM	CARRY
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Peta Karnaugh untuk *SUM*:

	y	0	1
x	0	0	1
	1	1	0

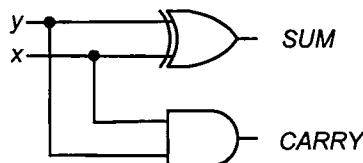
$$SUM = x'y + xy' = x \oplus y$$

Peta Karnaugh untuk *CARRY*:

	y	0	1
x	0	0	0
	1	0	1

$$CARRY = xy$$

Rangkaian logika penjumlahah-paruh:



Catatan:

Rangkaian penjumlahah-penuh (*full adder*) adalah rangkaian yang menjumlahkan 3 bit masukan. Sebagai latihan, coba anda buat tabel kebenaran, Peta Karnaugh, dan rangkaian logika dari penjumlahah-penuh dengan 3 bit masukan.

Contoh 7.54

Buatlah rangkaian logika yang menerima masukan dua-bit dan menghasilkan keluaran berupa kudrat dari masukan. Sebagai contoh, jika masukannya 11 (3 dalam sistem desimal), maka keluarannya adalah 1001 (9 dalam sistem desimal).

Penyelesaian:

Misalkan 2-bit masukan kita simbolkan dengan xy , dan kuadratnya (4-bit) kita simbolkan dengan $abcd$. Tabel kebenaran yang merepresentasikan rangkaian pengkuadrat ditunjukkan pada Tabel 7.22.

Tabel 7.22

Masukan		Keluaran			
w	x	a	b	c	d
0	0	0	0	0	0
0	1	0	0	0	1
1	0	0	1	0	0
1	1	1	0	0	1

Peta Karnaugh untuk masing-masing komponen keluaran:

x	y	0	1
0	0	0	0
1	0	1	1

$$a(x, y) = xy$$

x	y	0	1
0	0	0	0
1	0	1	0

$$b(x, y) = xy'$$

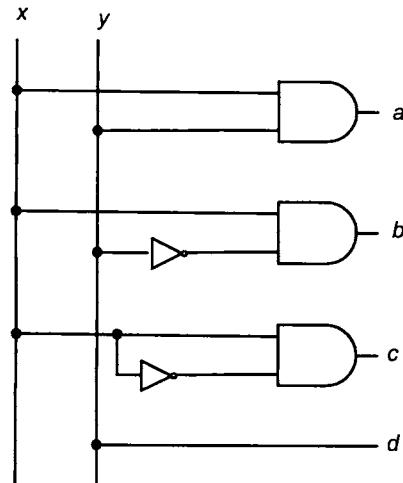
x	y	0	1
0	0	0	0
1	0	0	0

$$c(x, y) = 0 = xx'$$

x	y	0	1
0	0	0	1
1	0	0	1

$$d(x, y) = y$$

Rangkaian logikanya pengkuadrat 2-bit biner:



Soal Latihan

1. Diketahui himpunan B dengan tiga buah nilai $\{0, 1, 2\}$ dan dua buah operator, $+$ dan \bullet . Kaidah operasi dengan operator $+$ dan \bullet didefinisikan pada tabel di bawah ini:

$+$	0	1	2
0	0	0	0
1	0	1	1
2	0	1	2

\bullet	0	1	2
0	0	1	2
1	1	1	2
2	2	2	2

- (a) Dari keempat aksioma dasar (komutatif, distributif, identitas, dan komplemen), aksioma manakah yang dipenuhi oleh kedua tabel di atas? Apakah elemen identitas untuk masing-masing operator $+$ dan \bullet tersebut?
- (b) Apakah himpunan B dengan dua buah operator di atas membentuk aljabar Boolean? Jelaskan jawaban anda!
2. Buktikan hukum asosiatif $a + (b + c) = (a + b) + c$ dan $a \cdot (b \cdot c) = (a \cdot b) \cdot c$.
3. Nyatakan fungsi Boolean $f(x,y,z) = x'(x + y' + z')$ hanya dengan menggunakan operator $+$ dan komplemen ('') saja.
4. Nyatakan $f(a,b,c) = ((ab)' c)' ((a' + c)(b' + c'))'$ dalam bentuk baku SOP dan bentuk kanonik SOP.
5. Perlihatkan bahwa dual dari ekspresi XOR berikut: $(x \oplus y)$ sama dengan komplemennya (pertunjuk: nyatakan ekspresi XOR dalam operator $+$ dan \cdot).
6. Carilah komplemen dari fungsi $f(w, x, y, z) = x'z + w'xy' + wyz + w'xy$
7. Sederhanakan fungsi Boolean di bawah ini secara aljabar (menggunakan aksioma/ teorema):
- (a) $xy + x'z + yz$
(b) $(x + y)(x' + z)(y + z)$ (dengan memanfatkan prinsip dualitas dari fungsi (a) di atas)
7. Diketahui fungsi Boolean berikut:

$$f(w, x, y, z) = \prod (0, 1, 2, 3, 7, 11, 13)$$
$$d(w, x, y, z) = \sum (5, 9, 14, 15)$$

dengan $d(w, x, y, z)$ adalah fungsi *don't care*. Minimisasi fungsi tersebut di atas dengan menggunakan metode Peta Karnaugh. Setelah itu tuliskan fungsi sederhana itu dalam bentuk baku SOP dan bentuk baku POS.

8. Temukan fungsi Boolean yang paling sederhana dalam bentuk *product of sum* (POS) dari fungsi berikut:

$$f(w, x, y, z) = \prod (0, 1, 2, 3, 7, 8, 11, 13) \text{ dan } d(w, x, y, z) = \sum (5, 9, 14, 15)$$

9. Diberikan dua buah fungsi Boolean f dan g . Maka, fungsi $h = fg$ diperoleh dengan meng-AND-kan dua buah fungsi, yang hasilnya adalah *minterm* bersama yang terdapat baik pada f maupun pada g . Jika

$$f = wxy' + y'z + w'yz' + x'yz'$$

dan

$$g = (w + x + y' + z')(x' + y' + z)(w' + y + z')$$

maka, dengan menggunakan peta Karnaugh, temukan bentuk yang paling sederhana dari $h = fg$.

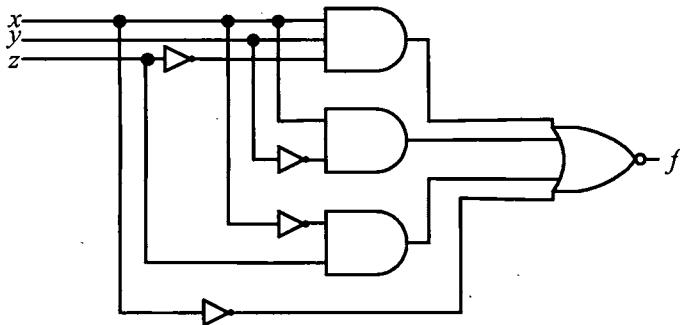
10. Minimisasi fungsi-fungsi Boolean berikut dengan metode peta Karnaugh, dalam bentuk baku SOP dan bentuk baku POS:

- (a) $f(x, y, z) = \Sigma (2, 3, 6, 7)$
- (b) $f(x, y, z) = xy + x'y'z' + x'yz'$
- (c) $f(w, x, y, z) = \Sigma (4, 6, 7, 15)$
- (d) $f(w, x, y, z) = \prod (0, 1, 2, 6, 8, 9, 12)$.
- (e) Diberikan tabel kebenaran:

x	y	z	$f(x, y, z)$
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

11. Gunakan Peta Karnaugh untuk membuat rangkaian logika yang menerima masukan berupa kode biner dari suatu digit desimal dan menghasilkan keluaran 1 jika digit yang berkoresponden dengan masukan tersebut tidak habis dibagi 3. Fungsi minimasi dalam bentuk baku SOP.
12. Rancang dan gambarkan rangkaian kombinasional yang menerima masukan bilangan 3-bit dan membangkitkan keluaran bilangan biner yang sama dengan kuadrat dari bilangan masukannya.

13. Diberikan gambar rangkaian logika seperti di bawah ini:



- (a) Tuliskan fungsi Boolean $f(x, y, z)$ yang merepresentasikan rangkaian di atas.
- (b) Tuliskan fungsi Boolean $f(x, y, z)$ dalam bentuk kanonik POS.
- (c) Sederhanakan rangkaian di atas dengan menggunakan Peta Karnaugh, lalu gambarkan rangkaian hasil penyederhanaan.
14. Rancanglah rangkaian logika untuk menghitung koin uang logam yang dimasukkan pada pengumpul bea otomatis sebagai pembayar jasa tol. Mesin penghitung ditempatkan pada gerbang tol. Tarif tol adalah 15 sen. Mesin hanya dapat menerima koin 5 sen dan koin 10 sen. Bila mesin telah menerima sejumlah koin senilai 15 sen, maka lampu hijau menyala (artinya boleh lewat gerbang tol), dan jika belum 15 sen, lampu merah tetap menyala (artinya belum boleh lewat gerbang tol). Gambarkan rangkaian logika yang dimaksud!

Catatan:

Pembayaran dapat dilakukan dengan koin 5 sen saja atau koin 10 sen saja atau kombinasi keduanya. Karena biaya tol 15 sen, maka jumlah koin 5 sen yang digunakan maksimal 3 buah (= 15 sen), jumlah koin 10 sen yang digunakan maksimal 2 buah (= 20 sen). Di luar jumlah koin itu, keluaran mesin tidak penting nilainya (kondisi don't care). Anda terlebih dahulu harus menentukan berapa banyak peubah (variable) Boolean yang dibutuhkan.

15. (a) Sederhanakan fungsi Boolean f berikut dalam bentuk *product of sum* dengan menggunakan fungsi *don't care* (disimbolkan dengan d):

$$f(w, x, y, z) = w'x'z' + w'yz + w'xy \\ d(w, x, y, z) = w'xy'z + wyz + wx'z'$$

- (b) Gambarkan rangkaian logika fungsi yang telah disederhanakan pada jawaban (a) di atas dengan hanya gerbang NOT dan NOR saja.

16. Sederhanakan fungsi Boolean berikut dengan menggunakan metode Quine-Mc Cluskey:

$$f(v, w, x, y, z) = \sum (9, 11, 13, 15, 17, 21, 25, 27, 29, 31)$$

17. Minimisasi fungsi $f(x, y, z) = \sum(0, 2, 4, 5, 6)$ dengan metode Quine-McCluskey, lalu gambarkan hasil penyederhanaannya hanya dengan menggunakan gerbang *NAND* saja.

18. Sederhanakan dan implementasikan fungsi Boolean berikut dalam rangkaian digital:

- (a) menggunakan sembarang gerbang

$$f(x, y) = xy' + x'y$$

- (b) menggunakan hanya gerbang *NAND*

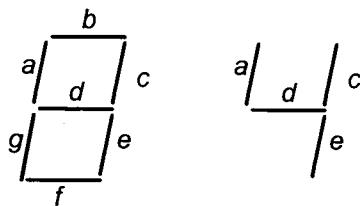
$$f(v, w, x, y, z) = vx' + vxz + vxz' + v'xy' + v'y'z'$$

19. Carilah komplemen dari fungsi Boolean berikut:

$$f(w, x, y, z) = x'z + w'xy' + wyz + w'xy$$

20. Gambarkan rangkaian pensaklaran yang menyatakan ekspresi Boolean $xy + xy'z + y(x' + z) + y'z'$

21. Sebuah Peraga angka digital disusun oleh tujuh buah segmen (selanjutnya disebut *dekoder tujuh-segmen*).



dekoder 7-segmen

angka 4

Piranti tersebut mengubah masukan 4-bit BCD menjadi keluaran yang dapat menunjukkan angka desimal yang dinyatakannya (misalnya, jika masukan adalah 0100 (angka 4 dalam desimal), maka batang/segmen yang menyala adalah a, d, c, dan e). Tulislah fungsi Boolean untuk tiap segmen, dan gambarkan rangkaian kombinasionalnya.

22. Sebuah instruksi dalam sebuah algoritma adalah

if A ≠ B then A ← A + 1 else A ← A + 2

- (b) Nilai A dan B yang dibandingkan masing-masing panjangnya dua bit (misalkan a_1a_2 dan b_1b_2). Buatlah rangkaian logika (yang sudah disederhanakan tentunya) yang menghasilkan keluaran 1 jika $A \neq B$ atau 0 jika tidak.
- (c) Gambarkan kembali rangkaian logikanya jika hanya menggunakan gerbang NOR saja (petunjuk: gunakan hukum de Morgan)

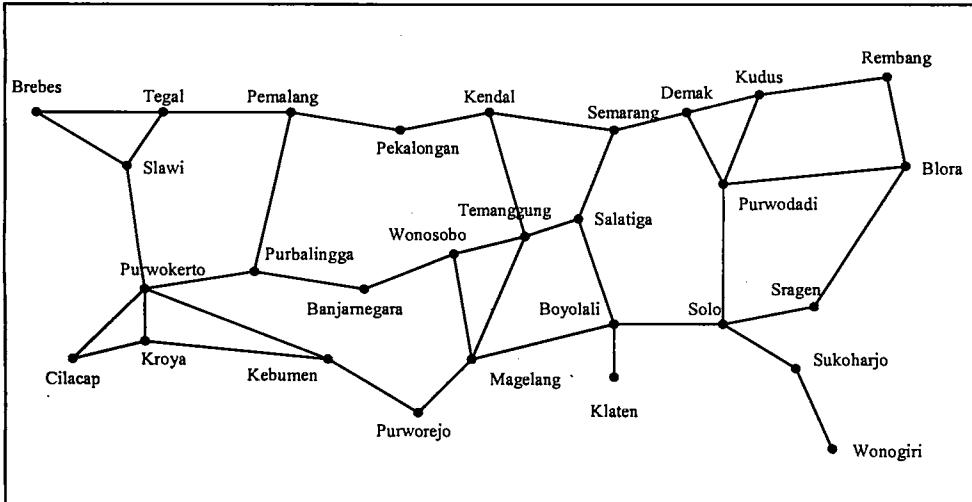
BAB 8

Graf

Jangan ikuti kemana jalan menuju,
tetapi buatlah jalan sendiri dan tinggalkan jejak¹
(Anonim)

Teori graf merupakan pokok bahasan yang sudah tua usianya namun memiliki banyak terapan sampai saat ini. Graf digunakan untuk merepresentasikan objek-objek diskrit dan hubungan antara objek-objek tersebut. Representasi visual dari graf adalah dengan menyatakan objek dinyatakan sebagai noktah, bulatan, atau titik, sedangkan hubungan antara objek dinyatakan dengan garis. Sebagai contoh, Gambar 8.1 adalah sebuah peta jaringan jalan raya yang menghubungkan sejumlah kota di Provinsi Jawa Tengah. Sesungguhnya peta tersebut adalah sebuah graf, yang dalam hal ini kota dinyatakan sebagai bulatan sedangkan jalan dinyatakan sebagai garis. Dengan diberikannya peta tersebut, kita dapat mengetahui apakah ada lintasan jalan antara dua buah kota. Selain itu, bila panjang jalan kereta api antara dua buah kota bertetangga diketahui, kita juga dapat menentukan rute perjalanan yang tersingkat dari kota A ke kota B. Masih banyak pertanyaan lain yang dapat kita munculkan berkenaan dengan graf. Sebelum kita mempelajari teori graf lebih lanjut, ada baiknya kita melakukan kilas balik menelusuri sejarah graf yang dimulai pada Abad 19.

¹ Terjemahan bebas dari kalimat: "Do not follow where the path may lead. Go, instead, where there is no path and leave a trail".



Gambar 8.1 Jaringan jalan raya di Provinsi Jawa Tengah

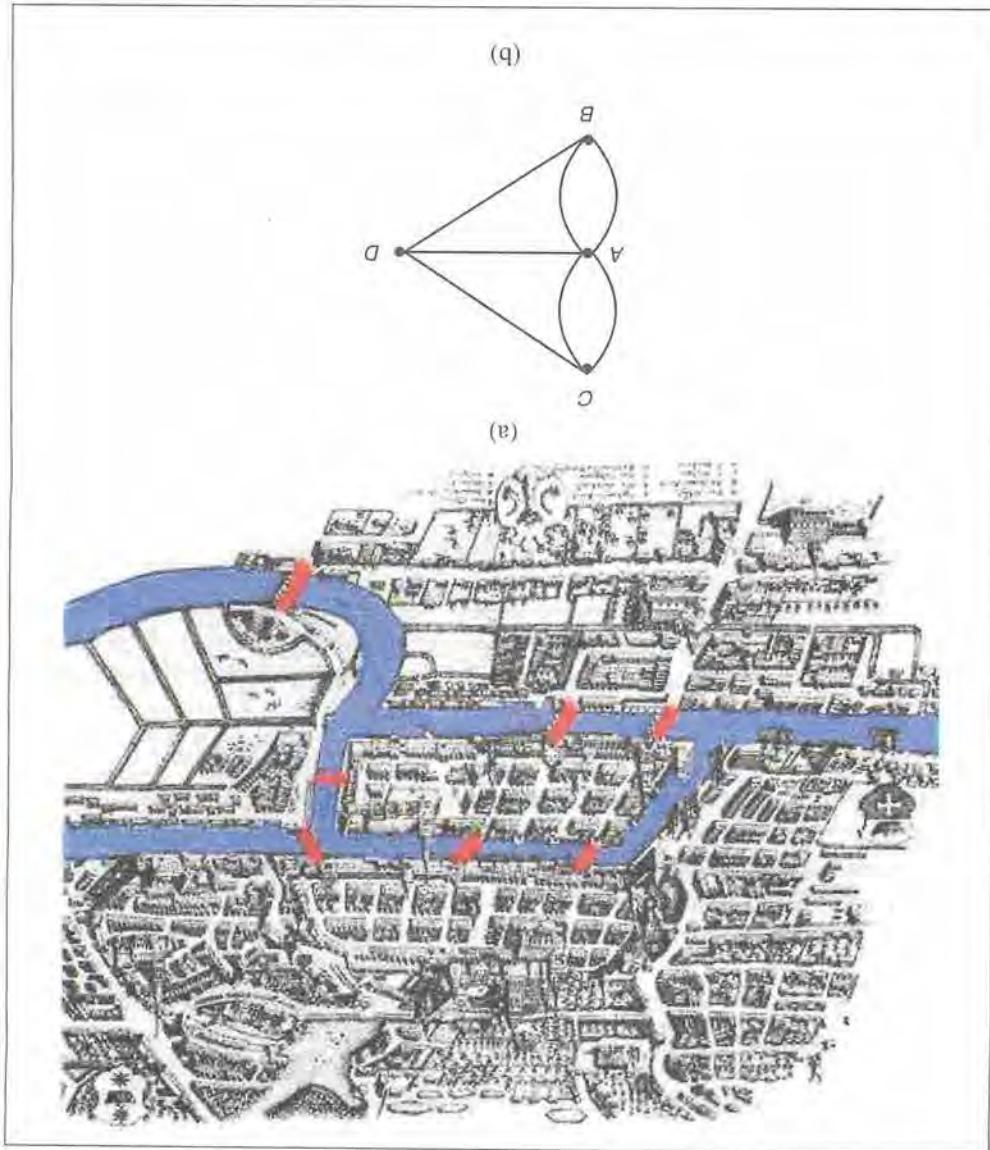
8.1 Sejarah Graf

Menurut catatan sejarah, masalah jembatan Königsberg adalah masalah yang pertama kali menggunakan graf (tahun 1736). Di kota Königsberg (sebelah timur negara bagian Prussia, Jerman), sekarang bernama kota Kaliningrad, terdapat sungai Pregal yang mengalir mengitari pulau Kneiphof lalu bercabang menjadi dua buah anak sungai (Gambar 8.1(a)).

Ada tujuh buah jembatan yang menghubungkan daratan yang dibelah oleh sungai tersebut. Masalah jembatan Königsberg adalah: apakah mungkin melalui ketujuh buah jembatan itu masing-masing tepat satu kali, dan kembali lagi ke tempat semula? Sebagian penduduk kota sepakat bahwa memang tidak mungkin melalui setiap jembatan itu hanya sekali dan kembali lagi ke tempat asal mula keberangkatan, tetapi mereka tidak dapat menjelaskan mengapa demikian jawabannya, kecuali dengan cara coba-coba. Tahun 1736, seorang matematikawan Swiss, L.Euler, adalah orang pertama yang berhasil menemukan jawaban masalah itu dengan pembuktian yang sederhana. Ia memodelkan masalah ini ke dalam graf. Daratan (titik-titik yang dihubungkan oleh jembatan) dinyatakannya sebagai titik (nokta) –yang disebut simpul (*vertex*)– dan jembatan dinyatakan sebagai garis –yang disebut sisi (*edge*). Setiap titik diberi label huruf *A*, *B*, *C*, dan *D*. Graf yang dibuat oleh Euler diperlihatkan pada Gambar 8.2(b).

Jawaban yang dikemukakan oleh Euler adalah: orang tidak mungkin melalui ketujuh jembatan itu masing-masing satu kali dan kembali lagi ke tempat asal keberangkatan jika derajat setiap simpul tidak seluruhnya genap. Yang dimaksud

Gambar 8.2 (a) Petak kota Ko niggaberig kuno dan jembatan bersejarahnya
 (b) graf yang merepresentasikan jembatan Konigsberg



dengannya, simpul C memiliki derajat 3 karena ada tiga buah gerbang yang bersisiian contohnya, simpul B dan D juga berderajat dua, sedangkan simpul A berderajat 5. Karenanya tidak semua simpul berderajat genap, maka tidak mungkin dilakukan perjalananan berupa sikuit (yang dimakan dengan sikuit Euler) pada graf tersebut. Kelak kita akan membahas lebih mendalam mengenai derajat dan strukturnya pada upabab selanjutnya.

8.2 Definisi Graf

Secara matematis, graf didefinisikan sebagai berikut:

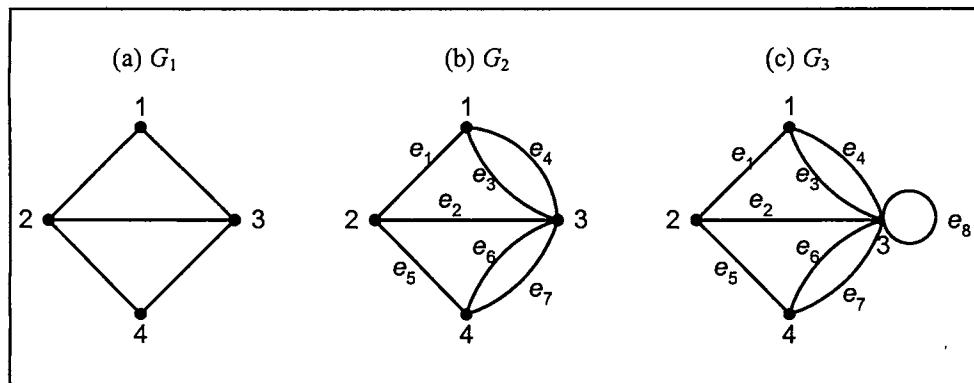
DEFINISI 8.1. Graf G didefinisikan sebagai pasangan himpunan (V, E) , ditulis dengan notasi $G = (V, E)$, yang dalam hal ini V adalah himpunan tidak-kosong dari simpul-simpul (*vertices* atau *node*) dan E adalah himpunan sisi (*edges* atau *arcs*) yang menghubungkan sepasang simpul.

Definisi 8.1 menyatakan bahwa V tidak boleh kosong, sedangkan E boleh kosong. Jadi, sebuah graf dimungkinkan tidak mempunyai sisi satu buah pun, tetapi simpulnya harus ada, minimal satu. Graf yang hanya mempunyai satu buah simpul tanpa sebuah sisi pun dinamakan **graf trivial**.

Simpul pada graf dapat dinomori dengan huruf, seperti $a, b, c, \dots, v, w, \dots$, dengan bilangan asli $1, 2, 3, \dots$, atau gabungan keduanya. Sedangkan sisi yang menghubungkan simpul u dengan simpul v dinyatakan dengan pasangan (u, v) atau dinyatakan dengan lambang e_1, e_2, \dots . Dengan kata lain, jika e adalah sisi yang menghubungkan simpul u dengan simpul v , maka e dapat ditulis sebagai

$$e = (\nu, \nu)$$

Secara geometri graf digambarkan sebagai sekumpulan noktah (simpul) di dalam bidang dwimatra yang dihubungkan dengan sekumpulan garis (sisi).



Gambar 8.3 Tiga buah graf (a) graf sederhana, (b) graf ganda, dan (c) graf semu

Contoh 8.1

Gambar 8.3 memperlihatkan tiga buah graf, G_1 , G_2 , dan G_3 . G_1 adalah graf dengan himpunan simpul V dan himpunan sisi E adalah

$$V = \{ 1, 2, 3, 4 \}$$

$$E = \{ (1, 2), (1, 3), (2, 3), (2, 4), (3, 4) \}$$

G_2 adalah graf dengan himpunan simpul V dan himpunan sisi E adalah:

$$\begin{aligned}V &= \{1, 2, 3, 4\} \\E &= \{(1, 2), (2, 3), (1, 3), (1, 3), (2, 4), (3, 4), (3, 4)\} \Rightarrow \text{himpunan ganda} \\&= \{e_1, e_2, e_3, e_4, e_5, e_6, e_7\}\end{aligned}$$

G_3 adalah graf dengan himpunan simpul V dan himpunan sisi E adalah:

$$\begin{aligned}V &= \{1, 2, 3, 4\} \\E &= \{(1, 2), (2, 3), (1, 3), (1, 3), (2, 4), (3, 4), (3, 4), (3, 3)\} \Rightarrow \text{himpunan ganda} \\&= \{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8\}\end{aligned}$$

Pada G_2 , sisi $e_3 = (1, 3)$ dan sisi $e_4 = (1, 3)$ dinamakan **sisi-ganda** (*multiple edges* atau *parallel edges*) karena kedua sisi ini menghubungi dua buah simpul yang sama, yaitu simpul 1 dan simpul 3. Pada G_3 , sisi $e_8 = (3, 3)$ dinamakan **gelang** atau **kalang** (*loop*) karena ia berawal dan berakhir pada simpul yang sama. ■

8.3 Jenis-jenis Graf

Graf dapat dikelompokkan menjadi beberapa kategori (jenis) bergantung pada sudut pandang pengelompokannya. Pengelompokan graf dapat dipandang berdasarkan ada tidaknya sisi ganda atau sisi kalang, berdasarkan jumlah simpul, atau berdasarkan orientasi arah pada sisi.

Berdasarkan ada tidaknya gelang atau sisi ganda pada suatu graf, maka secara umum graf dapat digolongkan menjadi dua jenis:

1. Graf sederhana (*simple graph*).

Graf yang tidak mengandung gelang maupun sisi-ganda dinamakan graf sederhana. G_1 pada Gambar 8.3(a) adalah contoh graf sederhana yang merepresentasikan jaringan komputer. Simpul menyatakan komputer, sedangkan sisi menyatakan saluran telepon untuk berkomunikasi. Saluran telepon dapat beroperasi pada dua arah.

Pada graf sederhana, sisi adalah pasangan tak-terurut (*unordered pairs*). Jadi, menuliskan sisi (u, v) sama saja dengan (v, u) . Kita dapat juga mendefinisikan graf sederhana $G = (V, E)$ terdiri dari himpunan tidak kosong simpul-simpul dan E adalah himpunan pasangan tak-terurut yang berbeda yang disebut sisi.

2. Graf tak-sederhana (*unsimple-graph*).

Graf yang mengandung sisi ganda atau gelang dinamakan graf tak-sederhana (*unsimple graph*). Ada dua macam graf tak-sederhana, yaitu **graf ganda** (*multigraph*) dan **graf semu** (*pseudograph*). Graf ganda adalah graf yang mengandung sisi ganda. Sisi ganda yang menghubungkan sepasang simpul bisa lebih dari dua buah. G_2 pada Gambar 8.3(b) adalah graf-ganda. Sisi ganda dapat diasosiasiaskan sebagai pasangan tak-terurut yang sama. Kita dapat juga mendefinisikan graf

ganda $G = (V, E)$ terdiri dari himpunan tidak kosong simpul-simpul dan E adalah himpunan-ganda (*multiset*) yang mengandung sisi ganda. Pada jaringan telekomunikasi, sisi ganda pada G_2 dapat diandaikan sebagai saluran telepon tambahan apabila beban komunikasi data antar komputer sangat padat. Perhatikanlah bahwa setiap graf sederhana juga adalah graf ganda, tetapi tidak setiap graf ganda merupakan graf sederhana.

Graf semu adalah graf yang mengandung gelang (*loop*). G_3 adalah graf semu (termasuk bila memiliki sisi ganda sekalipun). Sisi gelang pada G_3 dapat dianggap sebagai saluran telepon tambahan yang menghubungkan komputer dengan dirinya sendiri (mungkin untuk tujuan diagnostik). Graf semu lebih umum daripada graf ganda, karena sisi pada graf semu dapat terhubung ke dirinya sendiri.

Jumlah simpul pada graf kita sebut sebagai kardinalitas graf, dan dinyatakan dengan $n = |V|$, dan jumlah sisi kita nyatakan dengan $m = |E|$. Pada contoh di atas, G_1 mempunyai $n = 4$, dan $m = 4$, sedangkan G_2 mempunyai $n = 3$ dan $m = 4$.

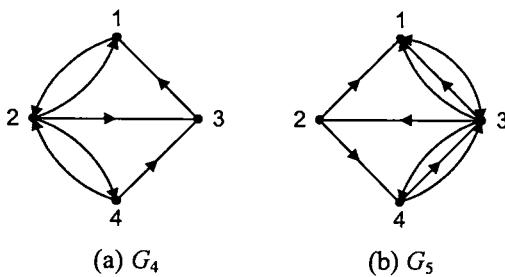
Sisi pada graf dapat mempunyai orientasi arah. Berdasarkan orientasi arah pada sisi, maka secara umum graf dibedakan atas 2 jenis:

1. Graf tak-berarah (*undirected graph*)

Graf yang sisinya tidak mempunyai orientasi arah disebut graf tak-berarah. Pada graf tak-berarah, urutan pasangan simpul yang dihubungkan oleh sisi tidak diperhatikan. Jadi, $(u, v) = (v, u)$ adalah sisi yang sama. Tiga buah graf pada Gambar 8.3 adalah graf tak-berarah. Pada jaringan telepon, sisi pada graf berarah menyatakan bahwa saluran teelpon dapat beroperasi pada dua arah.

2. Graf berarah (*directed graph* atau *digraph*)

Graf yang setiap sisinya diberikan orientasi arah disebut sebagai graf berarah. Kita lebih suka menyebut sisi berarah dengan sebutan **busur** (*arc*). Pada graf berarah, (u, v) dan (v, u) menyatakan dua buah busur yang berbeda, dengan kata lain $(u, v) \neq (v, u)$. Untuk busur (u, v) , simpul u dinamakan **simpul asal** (*initial vertex*) dan simpul v dinamakan **simpul terminal** (*terminal vertex*). G_4 pada Gambar 8.4(a) adalah contoh graf berarah. Pada G_4 dapat dibayangkan sebagai saluran telepon tidak dapat beroperasi pada dua arah. Saluran hanya beroperasi pada arah yang ditunjukkan oleh anak panah. Jadi, sebagai contoh, saluran telepon $(1, 2)$ tidak sama dengan saluran telepon $(2, 1)$. Graf berarah sering dipakai untuk menggambarkan aliran proses, peta lalu lintas suatu kota (jalan searah atau dua arah), dan sebagainya. Pada graf berarah, gelang diperbolehkan, tetapi sisi ganda tidak.



Gambar 8.4 (a) graf berarah, (b) graf-ganda berarah

Definisi graf dapat diperluas sehingga mencakup **graf-ganda berarah** (*directed multigraph*). Pada graf-ganda berarah, gelang dan sisi ganda diperbolehkan ada. G_5 pada Gambar 8.4(b) adalah contoh graf-ganda berarah. Tabel 8.1 meringkas perluasan definisi graf.

Tabel 8.1 Jenis-jenis graf [ROS99]

Jenis	Sisi	Sisi ganda dibolehkan?	Sisi gelang dibolehkan?
Graf sederhana	Tak-berarah	Tidak	Tidak
Graf ganda	Tak-berarah	Ya	Tidak
Graf semu	Tak-berarah	Ya	Ya
Graf berarah	Bearah	Tidak	Ya
Graf-ganda berarah	Bearah	Ya	Ya

Di dalam buku ini, kita menyebut **graf** untuk jenis graf apa saja, baik sisinya tak-berarah maupun berarah, baik mengandung gelang maupun sisi ganda, baik graf sederhana maupun graf tak-sederhana.

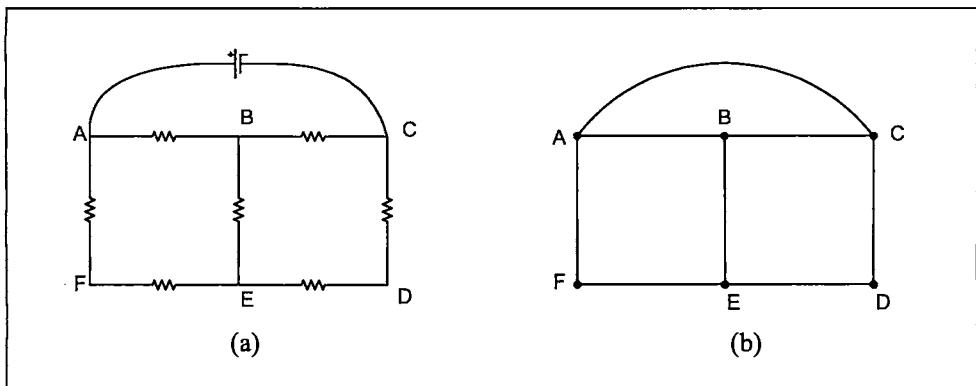
8.4 Contoh Terapan Graf

Seperti yang sudah disebutkan di atas, aplikasi graf sangat luas. Graf dipakai di berbagai disiplin ilmu maupun dalam kehidupan sehari-hari. Penggunaan graf di berbagai bidang tersebut adalah untuk memodelkan persoalan. Di bawah ini dikemukakan terapan graf dalam bidang kelistrikan, kimia, ilmu komputer, dan pertandingan olahraga. Terapan graf lainnya akan dibicarakan pada waktu membahas teori-teori di dalam graf.

1. Rangkaian listrik.

Kirchoff (1847) menggunakan graf untuk memodelkan rangkaian listrik. Berdasarkan graf tersebut Kirchoff menurunkan persamaan arus yang masuk dan keluar pada

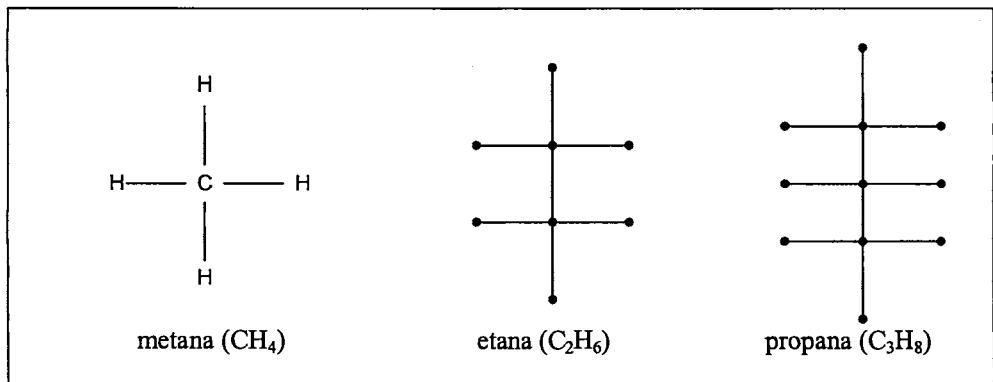
tiap simpul. Dari sistem persamaan lanjar (*linier*) simultan yang diperoleh dapat dapat dihitung arus listrik yang mengalir pada setiap komponen (Gambar 8.5).



Gambar 8.5 (a) Rangkaian listrik, (b) graf yang menyatakan rangkaian listrik

2. Isomer senyawa kimia karbon

Arthur Cayley (1857) menggunakan graf dalam memodelkan molekul senyawa alkana C_nH_{2n+2} untuk menghitung jumlah isomernya. Atom karbon (C) dan atom hidrogen (H) dinyatakan sebagai simpul, sedangkan ikatan antara atom C dan H dinyatakan sebagai sisi (Gambar 8.6). Isomer adalah senyawa kimia yang mempunyai rumus molekul sama tetapi rumus bangun (bentuk graf) berbeda.



Gambar 8.6 Graf senyawa alkana, masing-masing metana, etana, dan propana

3. Transaksi konkuren pada basis data terpusat

Ini adalah terapan graf dalam bidang komputer. Basis data (*database*) terpusat melayani beberapa transaksi (T) yang dilakukan secara konkuren (bersamaan). Transaksi terhadap basis data dapat berupa operasi pembacaan dan operasi

penulisan terhadap data yang sama. Persoalan kritis pada proses konkuren adalah *deadlock*, yaitu keadaan yang timbul karena beberapa transaksi saling menunggu transaksi lainnya sehingga sistem menjadi *hang*. Misalnya, transaksi T_1 akan membaca data B yang sedang ditulis oleh transaksi T_2 , sedangkan T_2 akan membaca data A yang sedang ditulis T_1 . Kedua transaksi saling menunggu data yang sedang dikuncinya (*circular wait*). Bila terdapat lebih dari dua transaksi yang saling menunggu sehingga membentuk siklus, maka timbul *deadlock*. Cara yang digunakan sistem untuk mendeteksi *deadlock* adalah dengan membangun graf transaksi secara periodik dan memeriksa apakah terdapat siklus pada grafnya. Jika ada siklus, maka kondisi *deadlock* terjadi .

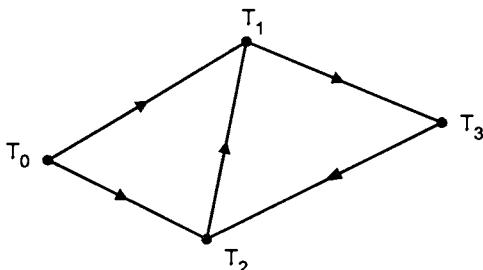
Misalkan:

- transaksi T_0 menunggu transaksi T_1 dan T_2 ;
- transaksi T_2 menunggu transaksi T_1 ;
- transaksi T_1 menunggu transaksi T_3 ;
- transaksi T_3 menunggu transaksi T_2 ;

Graf berarah yang menyatakan transaksi menunggu transaksi lainnya ditunjukkan pada Gambar 8.7. Simpul menyatakan transaksi, sedangkan busur (T_i, T_j) menyatakan transaksi T_i menunggu transaksi T_j . Graf ini mengandung siklus, yaitu

$$T_1 - T_3 - T_2 - T_1$$

Untuk mengatasi *deadlock*, sistem harus memutuskan siklus dengan cara membatalkan satu atau lebih transaksi di dalam siklus. Metode penanganan *deadlock* tidak dibahas di dalam buku ini, karena merupakan bagian dari kuliah Sistem Operasi dan Sistem Basis Data.



Gambar 8.7 Graf transaksi yang menunjukkan keadaan deadlock

4. Pengujian program

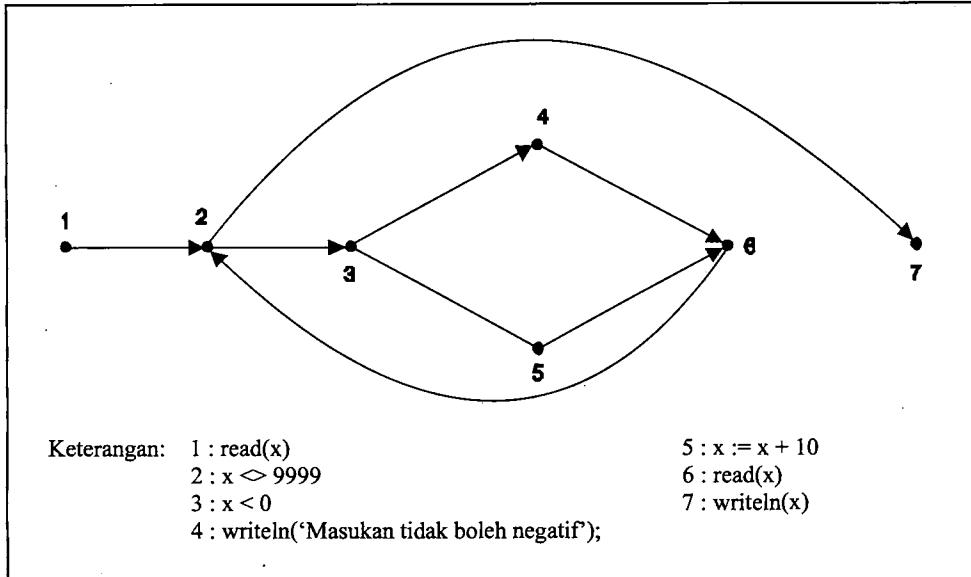
Dalam bidang rekayasa perangkat lunak, sebuah program harus mengalami tahap pengujian untuk menemukan kesalahan (*bug*). Salah satu pengujian program adalah

pengujian eksekusi. Aliran kendali program harus diperiksa untuk memastikan apakah aliran tersebut sudah benar untuk berbagai kasus data uji. Aliran kendali program dimodelkan dengan graf berarah yang dinamakan **graf alir** (*flow graph*). Pada graf berarah tersebut, simpul menyatakan pernyataan atau kondisi yang dievaluasi, sedangkan busur menyatakan aliran kendali program ke pernyataan atau kondisi berikutnya.

Sebagai contoh, misalkan terdapat sebagian teks program Pascal di bawah ini:

```
read(x);
while x <> 9999 do
begin
    if x < 0 then
        writeln('Masukan tidak boleh negatif')
    else
        x := x + 10;
    read(x);
end;
writeln(x);
```

Graf alir yang menggambarkan aliran kendali program ditunjukkan pada Gambar 8.8.

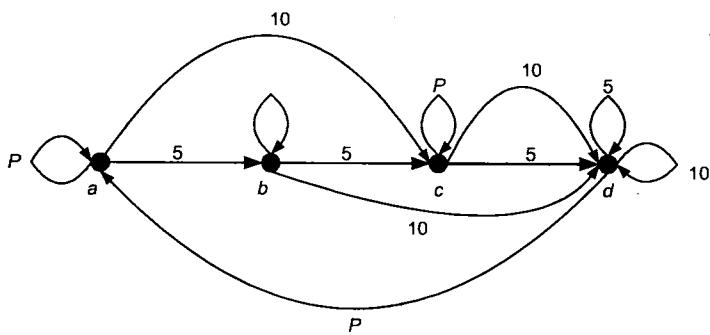


Gambar 8.8 Graf alir dari sebagian teks program

Data pengujian harus dirancang sedemikian sehingga semua lintasan di dalam graf alir pernah dilalui minimum satu kali. Tujuannya agar kesalahan pada setiap lintasan eksekusi dapat ditemukan dan perbaikan program dilakukan.

5. Terapan graf di dalam teori otomata.

Marilah kita simak masalah pemodelan perilaku sebuah mesin jaja (*vending machine*) yang menjual coklat seharga 15 sen [LIU85]. Untuk memudahkan, kita akan memisalkan bahwa mesin tersebut hanya menerima uang logam 5 sen dan 10 sen, dan mesin tidak akan memberi kembalian bila yang dimasukkan lebih dari 15 sen. Graf berbobot (setiap sisi diberi sebuah harga, akan dijelaskan kemudian) pada Gambar 8.10 menggambarkan perilaku mesin ini, dengan simpul menyatakan banyaknya uang logam yang dimasukkan, yaitu 0, 5, 10, dan 15 sen atau lebih. Setiap saat seorang pembeli dapat melakukan salah satu dari tiga hal berikut: memasukkan sebuah uang logam 5 sen, memasukkan sebuah uang logam 10 sen, dan menekan tombol coklat (P) pilihannya. Dengan demikian, di dalam graf pada Gambar 8.9 ada tiga buah sisi dari setiap simpul yang berbobot 5, 10, dan P . Sisi dengan bobot 5 menghitung kembali jumlah uang yang ada di dalam mesin ketika pembeli memasukkan sebuah uang logam 5 sen, dan sisi dengan bobot 10 menghitung kembali jumlah uang yang ada di dalam mesin ketika seorang pembeli memasukkan uang logam 10 sen. Kiranya jelas, ketika kita ada di simpul a , b , dan c , tidak akan terjadi apa-apa meskipun tombol kita tekan; mesin akan mengeluarkan sepotong coklat hanya bila kita sampai pada simpul d .



Keterangan:

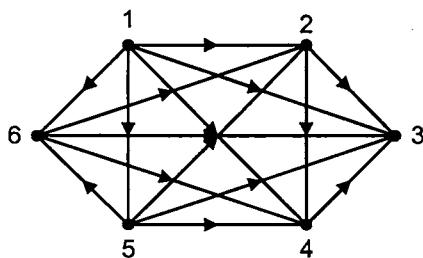
- a : 0 sen dimasukkan
- b : 5 sen dimasukkan
- c : 10 sen dimasukkan
- d : 15 sen atau lebih dimasukkan

Gambar 8.9 Graf yang memodelkan perilaku mesin jaja

6. Turnamen Round-Robin

Turnamen yang setiap tim bertanding dengan tim lainnya hanya sekali disebut turnamen *round-robin*. Turnamen semacam itu dimodelkan dengan graf berarah,

yang dalam hal ini simpul menyatakan tiap tim yang bertanding, dan busur menyatakan pertandingan. Busur (a, b) berarti tim a berhasil memukul tim b . Gambar 8.10 memperlihatkan turnamen *round-robin* untuk 6 buah tim. Tim 1 tidak terkalahkan, sedangkan tim 3 tidak pernah menang.

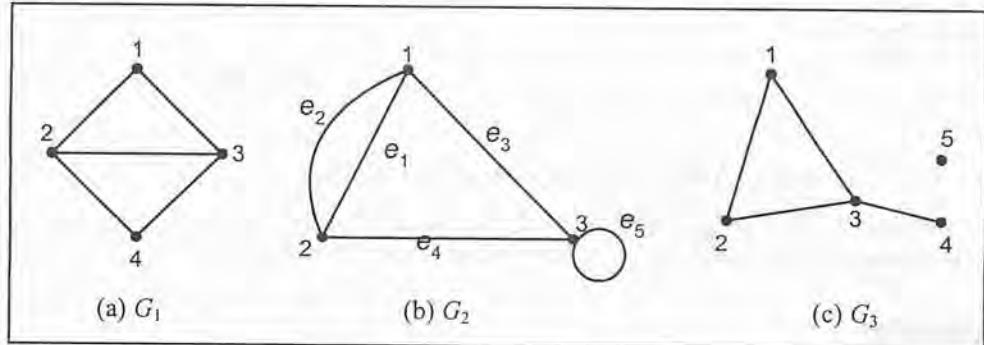


Gambar 8.10 Turnamen *round-robin*

Contoh terapan graf yang lain adalah menyatakan aliran informasi dalam pengolahan sinyal dan aliran massa dalam industri kimia. Graf juga berguna memodelkan sesuatu yang abstrak, seperti struktur perusahaan, tingkatan sosial, pohon keluarga, aliran kerja dalam proyek, perencanaan dan manajemen proyek, perpindahan dalam permainan (*game*), dan langkah-langkah pemecahan masalah. Terapan yang terakhir ini merupakan kemampuan dasar yang harus dikuasai dalam bidang kecerdasan buatan (*artificial intelligence*).

8.5 Terminologi Dasar

Kita akan sering menggunakan terminologi (istilah) yang berkaitan dengan graf. Di bawah ini didefinisikan beberapa terminologi yang sering dipakai. Contoh graf pada Gambar 8.11 akan digunakan untuk memperjelas terminologi yang kita definisikan. Graf yang pertama, G_1 , adalah graf sederhana, G_2 adalah graf semu yang mengandung sisi ganda maupun gelang, sedangkan G_3 adalah graf dengan sebuah simpul yang terpisah dari simpul lainnya. Ketiga buah graf ini adalah graf tidak-berarah. Untuk terminologi yang menyangkut graf berarah, contoh grafnya akan digambarkan pada waktu pembahasan.



Gambar 8.11 Tiga buah graf, G_1 , G_2 , dan G_3

1. Bertetangga (*Adjacent*)

DEFINISI 8.2. Dua buah simpul pada graf tak-berarah G dikatakan **bertetangga** bila keduanya terhubung langsung dengan sebuah sisi. Dengan kata lain, u bertetangga dengan v jika (u, v) adalah sebuah sisi pada graf G .

Contoh 8.2

Pada Gambar 8.11(a), simpul 1 bertetangga dengan simpul 2 dan 3, tetapi simpul 1 tidak bertetangga dengan simpul 4. ■

Pada graf berarah, sisi kita sebut busur. Jika (u, v) adalah busur maka u dikatakan bertetangga dengan v dan v dikatakan tetangga dari u . Pada Gambar 8.4(a), simpul 1 bertetangga dengan simpul 2, dan simpul 2 dikatakan tetangga dari simpul 1.

2. Bersisian (*Incident*)

DEFINISI 8.3. Untuk sembarang sisi $e = (u, v)$, sisi e dikatakan **bersisian** dengan simpul u dan simpul v .

Contoh 8.3

Pada Gambar 8.11(a), sisi $(2, 3)$ bersisian dengan simpul 2 dan simpul 3, sisi $(2, 4)$ bersisian dengan simpul 2 dan simpul 4, tetapi sisi $(1, 2)$ tidak bersisian dengan simpul 4. ■

3. Simpul Terpencil (*Isolated Vertex*)

DEFINISI 8.4. Simpul terpencil ialah simpul yang tidak mempunyai sisi yang bersisian dengannya. Atau, dapat juga dinyatakan bahwa simpul terpencil adalah simpul yang tidak satupun bertetangga dengan simpul-simpul lainnya.

Contoh 8.4

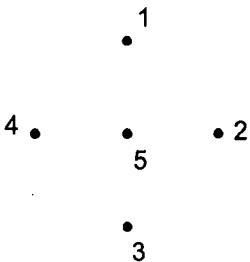
Pada Gambar 8.11(c), simpul 5 adalah simpul terpencil.

4. Graf Kosong (*Null Graph* atau *Empty Graph*)

DEFINISI 8.5. Graf yang himpunan sisinya merupakan himpunan kosong disebut sebagai graf kosong dan ditulis sebagai N_0 , yang dalam hal ini N_0 adalah jumlah simpul.

Contoh 8.5

Graf pada Gambar 8.12 adalah graf N_5 . ■



Gambar 8.12 Graf kosong N_5

5. Derajat (*Degree*)

DEFINISI 8.6. Derajat suatu simpul pada graf tak-berarah adalah jumlah sisi yang bersifian dengan simpul tersebut.

Notasi: $d(v)$ menyatakan derajat simpul v .

Contoh 8.6

Pada Gambar 8.11(a),

$$\begin{aligned}d(1) &= d(4) = 2 \\d(2) &= d(3) = 3\end{aligned}$$

Simpul terpencil adalah simpul dengan $d(v) = 0$, karena tidak ada satupun sisi yang bersifian dengan simpul tersebut. Pada Gambar 8.11(c), $d(5) = 0$.

Sisi gelang (*loop*) dihitung berderajat dua. Jadi, untuk graf pada Gambar 8.11(b), $d(2) = 4$. Secara umum, jika terdapat g buah gelang dan e buah sisi bukan-gelang yang bersifian dengan simpul v , maka derajat simpul v adalah

$$d(v) = 2g + e \quad (8.1)$$

Alasan mengapa gelang mengkontribusikan dua untuk derajat simpulnya adalah karena gelang direpresentasikan sebagai (v, v) , dan simpul v bersisian dua kali pada sisi (v, v) .

Simpul yang berderajat satu disebut **anting-anting** (*pendant vertex*). Dengan kata lain, anting-anting hanya bertetangga dengan sebuah simpul. Pada Gambar 8.12(c), $d(4) = 1$, karena itu simpul 4 adalah anting-anting.

Pada graf berarah, derajat suatu simpul dibedakan menjadi dua macam untuk mencerminkan jumlah busur dengan simpul tersebut sebagai simpul asal dan jumlah busur dengan simpul tersebut sebagai simpul terminal.

DEFINISI 8.7. Pada graf berarah, derajat simpul v dinyatakan dengan $d_{in}(v)$ dan $d_{out}(v)$, yang dalam hal ini

$$\begin{aligned}d_{in}(v) &= \text{derajat-masuk (in-degree)} = \text{jumlah busur yang masuk ke simpul } v \\d_{out}(v) &= \text{derajat-keluar (out-degree)} = \text{jumlah busur yang keluar dari simpul } v\end{aligned}$$

dan

$$d(v) = d_{in}(v) + d_{out}(v) \quad (8.2)$$

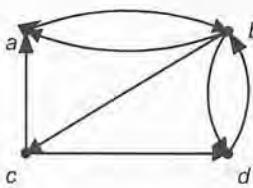
Catatlah bahwa sisi gelang pada graf berarah menyumbangkan 1 untuk derajat-masuk dan 1 untuk derajat-keluar.

Contoh 8.7

Tinjau graf berarah berikut ini:

Derajat setiap simpul adalah

$$\begin{aligned}d_{in}(a) &= 3; d_{out}(a) = 2 \\d_{in}(b) &= 2; d_{out}(b) = 3 \\d_{in}(c) &= 1; d_{out}(c) = 2 \\d_{in}(d) &= 2; d_{out}(d) = 1\end{aligned}$$



Pada graf berarah $G = (V, E)$ selalu berlaku hubungan

$$\sum_{v \in V} d_{in}(v) = \sum_{v \in V} d_{out}(v) = |E|$$

Misalnya pada Contoh 8.7 di atas,

$$\sum_{v \in V} d_{in}(v) = 3 + 2 + 1 + 2 = 8 = \sum_{v \in V} d_{out}(v) = 2 + 3 + 2 + 1 = 8 = |E|$$

Lemma Jabat Tangan. Jumlah derajat semua simpul pada suatu graf adalah genap, yaitu dua kali jumlah sisi pada graf tersebut. Dengan kata lain, jika $G = (V, E)$, maka

$$\sum_{v \in V} d(v) = 2|E| \quad (8.3)$$

(catatan: ingatlah $2|E|$ selalu bernilai genap)

Lemma ini dikenal dengan lemma jabat tangan (*handshaking lemma*). Hal ini disebabkan oleh setiap sisi dihitung dua kali, yaitu pada ujung kiri sebagai bagian dari simpul kiri dan pada ujung kanan dihitung sebagai bagian dari simpul kanan. Layaknya orang berjabat tangan, maka jumlah tangan yang berjabatan adalah genap dan jumlah tangan yang berjabatan adalah dua kali jumlah jabatan tangan yang terjadi [DUL94]. Cataolah bahwa Lemma Jabat Tangan juga benar untuk graf berarah, yang dalam hal ini $d(v) = d_{\text{in}}(v) + d_{\text{out}}(v)$.

Contoh 8.8

Jumlah derajat seluruh simpul pada graf Gambar 8.11(a) adalah:

$$d(1) + d(2) + d(3) + d(4) = 2 + 3 + 3 + 2 = 10 = 2 \times \text{jumlah sisi} = 2 \times 5$$

Jumlah derajat seluruh simpul pada graf Gambar 8.11(b) adalah:

$$d(1) + d(2) + d(3) = 3 + 3 + 4 = 10 = 2 \times \text{jumlah sisi} = 2 \times 5$$

Jumlah derajat seluruh simpul pada graf Gambar 8.11(c) adalah

$$d(1) + d(2) + d(3) + d(4) + d(5) = 2 + 2 + 3 + 1 + 0 = 8 = 2 \times \text{jumlah sisi} = 2 \times 4$$

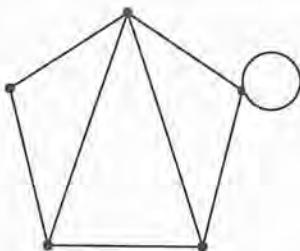
Contoh 8.9

Diketahui graf dengan lima buah simpul. Dapatkah kita menggambar graf tersebut jika derajat masing-masing simpul adalah:

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

Penyelesaian:

- (a) tidak dapat, karena jumlah derajat semua simpulnya ganjil ($2 + 3 + 1 + 1 + 2 = 9$).
- (b) dapat, karena jumlah derajat semua simpulnya genap ($2 + 3 + 3 + 4 + 4 = 16$). Salah satu kemungkinan graf yang dapat digambar ditunjukkan pada Gambar 8.13. Ternyata grafnya bukan graf sederhana. ■



Gambar 8.13. Graf dengan derajat setiap simpul masing-masing 2, 3, 3, 4, 4

Akibat dari Lemma Jabat Tangan di atas kita menurunkan teorema berikut:

Teorema 8.1. Untuk sembarang graf G , banyaknya simpul yang berderajat ganjil selalu genap.

Bukti:

Misalkan V_1 dan V_2 masing-masing adalah himpunan simpul yang berderajat genap dan berderajat ganjil pada graf $G = (V, E)$. Persamaan (8.3) dapat ditulis sebagai

$$\sum_{v \in V_1} d(v) + \sum_{v \in V_2} d(v) = 2|E| \quad (8.4)$$

Karena $d(v)$ genap untuk $v \in V_1$, maka suku pertama dari ruas kiri persamaan selalu bernilai genap. Ruas kanan persamaan (8.4) juga bernilai genap. Nilai genap pada ruas kanan hanya benar bila suku kedua dari ruas kiri juga harus genap agar
genap + genap = genap

Karena $d(v)$ ganjil untuk $v \in V_2$, maka banyaknya simpul v di dalam V_2 harus genap agar jumlah seluruh derajatnya bernilai genap. Jadi, banyaknya simpul yang berderajat ganjil selalu genap. ■

Perhatikan graf pada Gambar 8.11(c), di sini banyaknya simpul yang berderajat ganjil ada dua buah, yaitu simpul 3 dan simpul 4.

6. Lintasan (*Path*)

DEFINISI 8.7. Lintasan yang panjangnya n dari simpul awal v_0 ke simpul tujuan v_n di dalam graf G ialah barisan berselang-seling simpul-simpul dan sisi-sisi yang membentuk $v_0, e_1, v_1, e_2, v_2, \dots, v_{n-1}, e_n, v_n$ sedemikian sehingga $e_1 = (v_0, v_1)$, $e_2 = (v_1, v_2), \dots, e_n = (v_{n-1}, v_n)$ adalah sisi-sisi dari graf G .

Jika graf yang ditinjau adalah graf sederhana, maka kita cukup menuliskan lintasan sebagai barisan simpul-simpul saja: $v_0, v_1, v_2, \dots, v_{n-1}, v_n$, karena antara dua buah simpul berturutan di dalam lintasan tersebut hanya ada satu sisi. Sebagai contoh, pada Gambar 8.11(a), lintasan 1, 2, 4, 3 adalah lintasan dengan barisan sisi (1,2), (2,4), (4,3). Istilah lain untuk lintasan adalah jalur.

Pada graf yang mengandung sisi ganda, kita harus menulis lintasan sebagai barisan berselang-seling antara simpul dan sisi menghindari kerancuan sisi mana dari sisi-sisi ganda yang dilalui. Misalnya pada Gambar 8.11(b),

1, e_1 , 2, e_4 , 3, e_5 , 3

adalah lintasan dari simpul 1 ke simpul 3 yang melalui sisi e_1 , e_4 , dan e_5 .

Catatlah bahwa simpul dan sisi yang dilalui di dalam lintasan boleh berulang. Sebuah lintasan dikatakan **lintasan sederhana** (*simple path*) jika semua simpulnya berbeda (setiap sisi yang dilalui hanya satu kali).

Lintasan yang berawal dan berakhir pada simpul yang sama disebut **lintasan tertutup** (*closed path*), sedangkan lintasan yang tidak berawal dan berakhir pada simpul yang sama disebut **lintasan terbuka** (*open path*).

Contoh 8.10

Pada Gambar 8.11 (a),

lintasan 1, 2, 4, 3 adalah lintasan sederhana, juga lintasan terbuka.

lintasan 1, 2, 4, 3, 1 adalah juga lintasan sederhana, juga lintasan tertutup.

lintasan 1, 2, 4, 3, 2 bukan lintasan sederhana, tetapi lintasan terbuka. ■

Panjang lintasan adalah jumlah sisi dalam lintasan tersebut. Lintasan 1, 2, 4, 3 pada Gambar 8.11(a) memiliki panjang 3.

7. Siklus (*Cycle*) atau Sirkuit (*Circuit*)

DEFINISI 8.8. Lintasan yang berawal dan berakhir pada simpul yang sama disebut **sirkuit** atau **siklus**.

Pada Gambar 8.11(a), 1, 2, 3, 1 adalah sebuah sirkuit. Panjang sirkuit adalah jumlah sisi di dalam sirkuit tersebut. Sirkuit 1, 2, 3, 1 pada Gambar 8.11(a)

memiliki panjang 3. Sebuah sirkuit dikatakan **sirkuit sederhana** (*simple circuit*) jika setiap sisi yang dilalui berbeda.

Contoh 8.11

Pada Gambar 8.11(a), 1, 2, 3, 1 adalah sirkuit sederhana, sedangkan 1, 2, 4, 3, 2, 1 bukan sirkuit sederhana, karena sisi (1, 2) dilalui dua kali. ■

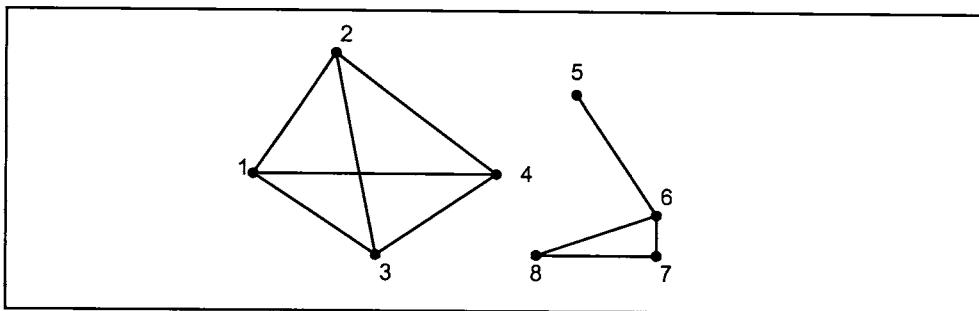
8. Terhubung (*Connected*)

Keterhubungan dua buah simpul adalah penting di dalam graf. Dua buah simpul u dan simpul v dikatakan **terhubung** jika terdapat lintasan dari u ke v . Jika dua buah simpul terhubung maka pasti simpul yang pertama dapat dicapai dari simpul yang kedua. Dua simpul terminal pada jaringan komputer hanya dapat berkomunikasi bila keduanya terhubung.

Jika setiap pasang simpul di dalam graf terhubung, maka graf tersebut kita katakan graf terhubung. Secara formal, definisi graf terhubung adalah sebagai berikut:

DEFINISI 8.9. Graf tak-berarah G disebut **graf terhubung** (*connected graph*) jika untuk setiap pasang simpul u dan v di dalam himpunan V terdapat lintasan dari u ke v (yang juga harus berarti ada lintasan dari v ke u). Jika tidak, maka G disebut **graf tak-terhubung** (*disconnected graph*).

G_1 dan G_2 pada Gambar 8.11 adalah graf terhubung, sedangkan G_3 tidak. Graf pada Gambar 8.14 di bawah ini juga adalah contoh graf yang tak-terhubung.



Gambar 8.14 Graf tak-berarah tidak terhubung

Sebagai catatan, graf yang hanya terdiri atas satu simpul saja (tidak ada sisi) tetap kita katakan terhubung, karena simpul tunggalnya terhubung dengan diringan sendiri, juga dikatakan graf terhubung.

Pada graf berarah, definisi graf terhubung kita rumuskan sebagai berikut:

DEFINISI 8.11. Graf berarah G dikatakan terhubung jika graf tak-berarahnya terhubung (graf tak-berarah dari G diperoleh dengan menghilangkan arahnya).

Keterhubungan dua buah simpul pada graf berarah dibedakan menjadi terhubung kuat dan terhubung lemah.

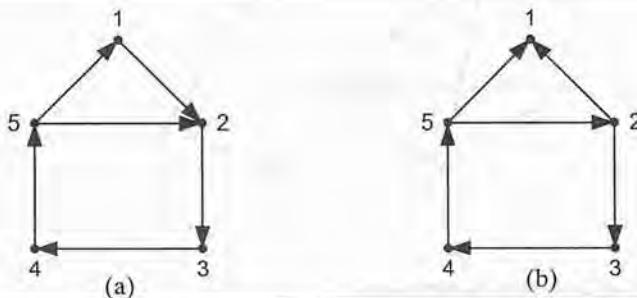
Dua simpul, u dan v pada graf berarah G disebut **terhubung kuat** (*strongly connected*) jika terdapat lintasan berarah dari u ke v , dan juga sebaliknya lintasan berarah dari v ke u . Pada Gambar 8.15(a), simpul 1 dan simpul 3 terhubung kuat karena terdapat lintasan dari 1 ke 3 (yaitu 1, 2, 3), begitu juga terdapat lintasan dari 3 ke 1 (yaitu 3, 4, 5, 1).

Jika u dan v tidak terhubung kuat tetapi tetap terhubung pada graf tak-berarahnya, maka u dan v dikatakan **terhubung lemah** (*weakly connected*). Pada Gambar 8.15(b), simpul 1 dan simpul 3 terhubung lemah karena hanya terdapat lintasan dari 3 ke 1 (yaitu 3, 5, 4, 1), tetapi tidak ada lintasan dari 1 ke 3.

Kedua hal di atas (terhubung kuat dan terhubung lemah) melahirkan definisi graf terhubung kuat:

DEFINISI 8.12. Graf berarah G disebut **graf terhubung kuat** (*strongly connected graph*) apabila untuk setiap pasang simpul sembarang v_i dan v_j di G terhubung kuat. Kalau tidak, G disebut **graf terhubung lemah**.

Graf pada Gambar 8.15(a) adalah graf terhubung kuat, karena untuk sembarang sepasang simpul di dalam graf terdapat lintasan, sedangkan graf pada Gambar 8.16(b) adalah graf terhubung lemah karena tidak semua pasangan simpul mempunyai lintasan dari dua arah.



Gambar 8.15 (a) graf berarah terhubung kuat, (b) graf berarah terhubung lemah

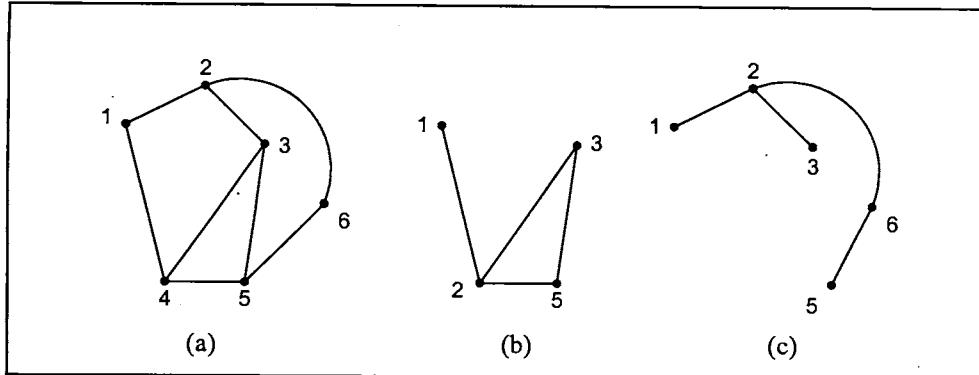
9. Upagraf (*Subgraph*) dan Komplemen Upagraf

DEFINISI 8.13. Misalkan $G = (V, E)$ adalah sebuah graf. $G_1 = (V_1, E_1)$ adalah **upagraf** (*subgraph*) dari G jika $V_1 \subseteq V$ dan $E_1 \subseteq E$

Gambar 8.17(b)) adalah upagraf dari graf pada Gambar 8.17(a).

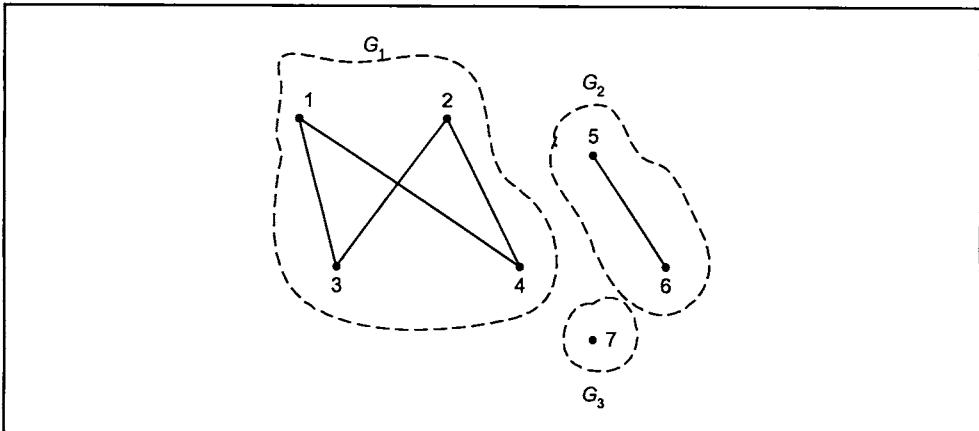
DEFINISI 8.14. Komplemen dari upagraph G_1 terhadap graf G adalah graf $G_2 = (V_2, E_2)$ sedemikian sehingga $E_2 = E - E_1$ dan V_2 adalah himpunan simpul yang anggota-anggota E_2 bersisian dengannya.

Gambar 8.16(c) adalah komplemen dari upagraf pada Gambar 8.16(b).



Gambar 8.16 (a) Graf G_1 , (b) sebuah upagraf dari G_1 , dan
 (c) komplemen dari upagraf yang bersesuaian.

Jika graf tidak terhubung, maka graf tersebut terdiri atas beberapa komponen terhubung (*connected component*). **Komponen terhubung** (atau disingkat “komponen” saja) adalah upagraf terhubung dari graf G yang tidak terdapat di dalam upagraf terhubung dari G yang lebih besar. Ini berarti setiap komponen terhubung di dalam graf G saling lepas (*disjoint*). Pada Gambar 8.17 di bawah ini, graf G mempunyai 3 buah komponen terhubung, yaitu G_1 , G_2 , dan G_3 . Catatlah bahwa graf terhubung hanya terdiri dari satu komponen, yaitu graf itu sendiri.



Gambar 8.17 Graf G yang mempunyai 4 buah komponen, yaitu G_1 , G_2 , dan G_3

Contoh 8.12

Tanpa menggambar grafnya, tentukan komponen terhubung dari $G = (V, E)$ yang dalam hal ini $V = \{a, b, c, d, e, f\}$ dan $E = \{(a, d), (c, d)\}$.

Penyelesaian:

Simpul a bertetangga dengan d , sedangkan simpul d bertetangga dengan c , ini berarti a juga terhubung dengan c . Simpul-simpul lainnya, b , e , dan f merupakan simpul terpencil. Dengan demikian, ada 4 buah komponen terhubung di dalam G , yaitu

$$G_1 = (V_1, E_1) \text{ dengan } V_1 = \{a, c, d\} \text{ dan } E_1 = \{(a, d), (c, d)\}$$

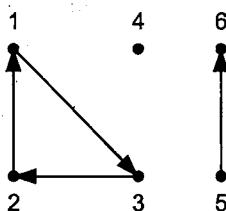
$$G_2 = (V_2, E_2) \text{ dengan } V_2 = \{b\} \text{ dan } E_2 = \{\}$$

$$G_3 = (V_3, E_3) \text{ dengan } V_3 = \{e\} \text{ dan } E_3 = \{\}$$

$$G_4 = (V_4, E_4) \text{ dengan } V_4 = \{f\} \text{ dan } E_4 = \{\}$$

dan $V_1 \cup V_2 \cup V_3 \cup V_4 = V$, $E_1 \cup E_2 \cup E_3 \cup E_4 = E$, $G_1 \cap V_2 \cap V_3 \cap V_4 = \emptyset$. ■

Pada graf berarah, **komponen terhubung kuat** (*strongly connected component*) adalah upagraf terhubung-kuat dari graf G yang tidak terdapat di dalam upagraf terhubung-kuat dari G yang lebih besar. Graf pada Gambar 8.18 di bawah ini mempunyai dua buah komponen terhubung kuat, yaitu upagraf dengan simpul 1, 2, 3 dan upagraf yang hanya mempunyai satu simpul, 4.



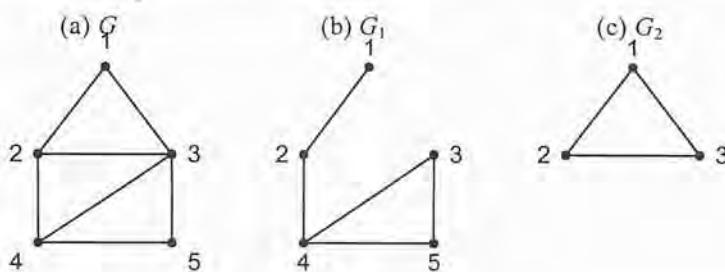
Gambar 8.18 Graf berarah G yang mempunyai 2 buah komponen terhubung kuat

10. Upagraf Merentang (*Spanning Subgraph*)

DEFINISI 8.14. Upagraf $G_1 = (V_1, E_1)$ dari $G = (V, E)$ dikatakan **upagraf merentang** jika $V_1 = V$ (yaitu G_1 mengandung semua simpul dari G).

Contoh 8.13

Pada Gambar 8.19, G_1 adalah upagraf merentang dari G , tetapi G_2 bukan upagraf merentang dari G karena G_2 tidak mengandung semua simpul G . ■



Gambar 8.19 (a) graf G ,
 (b) upagraf merentang dari G , dan
 (c) bukan upagraf merentang dari G

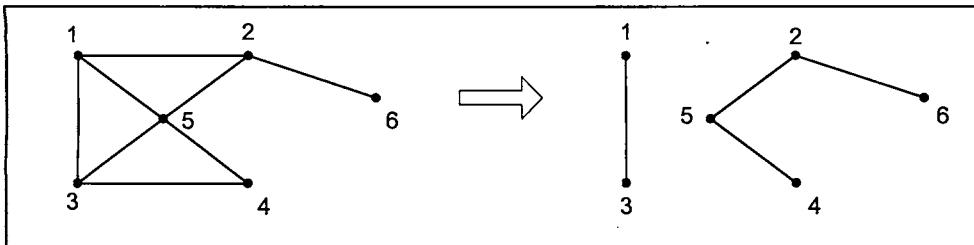
11. Cut-Set

DEFINISI 8.14. *Cut-set* dari graf terhubung G adalah himpunan sisi yang bila dibuang dari G menyebabkan G tidak terhubung. Jadi, *cut-set* selalu menghasilkan dua buah komponen terhubung.

Nama lain untuk *cut-set* adalah jembatan (*bridge*). Jembatan adalah himpunan sisi yang apabila dibuang dari graf menyebabkan graf tersebut tidak terhubung (menjadi dua buah komponen terhubung). Yang harus diingat, di dalam *cut-set* tidak boleh mengandung himpunan bagian yang juga *cut set*, sehingga *cut-set* yang dimaksudkan adalah *fundamental cut-set*.

Contoh 8.14

Pada Gambar 8.20, jika kita membuang sisi $(1, 2)$, graf masih tetap terhubung. Jika yang kita buang adalah $(1, 2)$ dan $(1, 5)$ graf masih tetap terhubung. Tetapi, jika kita buang sisi-sisi di dalam himpunan $\{(1,2), (1,5), (3,5), (3,4)\}$ barulah graf menjadi tidak terhubung. Jadi, $\{(1,2), (1,5), (3,5), (3,4)\}$ adalah *cut-set*. Terdapat banyak *cut-set* di dalam sebuah graf terhubung. Himpunan $\{(1,2), (2,5)\}$ juga adalah *cut-set*, $\{(1,3), (1,5), (1,2)\}$ adalah *cut-set*, $\{(2,6)\}$ juga *cut-set*, tetapi $\{(1,2), (2,5), (4,5)\}$ bukan *cut-set* sebab himpunan bagiannya, $\{(1,2), (2,5)\}$ adalah *cut-set*. ■



Gambar 8.20 $\{(1,2), (1,5), (3,5), (3,4)\}$ adalah *cut-set*.

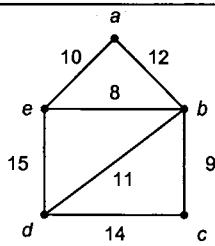
Cut-set berperanan besar dalam jaringan komunikasi dan jaringan transportasi. Misalkan keenam simpul pada Gambar 8.20(a) menyatakan enam kota yang dihubungkan dengan saluran telepon (sisi). Kita ingin menemukan apakah terdapat titik-titik lemah dalam jaringan yang memerlukan penguatan dengan alat saluran telepon tambahan. Kita lihat semua *cut-set* dari graf dan *cut-set* yang memiliki jumlah sisi paling sedikit adalah saluran yang mudah diserang/dipatahkan [DEO74].

12. Graf Berbobot (*Weighted Graph*)

DEFINISI 8.15. *Graf berbobot* adalah graf yang setiap sisinya diberi sebuah harga (bobot).

Bobot pada tiap sisi dapat berbeda-beda bergantung pada masalah yang dimodelkan dengan graf. Bobot dapat menyatakan jarak antara dua buah kota, biaya perjalanan antara dua buah kota, waktu tempuh pesan (*message*) dari sebuah simpul komunikasi ke simpul komunikasi lain (dalam jaringan komputer), ongkos produksi, dan sebagainya. Gambar 8.21 adalah contoh graf berbobot.

Istilah lain yang sering dikaitkan dengan graf berbobot adalah **graf berlabel**. Namun graf berlabel sesungguhnya lebih luas lagi definisinya. Label tidak hanya diberikan pada sisi, tetapi juga pada simpul. Sisi diberi label berupa bilangan tak-negatif, sedangkan simpul diberi label berupa data lain. Misalnya pada graf yang memodelkan kota-kota, simpul diberi nama kota-kota, sedangkan label pada sisi menyatakan jarak antara kota-kota.



Gambar 8.21 Graf berbobot

8.6 Beberapa Graf Sederhana Khusus

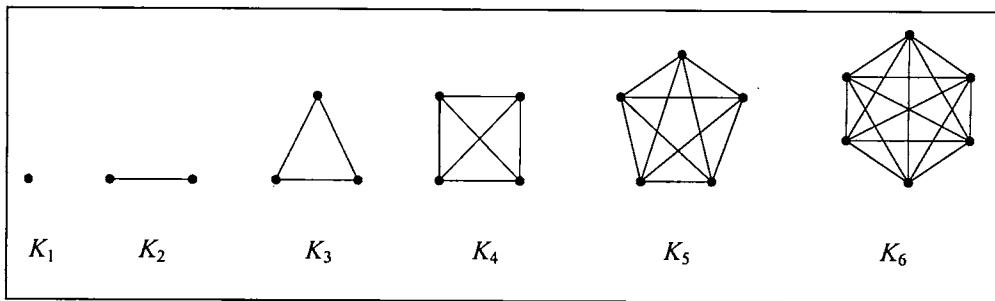
Ada beberapa graf sederhana khusus yang dijumpai pada banyak aplikasi. Beberapa di antaranya diperkenalkan di bawah ini.

a. Graf Lengkap (*Complete Graph*)

Graf lengkap ialah graf sederhana yang setiap simpulnya mempunyai sisi ke semua simpul lainnya. Graf lengkap dengan n buah simpul dilambangkan dengan K_n . Setiap simpul pada K_n berderajat $n - 1$.

Contoh 8.15

Enam buah graf lengkap, K_1 sampai K_6 , diperagakan pada Gambar 8.22. ■



Gambar 8.22 Graf lengkap K_n , $1 \leq n \leq 6$

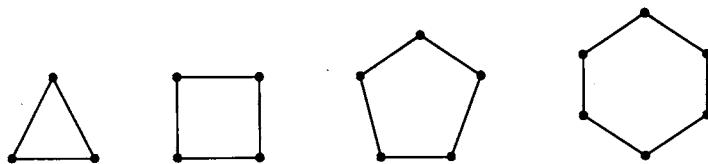
Jumlah sisi pada graf lengkap yang terdiri dari n buah simpul adalah $n(n - 1)/2$. Rumus ini diperoleh sbb: untuk 1 buah simpul terdapat $(n - 1)$ buah sisi ke $(n - 1)$ simpul lainnya, maka untuk n buah simpul terdapat $n(n - 1)$ buah sisi. Karena setiap sisi terhitung dua kali untuk pasangan simpul yang bersisisian dengannya, maka jumlah sisi seluruhnya dibagi dua, yaitu $n(n - 1)/2$.

b. Graf Lingkaran

Graf lingkaran adalah graf sederhana yang setiap simpulnya berderajat dua. Graf lingkaran dengan n simpul dilambangkan dengan C_n . Jika simpul-simpul pada C_n adalah v_1, v_2, \dots, v_n , maka sisi-sisinya adalah $(v_1, v_2), (v_2, v_3), \dots, (v_{n-1}, v_n)$, dan (v_n, v_1) . Dengan kata lain, ada sisi dari simpul terakhir, v_n , ke simpul pertama, v_1 .

Contoh 8.16

Gambar 8.23 adalah empat buah graf lingkaran. Salah satu topologi jaringan komputer area lokal (LAN) adalah topologi cincin (*ring topology*) yang direpresentasikan sebagai graf lingkaran.



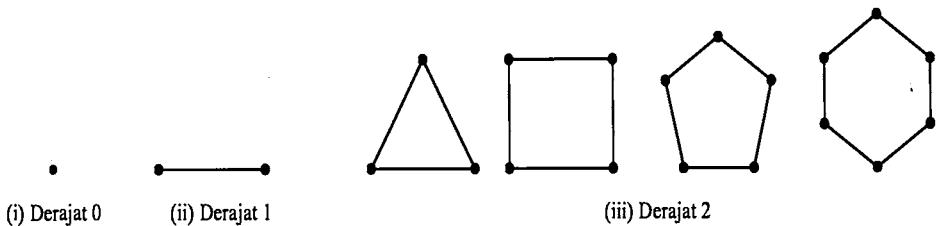
Gambar 8.23 Graf lingkaran C_n , $3 \leq n \leq 6$

c. Graf Teratur (*Regular Graphs*)

Graf yang setiap simpulnya mempunyai derajat yang sama disebut **graf teratur**. Apabila derajat setiap simpul adalah r , maka graf tersebut disebut sebagai graf teratur derajat r .

Contoh 8.17

Gambar 8.24 adalah graf teratur berderajat 0, 1, dan 2.

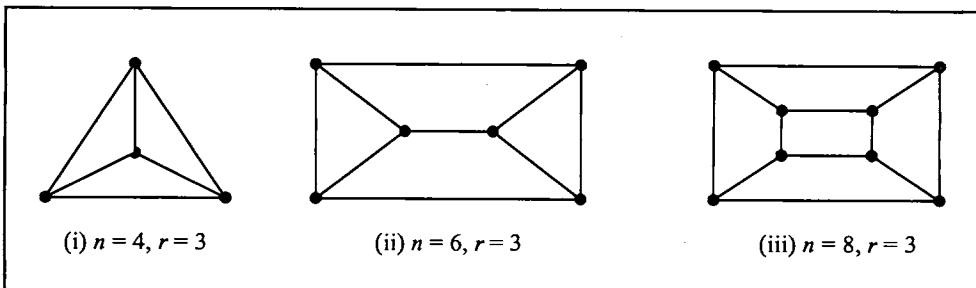


Gambar 8.24 Graf teratur derajat 0, 1, dan 2

Catatlah bahwa graf lengkap K_n juga adalah graf teratur berderajat $(n - 1)$. Demikian pula graf lingkaran C_n juga graf teratur berderajat 2. Mudah dihitung bahwa jumlah sisi pada graf teratur derajat r dengan n buah simpul adalah $nr/2$ (periksa!).

Contoh 8.18

Graf (i) pada gambar 8.25 adalah graf teratur berderajat 3 dengan 4 buah simpul, (ii) graf teratur derajat 3 dengan 6 buah simpul, dan (iii) adalah graf teratur derajat 3 dengan 8 buah simpul. ■



Gambar 8.25 Graf teratur berderajat 3, masing-masing dengan 4 , 6, dan 8 simpul

Contoh 8.19

Berapa jumlah maksimum dan jumlah minimum simpul pada graf sederhana yang mempunyai 12 buah sisi dan setiap simpul berderajat sama yang ≥ 3 ?

Penyelesaian:

Tiap simpul berderajat sama, berarti graf teratur.

Jumlah sisi pada graf teratur berderajat r adalah $e = nr/2$. Jadi, $n = 2e/r = (2)(12)/r = 24/r$

Untuk $r = 3$, jumlah simpul yang dapat dibuat adalah maksimum, yaitu $n = 24/3 = 8$

Untuk r yang lain ($r > 3$ dan r merupakan pembagi bilangan bulat dari 24),

$$r = 4 \rightarrow n = 24/4 = 6$$

$r = 6 \rightarrow n = 24/6 = 4 \rightarrow$ tidak mungkin membentuk graf sederhana

$r = 8 \rightarrow n = 24/8 = 3 \rightarrow$ tidak mungkin membentuk graf sederhana

$r = 12 \rightarrow n = 24/12 = 2 \rightarrow$ tidak mungkin membentuk graf sederhana

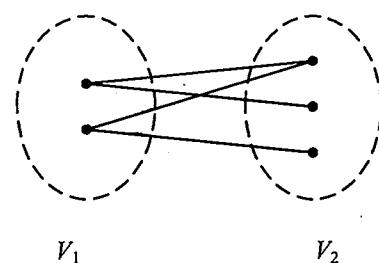
$r = 24 \rightarrow n = 24/24 = 1 \rightarrow$ tidak mungkin membentuk graf sederhana

Jadi, jumlah simpul paling sedikit 6 buah dan paling banyak 8 buah. ■

d. Graf Bipartit (*Bipartite Graph*)

Graf G yang himpunan simpulnya dapat dikelompokkan menjadi dua himpunan bagian V_1 dan V_2 , sedemikian sehingga setiap sisi di dalam G menghubungkan sebuah simpul di V_1 ke sebuah simpul di V_2 disebut **graf bipartit** dan dinyatakan sebagai $G(V_1, V_2)$. Dengan kata lain, setiap pasang simpul di V_1 (demikian pula dengan simpul-simpul di V_2) tidak bertetangga (lihat Gambar 8.26). Apabila

setiap simpul di V_1 bertetangga dengan semua simpul di V_2 , maka $G(V_1, V_2)$ disebut sebagai **graf bipartit lengkap** (*complete bipartite graph*), dilambangkan dengan $K_{m,n}$. Jumlah sisi pada graf bipartit lengkap adalah mn .

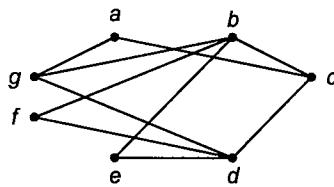


Gambar 8.26 Graf bipartit $G(V_1, V_2)$

Graf lengkap K_2 adalah graf bipartit, tetapi graf lengkap K_3 bukan graf bipartit. Untuk menunjukkan K_3 bukan graf bipartit, bagilah simpul-simpulnya menjadi dua bagian V_1 dan V_2 , yang dalam hal ini V_1 berisi satu buah simpul dan V_2 mengandung dua buah simpul. Ternyata, dua simpul di V_2 terhubung oleh sebuah sisi. Hal ini jelas tidak sesuai dengan definisi graf bipartit.

Contoh 8.20

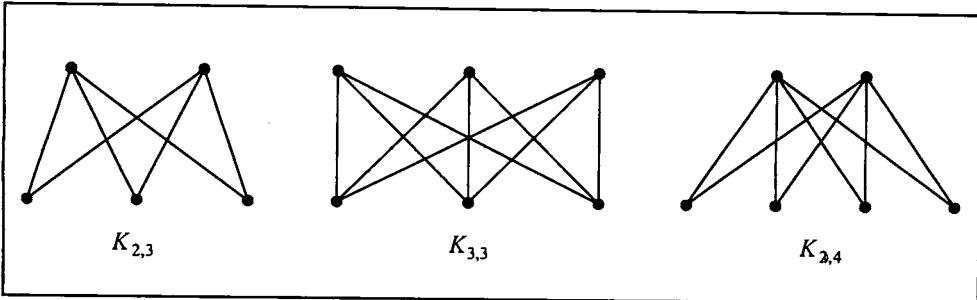
Graf G pada Gambar 8.27 adalah graf bipartit, karena simpul-simpulnya dapat dibagi menjadi $V_1 = \{a, b, d\}$ dan $V_2 = \{c, e, f, g\}$ dan setiap sisi menghubungkan simpul di V_1 ke simpul di V_2 . Dengan cara yang sama, perlihatkan bahwa C_6 adalah graf bipartit. ■



Gambar 8.27 Graf bipartit

Contoh 8.21

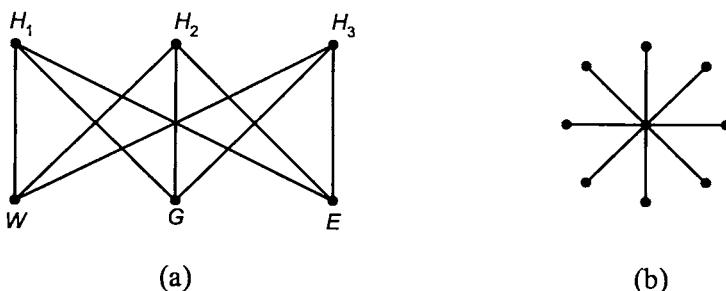
Graf G pada Gambar 8.28 adalah graf bipartit lengkap $K_{2,3}, K_{3,3}, K_{2,4}$. ■



Gambar 8.28 Graf bipartit lengkap $K_{2,3}$, $K_{3,3}$ dan $K_{2,4}$.

Contoh persoalan yang dinyatakan sebagai graf bipartit adalah persoalan utilitas: misalkan ada tiga buah rumah (Gambar 8.29(a)), H_1 , H_2 , dan H_3 , masing-masing rumah dihubungkan dengan tiga buah utilitas – air (W), gas (G), dan listrik (E) – dengan alat pengantar berupa pipa, kabel, dsb. Graf pada Gambar 8.29(a) adalah graf bipartit lengkap, $K_{3,3}$.

Contoh graf bipartit yang lain adalah topologi bintang (*star topology*) pada jaringan komputer *LAN* (Gambar 8.29(b)). Di sini V_1 berisi sebuah simpul di pusat, sedangkan V_2 berisi simpul-simpul sisanya. Catatlah bahwa graf topologi bintang dengan n simpul (n terminal komputer) adalah graf $K_{1,n}$.



Gambar 8.29 (a) Graf persoalan utilitas dan (b) topologi bintang keduanya adalah graf bipartit.

8.7 Representasi Graf

Bila graf akan diproses dengan program komputer, maka graf harus direpresentasikan di dalam memori. Terdapat beberapa representasi yang mungkin untuk graf. Di sini hanya diberikan tiga macam representasi yang sering digunakan, yaitu matriks ketetanggaan, matriks bersisian, dan senarai ketetanggaan.

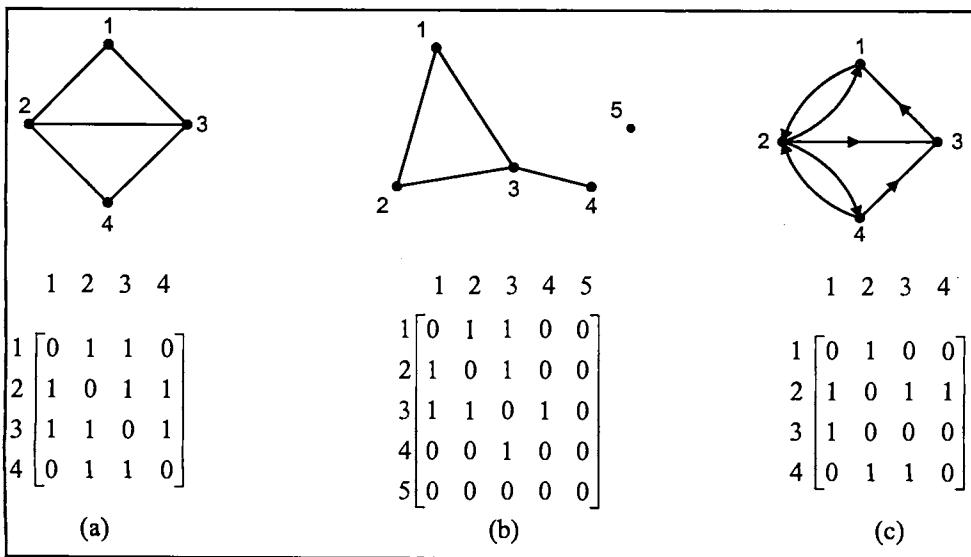
1. Matriks Ketetanggaan (*adjacency matrix*)

Matriks ketetanggaan adalah representasi graf yang paling umum. Misalkan $G = (V, E)$ adalah graf dengan n simpul, $n \geq 1$. Matriks ketetanggaan G adalah matriks dwimatra yang berukuran $n \times n$. Bila matriks tersebut dinamakan $A = [a_{ij}]$, maka $a_{ij} = 1$ jika simpul i dan j bertetangga, sebaliknya $a_{ij} = 0$ jika simpul i dan j tidak bertetangga.

Karena matriks ketetanggaan hanya berisi 0 dan 1, maka matriks tersebut dinamakan juga **matriks nol-satu** (*zero-one*). Selain dengan angka 0 dan 1, elemen matriks dapat juga dinyatakan dengan nilai *false* (menyatakan 0) dan *true* (menyatakan 1). Perhatikanlah bahwa matriks ketetanggaan didasarkan pada pengurutan nomor simpul. Di sini, terdapat $n!$ cara pengurutan nomor simpul, yang berarti ada $n!$ matriks ketetanggaan berbeda untuk graf dengan n simpul.

Contoh 8.23

Gambar 8.30 memperlihatkan graf sederhana dengan matriks ketetanggaannya, masing-masing graf terhubung, graf tak-terhubung, dan graf berarah.



Gambar 8.30 Tiga buah graf dengan matriks ketetanggaannya masing-masing.

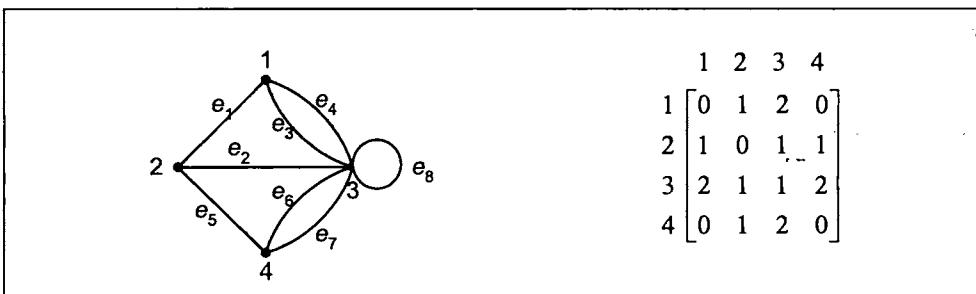
Matriks ketetanggaan untuk graf sederhana dan tidak berarah selalu simetri, sedangkan untuk graf berarah matriks ketetanggaannya belum tentu simetri (akan

simetri jika berupa graf berarah lengkap). Selain itu, diagonal utamanya selalu nol karena tidak ada sisi gelang.

Sayangnya, matriks ketetanggaan nol-satu tidak dapat digunakan untuk merepresentasikan graf yang mempunyai sisi ganda (graf ganda). Untuk menyiasatinya, maka elemen a_{ij} pada matriks ketetanggaan sama dengan jumlah sisi yang berasosiasi dengan (v_i, v_j) . Matriks ketetanggaannya tentu bukan lagi matriks nol-satu. Untuk graf semu, gelang pada simpul v_i dinyatakan dengan nilai 1 pada posisi (i, i) di matriks ketetanggaannya.

Contoh 8.24

Gambar 8.31 memperlihatkan matriks ketetanggaan untuk graf yang mengandung sisi ganda dan gelang. ■



Gambar 8.31 Matriks ketetanggaan untuk graf yang mengandung sisi ganda dan gelang.

Jumlah elemen matriis ketetanggan untuk graf dengan n simpul adalah adalah n^2 . Jika tiap elemen membutuhkan ruang memori sebesar p , maka ruang memori yang diperlukan seluruhnya adalah pn^2 . Pada matriks ketetanggaan untuk graf tak-berarah sederhana simetri, kita cukup menyimpan elemen segitiga atas saja, karena matriksnya simetri, sehingga ruang memori yang dibutuhkan dapat dihemat menjadi $pn^2/2$.

Keuntungan representasi dengan matriks ketetanggaan adalah elemen matriksnya dapat dikes langsung melalui indeks. Selain itu, kita juga dapat menentukan dengan langsung apakah simpul i dan simpul j bertetangga.

Derajat tiap simpul i dapat dihitung dari matriks ketetanggaan. Untuk graf tak-berarah,

$$d(v_i) = \sum_{j=1}^n a_{ij} \quad (8.5)$$

sedangkan untuk graf berarah,

$$d_{in}(v_j) = \text{jumlah nilai pada kolom } j = \sum_{i=1}^n a_{ij}$$

$$d_{out}(v_i) = \text{jumlah nilai pada baris } i = \sum_{j=1}^n a_{ij}$$

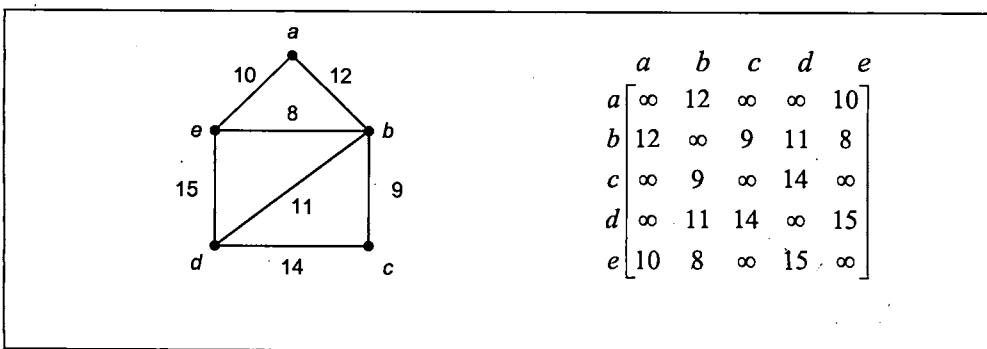
Contoh 8.23

Tinjau matriks ketetanggaan pada Gambar 8.30.

- (a) derajat simpul 2 pada Gambar 8.30(a) adalah $1 + 0 + 1 + 1 = 3$
derajat simpul 4 pada Gambar 8.30(a) adalah $0 + 1 + 1 + 0 = 2$
- (b) derajat simpul 4 pada Gambar 8.30(b) adalah $0 + 0 + 1 + 0 + 0 = 1$
derajat simpul 5 pada Gambar 8.30(b) adalah $0 + 0 + 0 + 0 + 0 = 0$
- (c) derajat-masuk simpul 2 pada Gambar 8.30(c) adalah $1 + 0 + 0 + 1 = 2$
derajat-keluar simpul 2 pada Gambar 8.30(c) adalah $1 + 0 + 1 + 1 = 3$ ■

Contoh 8.25

Untuk graf berbobot, a_{ij} menyatakan bobot tiap sisi yang menghubungkan simpul i dengan simpul j . Gambar 8.32 adalah graf berbobot beserta matriks ketetanggaannya. ■



Gambar 8.32 Graf berbobot (kiri) dan (b) matriks ketetanggaannya.

Tanda “ ∞ ” menyatakan bahwa tidak ada sisi dari simpul i ke simpul j atau dari simpul i ke simpul i itu sendiri, sehingga a_{ij} dapat diberi nilai tak berhingga.

2. Matriks Bersisian (*incidence matrix*)

Bila matriks ketetanggaan menyatakan ketetanggaan simpul-simpul di dalam graf, maka matriks bersisian menyatakan kebersisian simpul dengan sisi. Misalkan $G = (V, E)$ adalah graf dengan n simpul dan m buah sisi. Matriks bersisian G adalah matriks dwimatra yang berukuran $n \times m$. Baris menunjukkan label simpul, sedangkan kolom menunjukkan label sisinya. Bila matriks tersebut dinamakan $A = [a_{ij}]$, maka $a_{ij} = 1$ jika simpul i berseisian dengan sisi j , sebaliknya $a_{ij} = 0$ jika simpul i tidak bersisian dengan sisi j .

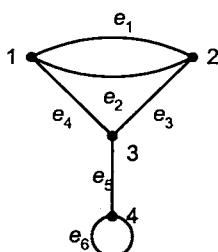
Matriks bersisian dapat digunakan untuk merepresentasikan graf yang mengandung sisi ganda atau sisi gelang.

Derajat setiap simpul i dapat dihitung dengan menghitung jumlah seluruh elemen pada baris i (kecuali pada graf yang mengandung gelang).

Jumlah elemen matriks bersisian adalah nm . Jika tiap elemen membutuhkan ruang memori sebesar p , maka ruang memori yang diperlukan seluruhnya adalah pnm .

Contoh 8.26.

Gambar 8.33 memperlihatkan matriks bersisian untuk graf yang direpresentasikannya. Jumlah elemen matriks adalah $4 \times 6 = 24$. ■



	e_1	e_2	e_3	e_4	e_5	e_6
1	1	1	0	1	0	0
2	1	1	1	0	0	0
3	0	0	1	1	1	0
4	0	0	0	0	1	1

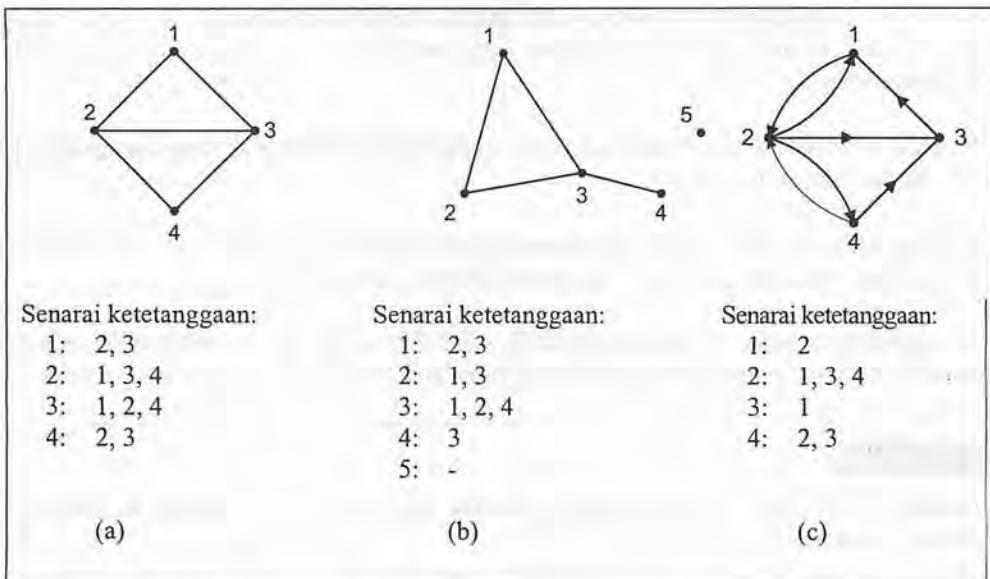
Gambar 8.33 Graf (kiri) dan matriks bersisianya (kanan)

3. Senarai Ketetanggaan (*adjacency list*)

Kelemahan matriks ketetanggaan adalah bila graf memiliki jumlah sisi relatif sedikit, karena matriksnya bersifat jarang (*sparse*), yaitu mengandung banyak elemen nol, sedangkan elemen yang bukan nol sedikit. Ditinjau dari implementasinya di dalam komputer, kebutuhan ruang memori untuk matriks jarang boros karena komputer menyimpan elemen 0 yang tidak perlu. Untuk mengatasi masalah ini, kita menggunakan representasi yang ketiga, yaitu senarai ketetanggaan. Senarai ketetanggaan mengenumerasi simpul-simpul yang bertetangga dengan setiap simpul di dalam graf.

Contoh 8.27

Gambar 8.34 memperlihatkan graf tak-berarah dan graf berarah beserta senarai ketetanggaannya masing-masing. ■



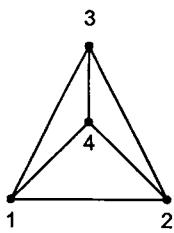
Gambar 8.34 Tiga buah graf dengan matriks ketetanggaannya masing-masing.

8.8 Graf Isomorfik (*Isomorphic Graph*)

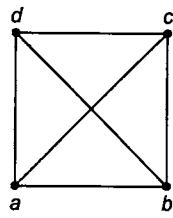
Misalkan kepada dua orang mahasiswa diminta menggambarkan sebuah graf dengan empat buah simpul, masing-masing simpul berderajat tiga. Graf yang digambar bisa beragam bentuknya, dua diantaranya ditunjukkan pada Gambar 8.35(a) dan 8.35(b). Meskipun kedua graf tersebut terlihat berbeda bentuknya, dengan penamaan simpul yang berbeda pula, namun sebenarnya keduanya merupakan graf yang sama. Dua buah graf yang sama tetapi secara geometri berbeda disebut graf yang saling **isomorfik**.

DEFINISI 8.16. Dua buah graf, G_1 dan G_2 dikatakan isomorfik jika terdapat korespondensi satu-satu antara simpul-simpul keduanya dan antara sisi-sisi keduanya sedemikian sehingga jika sisi e bersisian dengan simpul u dan v di G_1 , maka sisi e' yang berkorespon di G_2 juga harus bersisian dengan simpul u' dan v' di G_2 .

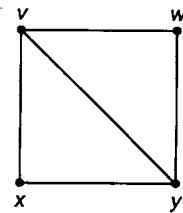
Dua buah graf yang isomorfik adalah graf yang sama, kecuali penamaan simpul dan sisinya saja yang berbeda. Ini benar karena sebuah graf dapat digambarkan dalam banyak cara [DEO74].



(a) G_1



(b) G_2



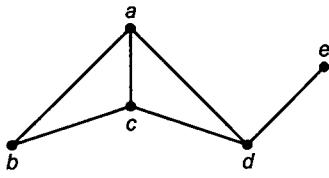
(c) G_3

Gambar 8.35 G_1 isomorfik dengan G_2 , tetapi G_1 tidak isomorfik dengan G_3

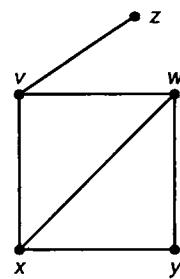
Pada Gambar 8.35, G_1 isomorfik dengan G_2 . Simpul 1, 2, 3, dan 4 di G_1 berkoresponden dengan simpul a, b, c , dan d di G_2 . Sisi $(1, 2), (2, 3), (3, 1), (3, 4)$, $(1, 4)$, dan $(2, 4)$ berkoresponden dengan sisi $(a, b), (b, c), (c, d), (a, d), (a, c)$, dan (b, d) . Semua simpul di G_1 dan G_2 berderajat 3. G_1 maupun G_2 tidak isomorfik dengan G_3 , karena simpul-simpul di G_3 dua buah berderajat dua dan dua buah lagi berderajat tiga, sedangkan simpul-simpul di G_1 dan G_2 semuanya berderajat tiga.

Dua buah graf pada Gambar 8.36 di bawah ini juga isomorfik. Simpul a, b, c, d , dan e di G_1 masing-masing berkoresponden dengan simpul x, y, w, v , dan z di G_2 . Masing-masing simpul yang disebutkan itu berderajat 3, 2, 3, 3, dan 1. Periksa pula bahwa sisi-sisi di G_1 berkoresponden dengan sisi-sisi di G_2 .

Contoh-contoh graf lain yang isomorfik diperlihatkan pada Gambar 8.37.

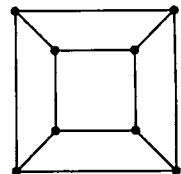
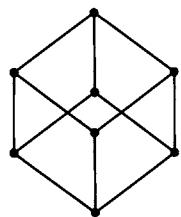


(a) G_1

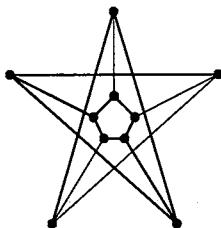
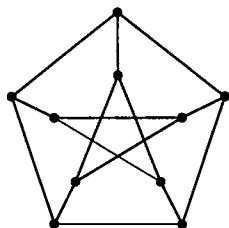


(b) G_2

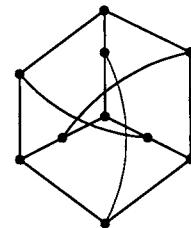
Gambar 8.36 Graf (a) dan graf (b) isomorfik [DEO74]



(a)



(b)



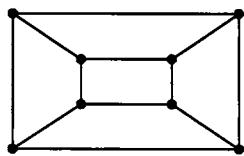
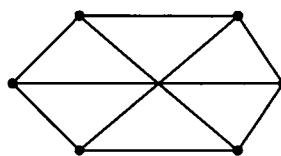
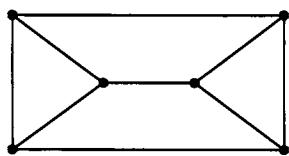
Gambar 8.37 (a) Dua buah graf isomorfik, (b) tiga buah graf isomorfik [DEO74]

Contoh 8.28

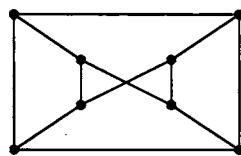
Gambarkan graf yang isomorfik dengan masing-masing graf teratur pada Gambar 8.38(i).

Penyelesaian:

Solusi untuk persoalan ini tidak tunggal, sebab banyak cara untuk menggambarkan graf isomorfiknya. Salah satunya ditunjukkan pada Gambar 8.38 (ii). ■



(i)



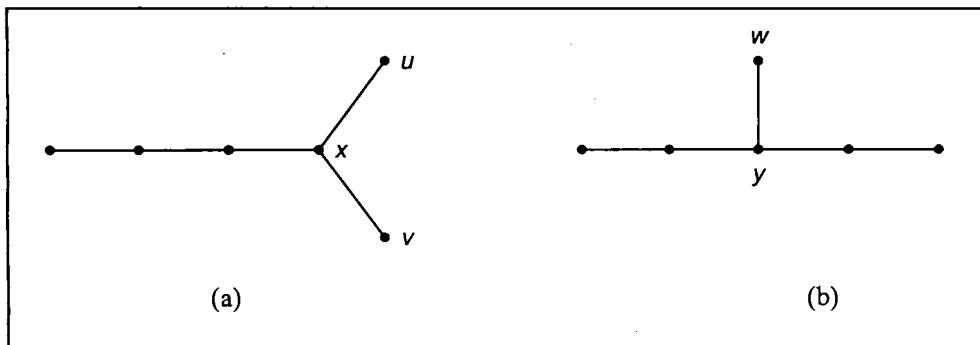
(ii)

Gambar 8.38 (i) Dua buah graf teratur, (ii) graf isomorfiknya [LIP92]

Tidak mudah menentukan apakah dua buah graf isomorfik hanya dengan melihat gambarnya saja. Dari definisi isomorfik kita menyimpulkan dua buah graf isomorfik memenuhi ketiga syarat berikut [DEO74]:

1. Mempunyai jumlah simpul yang sama.
2. Mempunyai jumlah sisi yang sama
3. Mempunyai jumlah simpul yang sama berderajat tertentu

Namun, ketiga syarat ini ternyata belum cukup menjamin keisomorfikan. Pemeriksaan secara visual masih tetap diperlukan. Contohnya, dua buah graf pada Gambar 8.39 memenuhi ketiga syarat yang disebutkan di atas, padahal keduanya tidak isomorfik. Di (a) terdapat dua simpul anting-anting (berderajat 1) yang bertetangga dengan simpul x , sedangkan di (b) hanya terdapat satu buah simpul anting-anting yang bertetangga dengan y .



Gambar 8.39 Dua buah graf yang tidak isomorfik [DEO74]

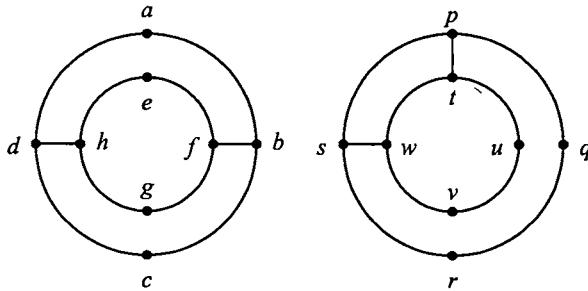
Contoh 8.29

Manakah di antara pasangan graf pada Gambar 8.40(a) isomorfik?

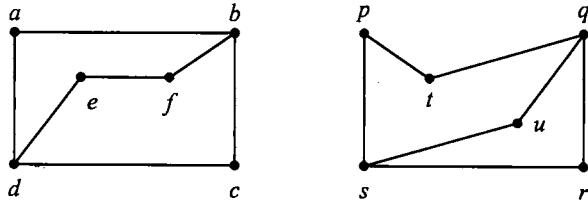
Penyelesaian:

- (a) Kedua graf pada Gambar 8.40(a) tidak isomorfik karena tidak ada korespondensi satu-satu antara simpul-simpul pada kedua graf. bertetangga. Tinjau misalnya simpul d ; simpul d bertetangga dengan dua buah simpul berderajat 2 (a dan c) dan sebuah simpul berderajat 3 (h). Graf di sebelah kanannya tidak mempunyai simpul yang berkoresponden dengan d (jika s dianggap sebagai simpul yang berkoresponden dengan d , maka ini jelas tidak benar, sebab s bertetangga dengan sebuah simpul berderajat 2 (r) dan dua buah simpul berderajat 3 (p dan w).
- (b) Kedua graf pada Gambar 8.40(a) isomorfik sebab terdapat korespondensi satu-satu antara simpul pada graf sebelah kiri dengan simpul-simpul pada graf sebelah kanan, yaitu

- a* berkoresponden dengan u ;
 - b* berkoresponden dengan q ;
 - c* berkoresponden dengan r ;
 - d* berkoresponden dengan s ;
 - e* berkoresponden dengan p ;
 - f* berkoresponden dengan t .



(a)



(b)

Gambar 8.40 (a) kedua graf tidak isomorfik, (b) kedua graf isomorfik

Untuk memperlihatkan bahwa dua buah graf isomorfik, kita dapat menunjukkan bahwa matriks ketetanggaannya kedua graf itu sama. Matriks ketetanggaan untuk dua buah graf yang isomorfik pada Gambar 8.36 adalah seperti di bawah ini.

$$A_{G1} = \begin{matrix} & \begin{matrix} a & b & c & d & e \end{matrix} \\ \begin{matrix} a \\ b \\ c \\ d \\ e \end{matrix} & \begin{bmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \end{matrix} \quad A_{G2} = \begin{matrix} & \begin{matrix} x & y & w & v & z \end{matrix} \\ \begin{matrix} x \\ y \\ w \\ v \\ z \end{matrix} & \begin{bmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \end{matrix}$$

Terlihat bahwa matriks ketetanggaan G_1 dan G_2 keduanya sama, yang berarti kedua graf tersebut isomorfik.

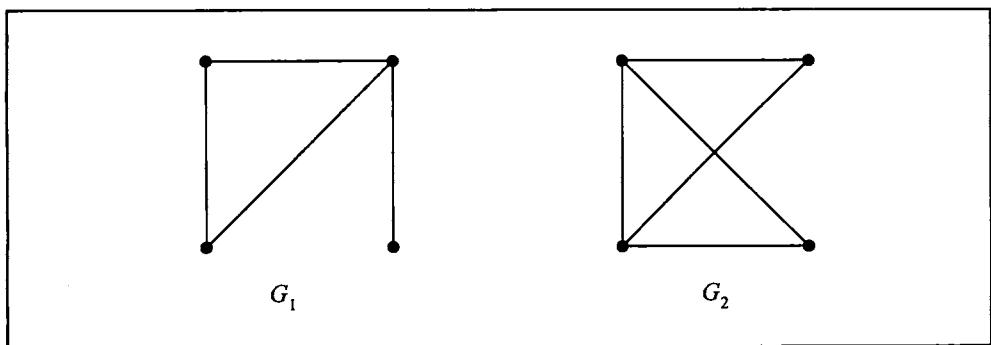
Contoh 8.30

Apakah graf sederhana yang disajikan oleh pasangan matriks ketetanggaan di bawah ini isomorfik? Jelaskan jawaban, lalu gambarkan grafnya!

$$\begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}; \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

Penyelesaian:

Keduanya tidak isomorfik karena jumlah sisi pada graf pertama dan graf kedua tidak sama. Graf pertama (G_1) mempunyai 4 buah sisi (hitung jumlah elemen 1 di atas diagonal utama), sedangkan graf kedua (G_2) mempunyai 5 buah sisi (hitung jumlah elemen 1 di atas diagonal utama). Gambar 8.41 memperlihatkan kedua graf yang bersesuaian dengan masing-masing matriks ketetanggaan di atas. ■



Gambar 8.41 Graf yang merepresentasikan masing-masing matriks ketetanggaan pada Contoh 8.30

Sampai saat ini belum ada algoritma yang mangkus untuk memeriksa apakah dua buah graf isomorfik. Algoritma terbaik untuk memeriksa apakah dua buah graf isomorfik mempunyai kompleksitas waktu eksponensial (bergantung pada jumlah simpul di dalam graf) pada kasus terburuk [ROS99]. Semakin banyak simpul graf, kebutuhan waktu algoritma meningkat sangat drastis.

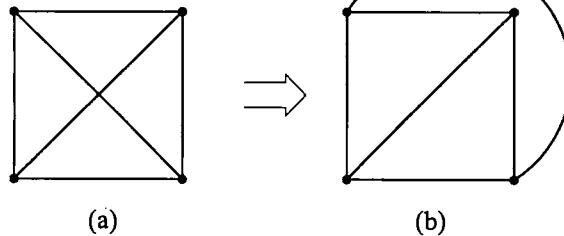
8.9 Graf Planar dan Graf Bidang

DEFINISI 8.17. Graf yang dapat digambarkan pada bidang datar dengan sisi-sisi yang tidak saling memotong (bersilangan) disebut sebagai **graf planar**, jika tidak, maka ia disebut **graf tak-planar**.

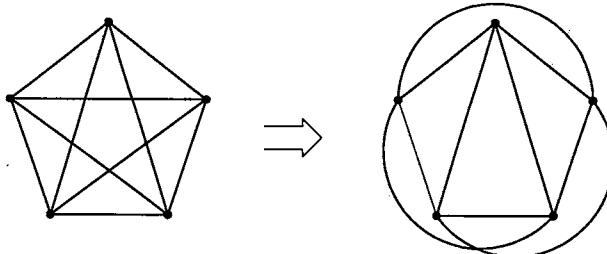
Perlu diperhatikan bahwa belum tentu suatu graf yang secara kasat mata terlihat sisi-sisinya saling berpotongan tidak planar. Graf tersebut mungkin saja planar, karena graf tersebut dapat digambarkan kembali dengan cara berbeda yang sisi-sisinya tidak saling berpotongan.

Contoh 8.31

Graf K_4 adalah graf planar, biasanya digambarkan dengan sisi yang bersilangan seperti ditunjukkan pada Gambar 8.42(a). Kita dapat menggambarkan graf itu kembali tanpa ada sisi-sisi yang berpotongan, seperti pada Gambar 8.42(b). K_5 pada Gambar 8.43 bukan graf planar. ■



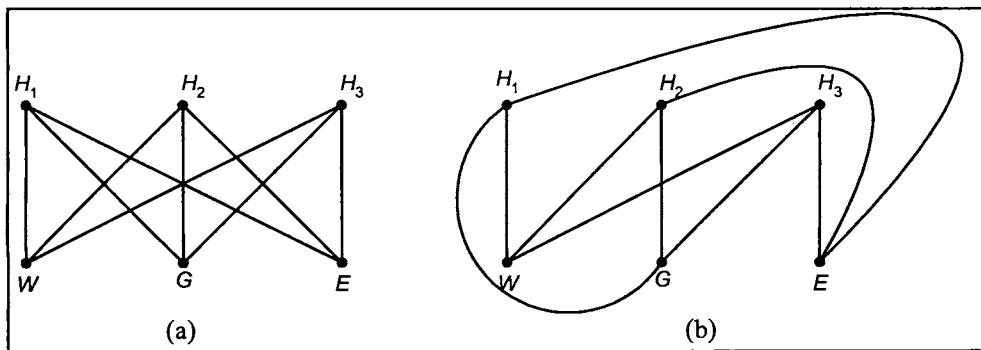
Gambar 8.42 K_4 adalah graf planar



Gambar 8.43 K_5 bukan graf planar

Contoh 8.32

Tinjau kembali persoalan utilitas: terdapat tiga buah rumah (*Gambar 8.44(a)*), H_1 , H_2 , dan H_3 , masing-masingnya dihubungkan tiga buah utilitas –air (*W*), gas (*G*), dan listrik (*E*) – dengan alat pengantar (pipa, kabel, dsb). Graf pada *Gambar 8.44(a)* tersebut juga graf bipartit lengkap, $K_{3,3}$. Mungkinkah membangun jaringan utilitas sehingga tidak ada pengantar yang saling berpotongan? (sebab kalau saling berpotongan, dikhawatirkan timbul masalah yang serius, misalnya bila kabel listrik berpotongan dengan pipa gas dapat terjadi ledakan). Jika graf pada *Gambar 8.44(a)* digambar ulang, ternyata tidak mungkin menggambar sisi yang tidak saling berpotongan (*Gambar 8.44(b)*). Dengan kata lain, graf persoalan utilitas tidak planar.



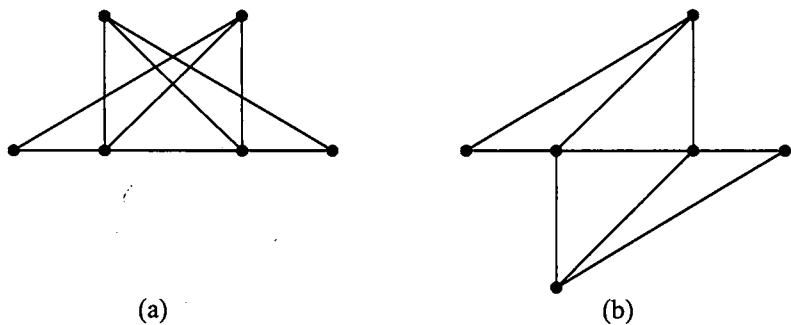
Gambar 8.44 (a) Graf persoalan utilitas ($K_{3,3}$), (b) graf persoalan utilitas bukan graf planar.

Contoh 8.33

[LIP92] Gambarkan graf planar pada *Gambar 8.45(a)* sehingga tidak ada sisi-sisi yang berpotongan (menjadi graf bidang).

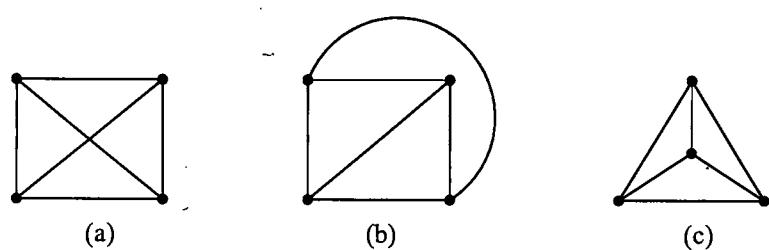
Penyelesaian:

Susun kembali posisi salah satu simpul untuk mendapatkan sebuah solusi yang ditunjukkan pada *Gambar 8.45(b)*.



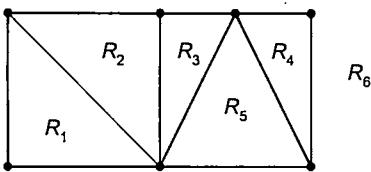
Gambar 8.45 (a) Graf dengan sisi-sisi yang berpotongan,
(b) setelah digambar ulang tanpa sisi yang berpotongan

Representasi graf planar yang digambarkan dengan sisi-sisi yang tidak saling berpotongan disebut **graf bidang** (*plane graph*). Pada Gambar 8.46, ketiga buah graf adalah graf planar, tetapi graf (a) bukan graf bidang, sedangkan graf (b) dan (c) adalah graf bidang. Ketiga graf ini isomorfik. Untuk selanjutnya, kita tetap menggunakan istilah graf planar baik untuk graf yang dapat digambar (ulang) pada bidang datar tanpa ada sisi-sisi yang berpotongan maupun graf yang memang sudah tergambar tanpa sisi-sisi yang berpotongan (graf bidang).



Gambar 8.46 Tiga buah graf planar. Graf (b) dan (c) adalah graf bidang.

- Sisi-sisi pada graf bidang membagi bidang datar menjadi beberapa wilayah (*region*) atau muka (*face*). Jumlah wilayah pada graf bidang dapat dihitung dengan mudah. Graf bidang pada Gambar 8.47 terdiri atas 6 wilayah (termasuk wilayah terluar):



Gambar 8.47 Graf planar yang terdiri atas 6 wilayah

Rumus Euler

Jumlah wilayah (f) pada graf planar sederhana juga dapat dihitung dengan rumus Euler sebagai berikut:

$$n - e + f = 2 \quad (8.5)$$

atau

$$f = e - n + 2 \quad (8.6)$$

yang dalam hal ini,

e = jumlah sisi

n = jumlah simpul

Contoh 8.34

Pada Gambar 8.47 di atas, $e = 11$ dan $n = 7$, maka $f = 11 - 7 + 2 = 6$.

Contoh 8.35

Misalkan graf sederhana planar dan terhubung memiliki 24 buah simpul, masing-masing simpul berderajat 4. Representasi planar dari graf tersebut membagi bidang datar menjadi sejumlah wilayah atau muka. Berapa banyak wilayah yang terbentuk?

Penyelesaian:

Diketahui n = jumlah simpul = 24, maka jumlah derajat seluruh simpul = $24 \times 4 = 96$. Menurut lemma jabat tangan, jumlah derajat = $2 \times$ jumlah sisi, sehingga

$$\text{jumlah sisi} = e = \text{jumlah derajat}/2 = 96/2 = 48$$

Dari rumus Euler, $n - e + f = 2$, sehingga $f = \text{jumlah wilayah} = 2 - n + e = 2 - 24 + 48 = 26$ buah.

Pada graf planar sederhana terhubung dengan f wilayah, n buah simpul, dan e buah sisi (dengan $e > 2$) selalu berlaku ketidaksamaan berikut:

$$e \geq 3f/2$$

dan

$$e \leq 3n - 6$$

Dua ketidaksamaan yang terakhir ini dapat kita buktikan sebagai berikut: setiap daerah pada graf planar dibatasi oleh tiga atau lebih sisi. Jadi, total banyaknya sisi lebih besar atau sama dengan $3f$. Tetapi, karena suatu sisi berada pada batas paling banyak dua wilayah, maka total banyaknya sisi lebih kecil atau sama dengan $2e$. Jadi,

$$2e \geq 3f$$

atau

$$2e/3 \geq f$$

Berdasarkan rumus Euler, kita memperoleh

$$n - e + 2e/3 \geq 2$$

atau

$$e \leq 3n - 6$$

Ketidaksamaan yang terakhir dinamakan **ketidaksamaan Euler**, yang dapat digunakan untuk menunjukkan keplanaran suatu graf sederhana (kalau graf planar, maka ia memenuhi ketidaksamaan Euler, sebaliknya jika tidak planar maka ketidaksamaan tersebut tidak dipenuhi). Ini dinyatakan dengan *corollary* berikut:

COROLLARY 8.1 Jika G adalah graf sederhana terhubung dengan e adalah jumlah sisi dan v adalah jumlah simpul, yang dalam hal ini $v \geq 3$, maka berlaku ketidaksamaan Euler $e \leq 3v - 6$.

Contoh 8.36

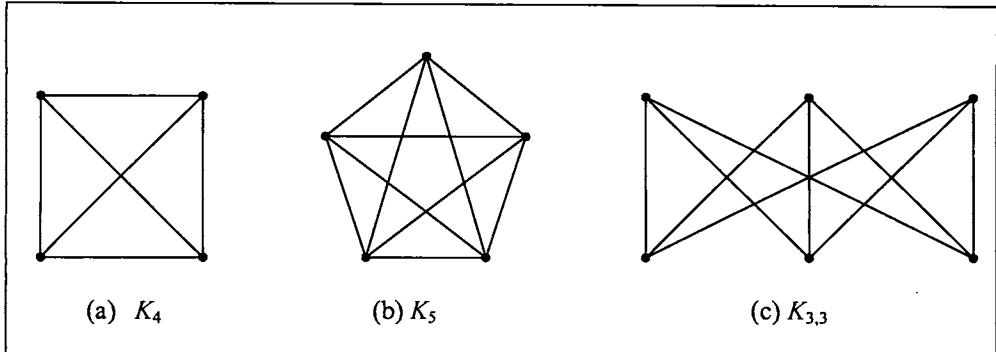
Pada graf K_4 (Gambar 8.48a), $n = 4$, $e = 6$, memenuhi $6 \leq 3(4) - 6$. Dengan kata lain, K_4 adalah graf planar. ■

Contoh 8.37

Perlihatkan bahwa K_5 (Gambar 8.48b), tidak planar dengan ketidaksamaan Euler.

Penyelesaian:

Pada graf K_5 , $n = 5$ dan $e = 10$. K_5 tidak memenuhi ketidaksamaan Euler sebab $10 \geq 3(5) - 6$. Hal ini menunjukkan bahwa K_5 tidak planar. ■



Gambar 8.48 (a) Graf lengkap K_3 , (b) K_5 , dan (c) graf bipartit $K_{3,3}$.

Sayangnya, ketidaksaamaan Euler hanyalah syarat *perlu* agar suatu graf dikatakan planar, tetapi bukan syarat *cukup* (ingat kembali mengenai makna syarat cukup dan syarat perlu pada pembahasan implikasi di dalam Bab 1). Artinya, meskipun suatu graf planar sederhana memenuhi kedua ketidaksamaan itu, tetapi tidak selalu menjamin keplanaran suatu graf. Misalnya graf $K_{3,3}$ (Gambar 8.48c) memenuhi ketidaksamaan Euler tersebut,

$$e = 9, n = 6$$

$$9 \leq 3(6) - 6 = 12 \quad (\text{jadi, } e \leq 3n - 6)$$

padahal graf $K_{3,3}$ bukan graf planar! Untuk menunjukkan bahwa $K_{3,3}$ bukan graf planar, kita membuat asumsi baru bahwa setiap wilayah pada graf bidang dibatasi oleh paling sedikit empat buah sisi (jadi, bukan 3 sisi seperti pembuktian ketidaksamaan di atas). Dengan demikian, total banyaknya sisi lebih besar atau sama dengan $4f$. Tetapi, karena suatu sisi berada pada batas paling banyak dua wilayah, maka total banyaknya sisi lebih kecil atau sama dengan $2e$. Jadi,

$$2e \geq 4f$$

atau

$$e/2 \geq f$$

Berdasarkan rumus Euler, kita memperoleh

$$n - e + e/2 \geq 2$$

atau

$$e \leq 2n - 4$$

Hal ini dinyatakan dengan *corollary* berikut:

COROLLARY 8.2 Jika G adalah graf sederhana terhubung dengan e adalah jumlah sisi dan v adalah jumlah simpul, yang dalam hal ini $v \geq 3$ dan tidak ada sirkuit yang panjangnya 3, maka berlaku $e \leq 2v - 4$.

Contoh 8.38

Graf $K_{3,3}$ tidak memenuhi ketidaksamaan $e \leq 2n - 4$, karena

$$\begin{aligned} e &= 9, n = 6 \\ 9 &\leq (2)(6) - 4 = 8 \quad (\text{salah}) \end{aligned}$$

yang berarti $K_{3,3}$ bukan graf planar. ■

Teorema Kuratowski

Berikut ini diberikan teorema dari Kuratowski yang memungkinkan kita menentukan dengan tegas keplanaran suatu graf.

Dalam literatur tentang graf, dikenal dua buah graf tidak-planar yang khusus, yaitu graf Kuratowski, setelah matematikawan Polandia, Kasimir Kuratowski, menemukan sifatnya yang unik [DEO74].

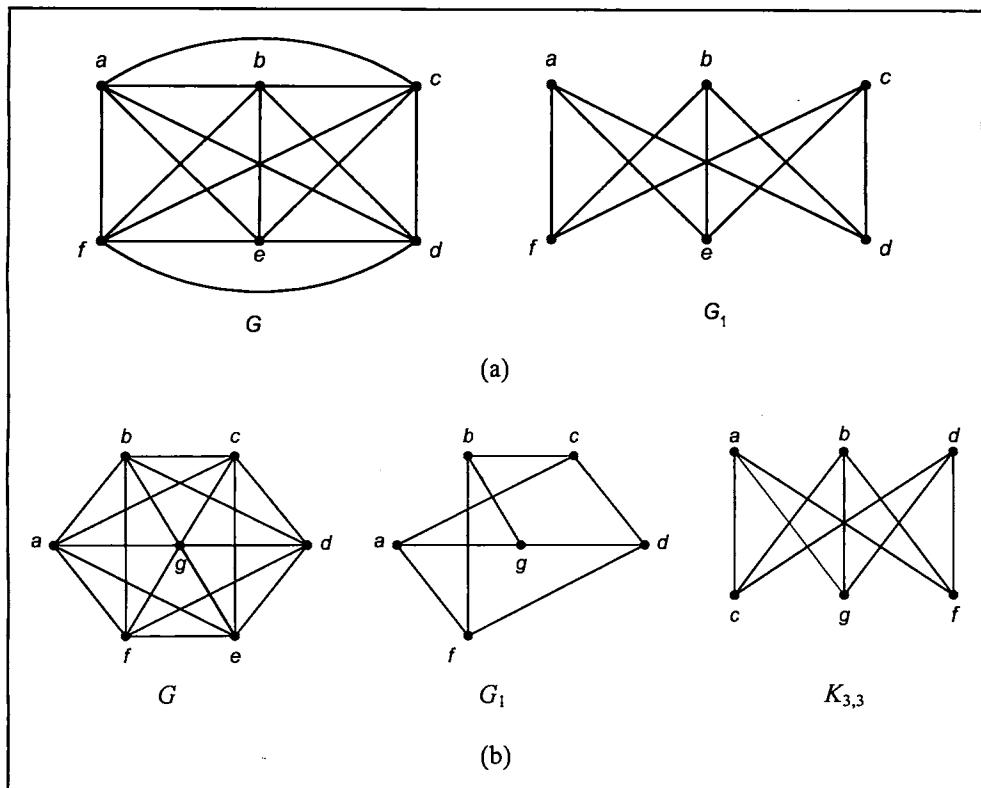
1. Graf Kuratowski pertama, yaitu graf lengkap yang mempunyai lima buah simpul (K_5), adalah graf tidak-planar.
2. Graf Kuratowski kedua, yaitu graf terhubung teratur dengan 6 buah simpul dan 9 buah sisi ($K_{3,3}$) adalah graf tidak-planar.

Sifat graf Kuratowski adalah:

1. Kedua graf Kuratowski adalah graf teratur.
2. Kedua graf Kuratowski adalah graf tidak-planar
3. Penghapusan sisi atau simpul dari graf Kuratowski menyebabkannya menjadi graf planar.
4. Graf Kuratowski pertama adalah graf tidak-planar dengan jumlah simpul minimum, dan graf Kuratowski kedua adalah graf tidak-planar dengan jumlah sisi minimum. Keduanya adalah graf tidak planar paling sederhana.

TEOREMA 8.2. (Teorema Kuratowski) Graf G adalah tidak planar jika dan hanya jika ia mengandung upagraf yang isomorfik dengan K_5 atau $K_{3,3}$ atau homeomorfik (homeomorphic) dengan salah satu dari keduanya.

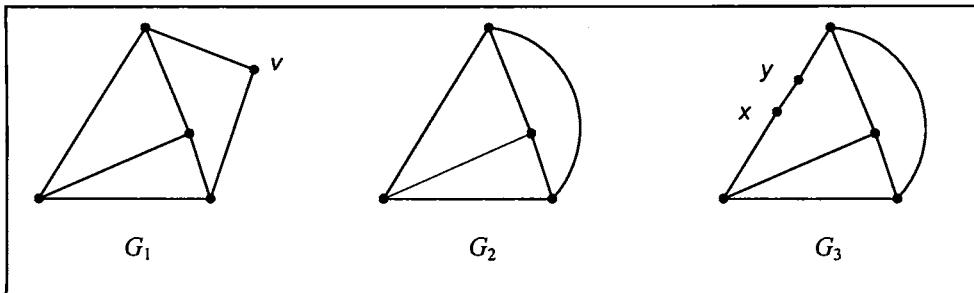
Isomorfisme graf sudah kita pahami dengan baik. Bila dua graf, G_1 dan G_2 , isomorfik berarti kedua graf tersebut sama, hanya penggambarannya saja yang berbeda. Tinjau graf G pada Gambar 8.49(a). Sangat jelas G mengandung upagraf G_1 yang sama dengan $K_{3,3}$, karena itu G tidak planar. Sekarang tinjau graf G pada Gambar 8.49(b). G mengandung upagraf G_1 yang isomorfik dengan $K_{3,3}$. Jadi, G juga tidak planar.



Gambar 8.49 (a) Graf G tidak planar karena ia mengandung upagraf, G_1 , yang sama dengan $K_{3,3}$.
 (b) Graf G tidak planar karena upagrafnya, G_1 , isomorfik dengan $K_{3,3}$.

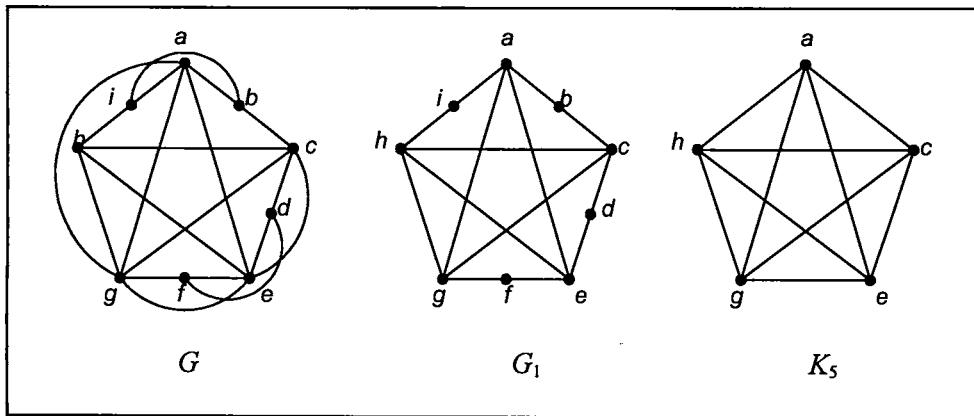
Apa yang dimaksud dengan homeomorfik?

Dua graf G_1 dan G_2 dikatakan homeomorfik jika salah satu dari kedua graf dapat diperoleh dari graf yang lain dengan cara menyisipkan dan/atau membuang secara berulang-ulang simpul berdererajat 2, seperti yang diilustrasikan pada Gambar 8.50. Ketiga graf pada Gambar 8.50 homeomorfik satu sama lain. G_2 diperoleh dengan membuang simpul v (yang berderajat 2) dari G_1 , sedangkan G_3 diperoleh dari G_2 dengan berturut-turut menambahkan simpul x dan y (yang masing-masing berderajat 2). Catatlah bahwa penambahan dan penghapusan simpul hanya dilakukan terhadap simpul berderajat dua saja.



Gambar 8.50 Tiga buah graf yang homeomorfik satu sama lain.

Tinjau Gambar 8.51. Graf G tidak planar karena ia mengandung upagraf (G_1) yang homeomorfik dengan K_5 (dengan membuang simpul-simpul yang berderajat 2 dari G_1 , diperoleh K_5).



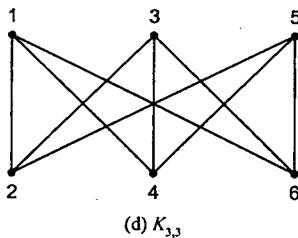
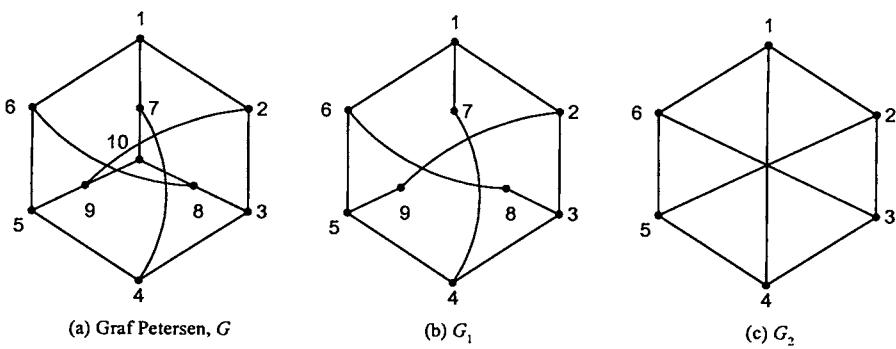
Gambar 8.51 Graf G tidak planar karena upagrafnya, G_1 , homeomorfik dengan K_5 .

Contoh 8.39

Perlihatkan dengan teorema Kuratowski bahwa graf Petersen (Gambar 8.52(a)) tidak planar.

Penyelesaian:

Dari graf Petersen (Gambar 8.52(a)), buatlah sebuah upagrafnya, misalkan G_1 (Gambar 8.52(b)). Dengan membuang simpul-simpul berderajat 2 dari G_1 , kita dapatkan G_2 (Gambar 8.52(c)) yang homeomorfik dengan G_1 . Jika G_2 digambar ulang, kita dapatkan bahwa G_2 isomorfik dengan $K_{3,3}$ (Gambar 8.52(d)). Rangkaian proses ini menunjukkan bahwa graf Petersen tidak planar. ■



Gambar 8.52

- (a) Graf Petersen
- (b) G_1 adalah upagraf dari G
- (c) G_2 homeomorfik dengan G_1
- (d) G_2 isomorfik dengan $K_{3,3}$

Perhatikanlah bahwa teorema Kuratowski lebih mudah digunakan untuk menentukan bahwa sebuah graf tidak planar. Untuk membuktikan bahwa suatu graf planar relatif lebih sulit, karena kita harus mencoba semua kemungkinan upagraf yang memiliki 5 simpul dengan 10 sisi atau upagraf yang memiliki 6 simpul dan 9 sisi, dan memeriksa apakah upagraf tersebut sama atau sama atau homeomorfik dengan K_5 atau $K_{3,3}$.

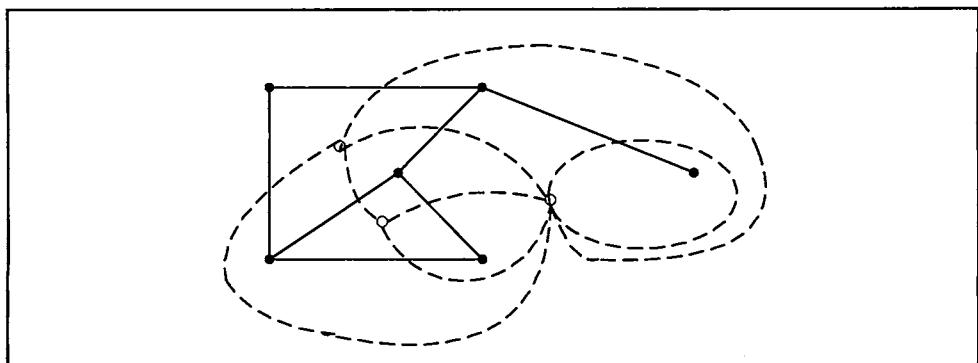
Terapan graf planar sudah kita sebutkan sebelum ini, yaitu persoalan utilitas dalam merancang jaringan pipa air, pipa gas, dan kabel listrik bawah tanah agar ketiganya tidak saling bersilangan. Terapan graf planar lainnya adalah pada perancangan *integrated circuit (IC)* pada sebuah papan. Kawat-kawat yang menghubungkan simpul-simpul IC harus dirancang sedemikian rupa agar tidak bersilangan, sebab persilangan dua buah kawat yang beraliran listrik dapat menimbulkan interferensi arus, yang mengakibatkan terganggunya fungsi IC tersebut.

8.10 Graf Dual (*Dual Graph*)

Misalkan kita mempunyai sebuah graf planar G yang direpresentasikan sebagai graf bidang. Kita dapat membuat suatu graf G^* yang secara geometri merupakan dual dari graf planar tersebut dengan cara sebagai berikut:

1. Pada setiap wilayah atau muka (*face*) f di G , buatlah sebuah simpul v^* yang merupakan simpul untuk G^* .
2. Untuk setiap sisi e di G , tariklah sisi e^* (yang menjadi sisi untuk G^*) yang memotong sisi e tersebut. Sisi e^* menghubungkan dua buah simpul v_1^* dan v_2^* (simpul-simpul di G^*) yang berada di dalam muka f_1 dan f_2 yang dipisahkan oleh sisi e di G . Untuk sisi e yang salah satu simpulnya merupakan simpul berderajat 1 (jadi, sisi e seluruhnya terdapat di dalam sebuah muka), maka sisi e^* adalah berupa sisi gelang.

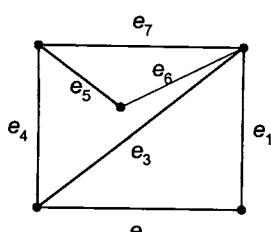
Graf G^* yang terbentuk dengan cara penggambaran demikian disebut **graf dual** (atau tepatnya **dual geometri**) dari graf G . Pada Gambar 8.53 digambarkan graf dual G^* dari graf planar G . Sisi-sisi graf G^* digambarkan dengan garis putus-putus.



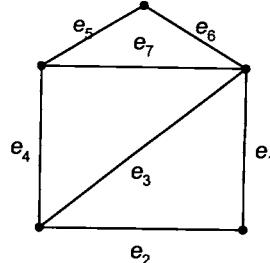
Gambar 8.53 Pembentukan graf dual G^* dari graf G

Berapakah banyak sisi, simpul, dan muka (wilayah) dari graf G^* ? Jika G adalah graf planar dalam representasi bidang dengan n buah simpul, e buah sisi dan f buah muka, maka graf G^* memiliki $n^* = f$ buah simpul, $e^* = e$ buah sisi dan $f^* = n$ buah muka.

Sebuah graf mempunyai dual hanya jika graf tersebut planar. Pertanyaanya, apakah dual dari sebuah graf adalah unik? Dengan kata lain, apakah dual-dual dari sebuah graf planar isomorfik? Jawabannya adalah bahwa sebuah graf planar G mempunyai dual yang unik hanya jika representasi bidangnya unik. Sebagai contoh, pada Gambar 8.54, kedua graf adalah sama (isomorfik), tetapi mempunyai representasi bidang yang berbeda. Akibatnya, dual dari kedua graf yang isomorfik tersebut tidak isomorfik (ditunjukkan pada Gambar 8.55).

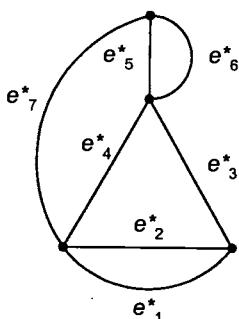


(a)

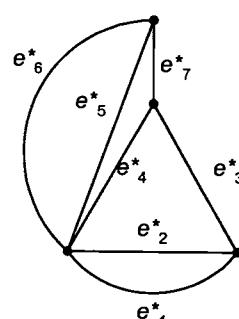


(b)

Gambar 8.54 Dua buah representasi bidang yang berbeda dari graf yang sama



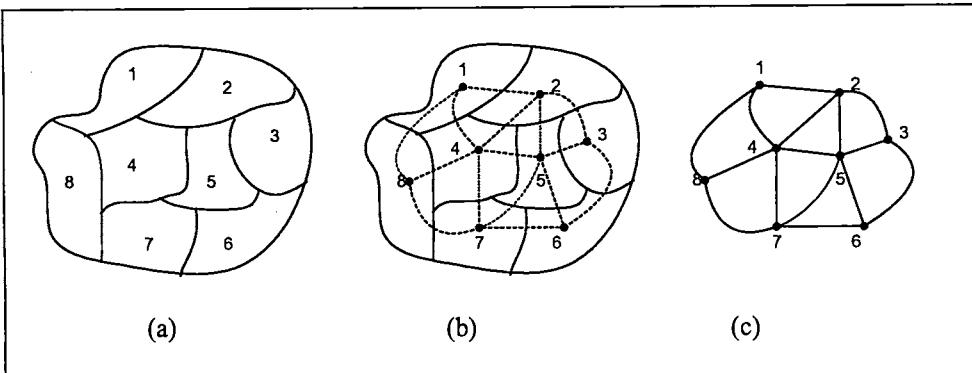
(a) Dual dari Gambar 8.54(a)



(b) Dual dari Gambar 8.54(b)

Gambar 8.55 Dual dari graf di Gambar 8.54

Salah satu aplikasi graf dual yang penting adalah untuk merepresentasikan peta (*map*). Setiap peta pada bidang datar terdiri dari sejumlah wilayah (*region*). Wilayah pada peta dapat menyatakan suatu negara, provinsi, atau kabupaten. Tiap wilayah pada peta dinyatakan sebagai sebuah simpul, sedangkan sisi menyatakan bahwa dua wilayah berbatasan langsung (bertetangga). Gambar 8.56 adalah contoh sebuah peta dan graf yang merepresentasikannya. Sedikit perbedaan dengan graf dual yang telah disebutkan sebelum ini, pada graf yang merepresentasikan peta bidang luar tidak dinyatakan sebagai sebuah simpul. Kita akan membahas kembali penggunaan graf yang merepresentasikan peta pada pewarnaan graf.



Gambar 8.56 (a) Peta, (b) Peta dan graf yang merepresentasikannya,
(c) Graf yang merepresentasikan peta

8.11 Lintasan dan Sirkuit Euler

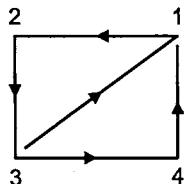
Pada bagian awal bab ini sudah diceritakan sejarah teori graf yang dimulai dengan masalah jembatan Königsberg. Seluruh jembatan hanya boleh dilalui sekali dan kembali lagi ke titik awal keberangkatan. Perjalanan melewati jembatan itu membentuk lintasan tertutup yang diberi nama sirkuit Euler. Jika perjalanan melewati ketujuh jembatan itu tidak harus kembali ke titik awal keberangkatan, maka lintasannya membentuk lintasan terbuka yang diberi nama lintasan Euler. Di dalam upabab ini akan kita pelajari apa yang menjadi syarat keberadaan lintasan atau sirkuit Euler.

DEFINISI 8.18. Lintasan Euler ialah lintasan yang melalui masing-masing sisi di dalam graf tepat satu kali. Bila lintasan tersebut kembali ke simpul asal, membentuk lintasan tertutup (sirkuit), maka lintasan tertutup itu dinamakan sirkuit Euler. Jadi, sirkuit Euler ialah sirkuit yang melewati masing-masing sisi tepat satu kali..

Graf yang mempunyai sirkuit Euler disebut **graf Euler** (*Eulerian graph*). Graf yang mempunyai lintasan Euler dinamakan juga **graf semi-Euler** (*semi-Eulerian graph*).

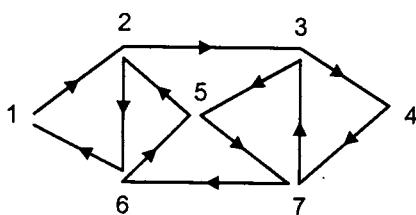
Contoh 8.40

Lintasan Euler pada graf Gambar 8.57(a) : 3, 1, 2, 3, 4, 1.
Gambar lintasannya (dimulai dari 3):

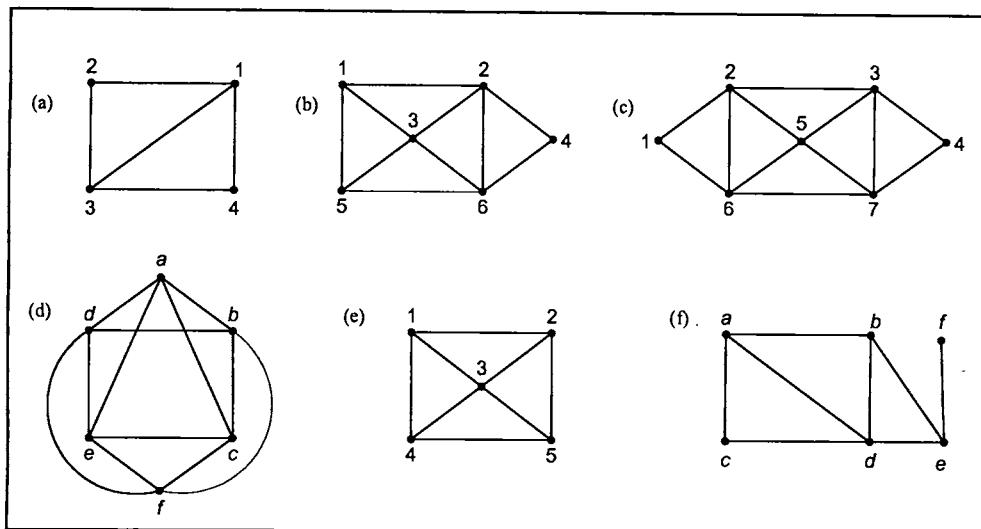


Lintasan Euler pada graf Gambar 8.57(b) : 1, 2, 4, 6, 2, 3, 6, 5, 1, 3 (periksa!)

Sirkuit Euler pada graf Gambar 8.57(c) : 1, 2, 3, 4, 7, 3, 5, 7, 6, 5, 2, 6, 1. Gambar sirkuitnya adalah (dimulai dari 1):



Sirkuit Euler pada graf Gambar 8.57(d) : a, c, f, e, c, b, d, e, a, d, f, b, a (periksa!). Graf (e) dan (f) tidak mempunyai lintasan maupun sirkuit Euler



Gambar 8.57 (a) dan (b) Graf yang mempunyai lintasan Euler (graf semi-Euler)
 (c) dan (d) Graf yang mempunyai sirkuit Euler (graf Euler)
 (e) dan (f) Graf yang tidak memiliki lintasan dan sirkuit Euler

Syarat cukup dan perlu mengenai keberadaan lintasan Euler maupun sirkuit Euler di dalam suatu graf ternyata sangat sederhana. Euler menemukan syarat tersebut ketika memecahkan masalah jembatan Königsberg, yang dinyatakan dengan biimpliksi di dalam Teorema 8.3.

TEOREMA 8.3. Graf terhubung tak-berarah G adalah graf Euler (memiliki sirkuit Euler) jika dan hanya jika setiap simpul di dalam graf tersebut berderajat genap.

Bukti untuk Teorema 8.3 sangat mudah ditunjukkan. Ketika mulai berangkat dari simpul awal, a , ke simpul selanjutnya, b , sisi (a, b) menyumbangkan 1 untuk derajat simpul a . Jika sirkuit Euler dilalui, maka pada setiap simpul terdapat dua buah sisi yang bersisian, yang berarti setiap simpul berderajat 2 (genap). Ketika

kembali ke simpul awal, a , sisi dari simpul terakhir ke simpul a menyumbangkan 1 lagi untuk derajat simpul a , sehingga derajat simpul asal adalah 2 (genap). Kita menyimpulkan bahwa graf terhubung tak-berarah memiliki sirkuit Euler jika setiap simpul di dalam graf berderajat genap. Dengan menggunakan teorema yang terakhir ini jelaslah masalah jembatan Königsberg tidak memiliki sirkuit Euler karena semua simpul berderajat ganjil.

Jika kita ingin membuat graf yang mempunyai sirkuit Euler, maka harus dipenuhi kondisi berikut: (1) graf tersebut harus terhubung, dan (2) semua simpul pada graf berderajat genap.

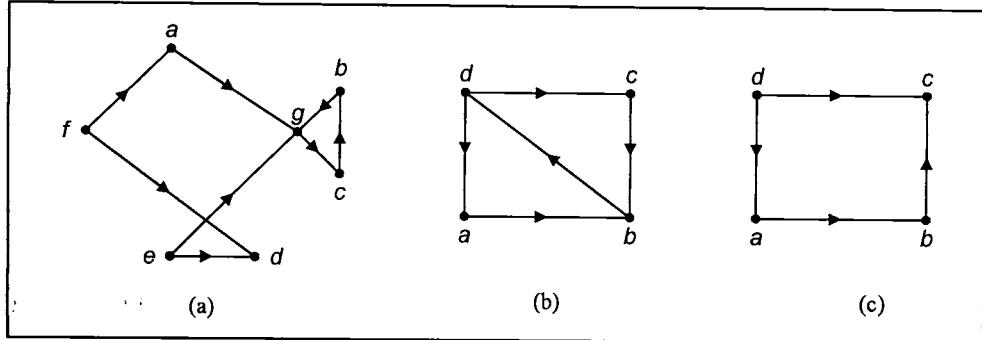
Jika lintasan yang dilalui tidak membentuk sirkuit, maka lintasan yang terbentuk adalah lintasan terbuka yang disebut lintasan Euler. Di sini simpul awal tidak sama dengan simpul terakhir. Baik simpul awal maupun simpul akhir di dalam lintasan Euler berderajat 1 (ganjil). Karena itu, agar sebuah graf mempunyai lintasan Euler (tapi bukan sirkuit Euler), maka graf tersebut harus memiliki tepat dua buah simpul berderajat ganjil. Persyaratan ini dinyatakan dalam Teorema 8.4 berikut.

TEOREMA 8.4. Graf terhubung tak-berarah G adalah graf semi-Euler (memiliki lintasan Euler) jika dan hanya jika di dalam graf tersebut terdapat tepat dua simpul berderajat ganjil.

Catatlah bahwa graf yang memiliki sirkuit Euler pasti mempunyai lintasan Euler, tetapi tidak sebaliknya. Jika kita ingin membuat graf yang mempunyai lintasan Euler (tanpa membentuk sirkuit), maka harus dipenuhi kondisi berikut: (1) graf tersebut harus terhubung, dan (2) graf memiliki tepat dua buah simpul berderajat ganjil.

Lintasan dan sirkuit Euler juga terdapat pada graf berarah. Teorema yang menyatakan syarat keberadaan lintasan dan sirkuit Euler pada graf berarah dinyatakan dengan Teorema 8.5, sedangkan contoh graf berarah yang mengandung lintasan dan sirkuit Euler ditunjukkan pada Gambar 8.58.

TEOREMA 8.5. Graf terhubung berarah G memiliki sirkuit Euler jika dan hanya jika G terhubung dan setiap simpul memiliki derajat-masuk dan derajat-keluar sama. G memiliki lintasan Euler jika dan hanya jika G terhubung dan setiap simpul memiliki derajat-masuk dan derajat-keluar sama kecuali dua simpul, yang pertama memiliki derajat-keluar satu lebih besar derajat-masuk, dan yang kedua memiliki derajat-masuk satu lebih besar dari derajat-keluar.

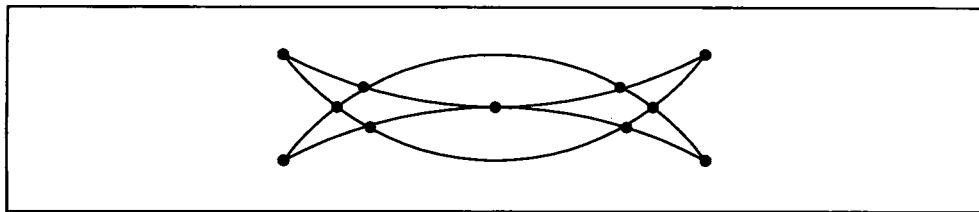


Gambar 8.58 (a) Graf berarah yang mempunyai sirkuit Euler ($a, g, c, b, g, e, d, f, a$)

(b) Graf berarah yang mempunyai lintasan Euler (d, a, b, d, c, b)

(c) Graf berarah yang tidak memiliki lintasan dan sirkuit Euler

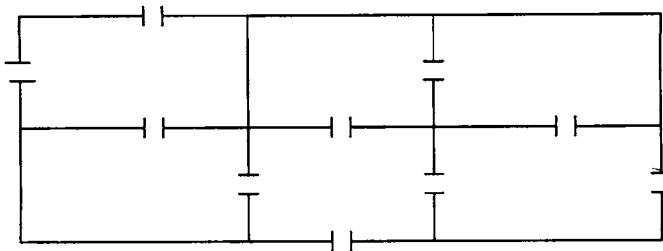
Ada satu permainan sederhana yang sering dimainkan dikala senggang. Permainan itu adalah menggambarkan sesuatu dengan gerakan pensil secara menerus tanpa mengangkat pensil dan tidak menggambar ulang garis-garisnya. Sebagai contoh, dapatkah graf bulan sabit (di dalam literatur graf dikenal dengan nama *Mohammed's scimitars*) pada Gambar 8.59 dilukis tanpa mengangkat pensil dan tanpa menggambar ulang garis-garisnya? Karena kita tidak boleh melukis garis lebih dari sekali, maka kita harus memeriksa apakah grafnya mengandung lintasan atau sirkuit Euler. Ternyata setiap simpul di dalam graf itu berderajat genap yang berarti terdapat sirkuit Euler di dalamnya. Dengan demikian kita dapat melukis graf tanpa mengangkat pensil dan tidak menggambar ulang garis-garisnya.



Gambar 8.59 Bulan sabit Muhammad [ROS99]

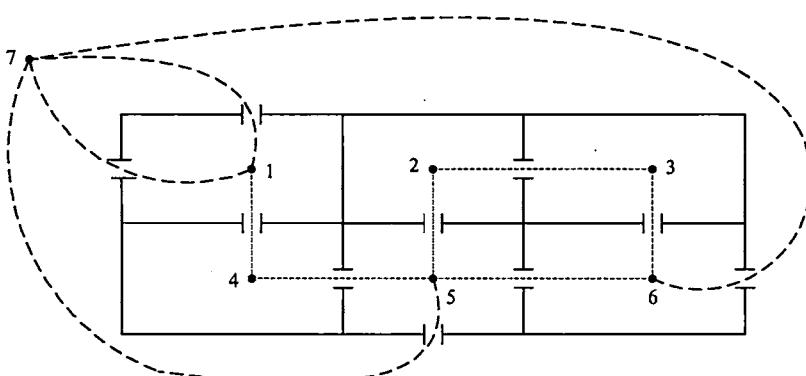
Contoh 8.41

Gambar di bawah ini adalah denah lantai dasar sebuah gedung. Apakah dimungkinkan berjalan melalui setiap pintu di lantai itu hanya satu kali saja jika kita boleh mulai memasuki pintu yang mana saja?



Penyelesaian:

Nyatakan setiap ruangan (termasuk ruang luar) sebagai simpul dan pintu-pintu yang menghubungkan antar ruangan sebagai sisi. Graf yang dihasilkan ditunjukkan dengan garis putus-putus di bawah ini. Setiap pintu hanya boleh dilalui sekali dipandang sebagai melalui setiap sisi di dalam graf hanya sekali. Karena kita tidak diharuskan harus kembali ke titik asal keberangkatan, maka persoalan ini sebenarnya adalah menentukan apakah di dalam graf tersebut terdapat lintasan Euler. Agar graf mempunyai lintasan Euler, maka harus terdapat dua simpul berderajat ganjil, simpul lainnya berderajat genap. Dari representasi graf yang diperoleh, terdapat dua simpul berderajat ganjil, yaitu simpul 1 dan 6, sedangkan simpul lainnya berderajat genap. Menurut Teorema 8.4, pasti terdapat lintasan Euler di dalam graf tersebut. Kesimpulannya, kita dapat melalui setiap pintu di lantai itu tepat sekali. ■

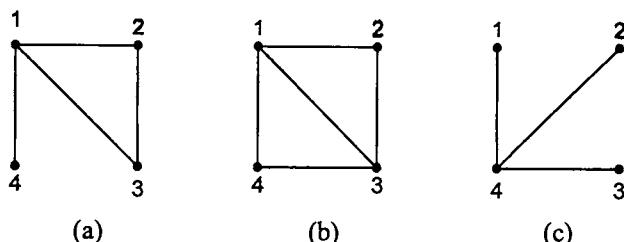


8.12 Lintasan dan Sirkuit Hamilton

Jika lintasan dan sirkuit Euler melalui sisi-sisi graf tepat sekali, maka lintasan dan sirkuit Hamilton melalui simpul-simpul graf tepat sekali.

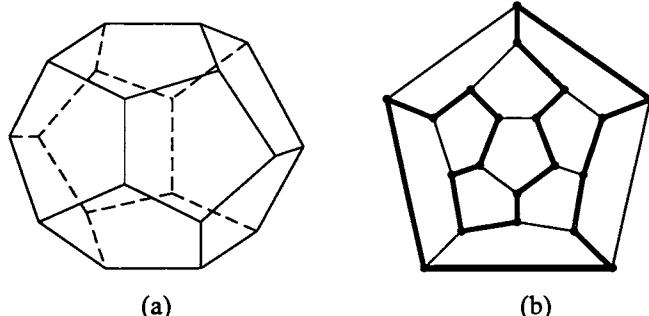
DEFINISI 8.19. **Lintasan Hamilton** ialah lintasan yang melalui tiap simpul di dalam graf tepat satu kali. Bila lintasan itu kembali ke simpul asal membentuk lintasan tertutup (sirkuit), maka lintasan tertutup itu dinamakan **sirkuit Hamilton**. Dengan kata lain, sirkuit Hamilton ialah sirkuit yang melalui tiap simpul di dalam graf tepat satu kali, kecuali simpul asal (sekaligus simpul akhir) yang dilalui dua kali.

Graf yang memiliki sirkuit Hamilton dinamakan **graf Hamilton**, sedangkan graf yang hanya memiliki lintasan Hamilton disebut **graf semi-Hamilton**. Gambar 8.60 memperlihat contoh graf yang mengandung lintasan atau sirkuit Euler.



Gambar 8.60 (a) graf yang memiliki lintasan Hamilton (misal: 3, 2, 1, 4)
 (b) graf yang memiliki sirkuit Hamilton (1, 2, 3, 4, 1)
 (c) graf yang tidak memiliki lintasan maupun sirkuit Hamilton

Nama sirkuit Hamilton muncul ketika Sir William Hamilton membuat permainan *dodecahedron*. Pada tahun 1859 Sir William Hamilton menawarkan mainan teka-teki ke pabrik alat mainan Dublin. Mainan itu terdiri dari *dodecahedron* (yaitu benda yang disusun oleh 12 buah pentagonal dan di sini ada 20 buah titik sudut) dan tiap titik sudut diberi nama ibukota negara (Gambar 8.61(a)). Permainan yang dapat dilakukan adalah membentuk perjalanan keliling dunia, yang mengunjungi setiap ibukota tepat satu kali dan kembali lagi ke kota asal. Persoalan ini dinamakan mencari sirkuit Hamilton. Gambar 8.61(b) adalah graf yang memodelkan *dodecahedron* dengan sebuah sirkuit Hamilton (garis tebal)



Gambar 8.61 (a) *Dodecahedron* Hamilton, dan (b) graf yang mengandung sirkuit Hamilton

Walaupun masalah penentuan lintasan atau sirkuit Hamilton mempunyai kemiripan dengan masalah lintasan/sirkuit Euler, namun sayangnya belum ditemukan syarat cukup dan syarat perlu yang sederhana. Untuk menunjukkan bahwa suatu graf tertentu mempunyai lintasan atau sirkuit Hamilton, kita terpaksa menggunakan cara pembuatan eksplisit lintasan atau sirkuit yang demikian [LIU85].

Di sini diberikan beberapa hasil umum tentang keberadaan lintasan atau sirkuit Hamilton dengan menggunakan beberapa syarat cukup (bukan syarat perlu) berupa Teorema Dirac dan Teorema Ore.

TEOREMA 8.6. (Teorema Dirac) Jika G adalah graf sederhana dengan n buah simpul ($n \geq 3$) sedemikian sehingga derajat tiap simpul paling sedikit $n/2$ (yaitu, $d(v) \geq n/2$ untuk setiap simpul v di G), maka G adalah graf Hamilton.

TEOREMA 8.7. (Teorema Ore) Jika G adalah graf sederhana dengan n buah simpul ($n \geq 3$) sedemikian sehingga $d(v) + d(u) \geq n$ untuk setiap pasang simpul tidak-bertetangga u dan v , maka G adalah graf Hamilton.

TEOREMA 8.8. Setiap graf lengkap adalah graf Hamilton.

TEOREMA 8.9. Di dalam graf lengkap G dengan n buah simpul ($n \geq 3$) terdapat sebanyak $(n - 1)/2$ buah sirkuit Hamilton.

TEOREMA 8.10. Di dalam graf lengkap G dengan n buah simpul ($n \geq 3$ dan n ganjil), terdapat $(n - 1)/2$ buah sirkuit Hamilton yang saling lepas (tidak ada sisi yang beririsan). Jika n genap dan $n \geq 4$, maka di dalam G terdapat $(n - 2)/2$ buah sirkuit Hamilton yang saling lepas.

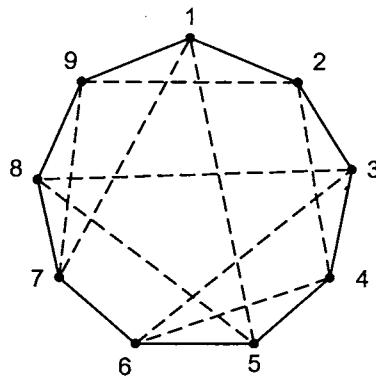
Contoh 8.42

(Persoalan pengaturan tempat duduk). Sembilan anggota sebuah klub bertemu tiap hari untuk makan siang pada sebuah meja bundar. Mereka memutuskan duduk sedemikian sehingga setiap anggota mempunyai tetangga duduk berbeda pada setiap makan siang. Berapa hari pengaturan tersebut dapat dilaksanakan?

Penyelesaian:

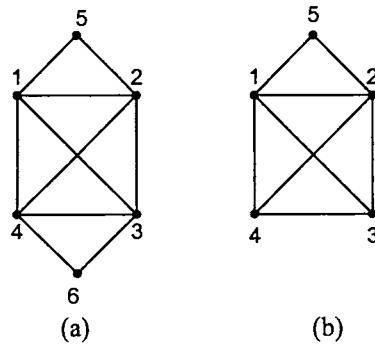
Persoalan di atas dapat direpresentasikan oleh sebuah graf dengan sembilan buah simpul sedemikian sehingga setiap simpul menyatakan anggota klub, dan sisi yang menghubungkan dua buah simpul menyatakan kedua simpul tersebut bertetangga tempat duduk (Gambar 8.62).

Contoh pengaturan tempat duduk adalah 1, 2, 3, 4, 5, 6, 7, 8, 9, 1 (garis tebal), dan 1, 3, 5, 2, 7, 4, 9, 6, 8, 1 (garis putus-putus). Ini adalah dua buah sirkuit Hamilton yang saling lepas. Jumlah pengaturan tempat duduk yang berbeda adalah $(9 - 1)/2 = 4$. Jadi, pengaturan tempat duduk yang berbeda itu dapat diterapkan selama 4 hari, yang setiap harinya setiap peserta mempunyai tetangga duduk yang berbeda dengan hari sebelumnya.



Gambar 8.62 Graf yang merepresentasikan persoalan pengaturan tempat duduk.

Beberapa graf dapat mengandung (i) sirkuit Euler dan sirkuit Hamilton sekaligus, (ii) mengandung sirkuit Euler tetapi tidak mengandung sirkuit Hamilton, (iii) mengandung sirkuit Euler dan lintasan Hamilton, (iv) mengandung lintasan Euler maupun lintasan Hamilton, (v) tidak mengandung lintasan Euler namun mengandung sirkuit Hamilton, dan sebagainya. Graf pada Gambar 8.63(a) mengandung sirkuit Hamilton maupun sirkuit Euler, sedangkan graf pada Gambar 8.63(b) mengandung sirkuit Hamilton dan lintasan Euler (periksa!).



Gambar 8.63 (a) Graf Hamilton sekaligus graf Euler
 (b) Graf Hamilton sekaligus graf semi-Euler

Terdapat banyak aplikasi yang berkaitan dengan graf. Di dalam aplikasi itu, graf digunakan sebagai alat untuk merepresentasikan atau memodelkan persoalan. Berdasarkan graf yang dibentuk, barulah persoalan tersebut diselesaikan. Di

bawah ini diberikan beberapa aplikasi yang berkaitan dengan lintasan/sirkuit di dalam graf, yaitu menentukan **lintasan terpendek** (*shortest path*), persoalan **pedagang keliling** (*travelling salesperson problem*), persoalan **tukang pos Cina**, dan **pewarnaan graf** (*graph colouring*).

8.13 Lintasan Terpendek (*Shortest Path*)

Persoalan mencari lintasan terpendek di dalam graf merupakan salah satu persoalan optimasi. Graf yang digunakan dalam pencarian lintasan terpendek adalah graf berbobot (*weighted graph*), yaitu graf yang setiap sisinya diberikan suatu nilai atau bobot. Bobot pada sisi graf dapat menyatakan jarak antar kota, waktu pengiriman pesan, ongkos pembangunan, dan sebagainya. Asumsi yang kita gunakan di sini adalah bahwa semua bobot bernilai positif. Kata “terpendek” jangan selalu diartikan secara fisik sebagai panjang minimum, sebab kata “terpendek” berbeda-beda maknanya bergantung pada tipikal persoalan yang akan diselesaikan. Namun, secara umum “terpendek” berarti meminimisasi bobot pada suatu lintasan di dalam graf.

Contoh-contoh terapan pencarian lintasan terpendek misalnya:

1. Misalkan simpul pada graf dapat merupakan kota, sedangkan sisi menyatakan jalan yang menghubungkan dua buah kota. Bobot sisi graf dapat menyatakan jarak antara dua buah kota atau rata-rata waktu tempuh antara dua buah kota. Apabila terdapat lebih dari satu lintasan dari kota *A* ke kota *B*, maka persoalan lintasan terpendek di sini adalah menentukan jarak terpendek atau waktu tersingkat dari kota *A* ke kota *B*.
2. Misalkan simpul pada graf dapat merupakan terminal komputer atau simpul komunikasi dalam suatu jaringan, sedangkan sisi menyatakan saluran komunikasi yang menghubungkan dua buah terminal. Bobot pada graf dapat menyatakan biaya pemakaian saluran komunikasi antara dua buah terminal, jarak antara dua buah terminal, atau waktu pengiriman pesan (*message*) antara dua buah terminal. Persoalan lintasan terpendek di sini adalah menentukan jalur komunikasi terpendek antara dua buah terminal komputer. Lintasan terpendek akan menghemat waktu pengiriman pesan dan biaya komunikasi.

Ada beberapa macam persoalan lintasan terpendek, antara lain:

- a. Lintasan terpendek antara dua buah simpul tertentu.
- b. Lintasan terpendek antara semua pasangan simpul.
- c. Lintasan terpendek dari simpul tertentu ke semua simpul yang lain.
- d. Lintasan terpendek antara dua buah simpul yang melalui beberapa simpul tertentu.

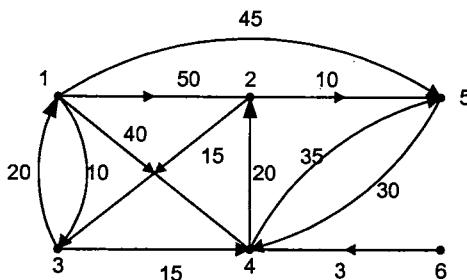
Pada dasarnya, jenis persoalan a mirip dengan jenis persoalan c, karena pencarian lintasan terpendek pada jenis persoalan c dapat dihentikan bila simpul tujuan

yang dikehendaki sudah ditemukan lintasan terpendeknya. Di dalam buku ini kita hanya akan membahas jenis persoalan c.

Deskripsi persoalan lintasan terpendek pada jenis persoalan c adalah sebagai berikut: diberikan graf berbobot $G = (V, E)$ dan sebuah simpul awal a . Tentukan lintasan terpendek dari a ke setiap simpul lainnya di dalam G .

Sebagai ilustrasi, tinjau graf berarah pada Gambar 8.64. Lintasan terpendek dari simpul 1 ke semua simpul lain diberikan pada tabel di bawah ini (diurut dari lintasan terpendek pertama, kedua, ketiga dst.)

Simpul asal	Simpul tujuan	Lintasan terpendek	Jarak
1	3	1, 3	10
1	4	1, 3, 4	25
1	2	1, 3, 4, 2	45
1	5	1, 5	45
1	6	tidak ada	-



Gambar 8.64 Graf yang digunakan sebagai contoh untuk persoalan lintasan terpendek

Perhatikan dari tabel di atas, bahwa lintasan terpendek dari 1 ke 2 berarti juga melalui lintasan terpendek dari 1 ke 3 dan dari 1 ke 4.

Sampai saat ini, sudah banyak algoritma mencari lintasan terpendek yang pernah ditulis orang. Algoritma lintasan terpendek yang paling terkenal adalah **algoritma Dijkstra** (sesuai dengan nama penemunya, Edsger W. Dijkstra). Dalam naskah aslinya, algoritma Dijkstra diterapkan pada untuk mencari lintasan terpendek pada graf berarah. Namun, algoritma ini juga benar untuk graf tak-berarah.

Algoritma Dijkstra mencari lintasan terpendek dalam sejumlah langkah. Algoritma ini menggunakan prinsip *greedy*. Prinsip *greedy* pada algoritma Dijkstra menyatakan

bahwa pada setiap langkah kita memilih sisi yang berbobot minimum dan memasukkannya ke dalam himpunan solusi.

Ada beberapa versi algoritma Dijkstra yang ditulis pada berbagai pustaka. Algoritma yang dibahas di bawah ini adalah salah satu versinya.

Misalkan sebuah graf berbobot dengan n buah simpul dinyatakan dengan matriks ketetanggan $M = [m_{ij}]$, yang dalam hal ini,

$$m_{ij} = \text{bobot sisi } (i, j) \quad (\text{pada graf tak-berarah } m_{ij} = m_{ji})$$

$$m_{ii} = 0$$

$$m_{ij} = \infty, \text{jika tidak ada sisi dari simpul } i \text{ ke simpul } j$$

Selain matriks M , kita juga menggunakan tabel $S = [s_i]$ yang dalam hal ini,

$$s_i = 1, \text{jika simpul } i \text{ termasuk ke dalam lintasan terpendek}$$

$$s_i = 0, \text{jika simpul } i \text{ tidak termasuk ke dalam lintasan terpendek}$$

dan tabel $D = [d_i]$ yang dalam hal ini,

$$d_i = \text{panjang lintasan dari simpul awal } a \text{ ke simpul } i$$

Algoritma Dijkstra dinyatakan dalam notasi *pseudo-code* sebagai berikut:

```
procedure Dijkstra(input m: matriks, a : simpul awal)
  (Mencari lintasan terpendek dari simpul awal a ke semua simpul lainnya
   Masukan: matriks ketetanggan (m) dari graf berbobot G dan simpul awal a
   Keluaran: lintasan terpendek dari a ke semua simpul lainnya)
}

Deklarasi
   $s_1, s_2, \dots, s_n : \underline{\text{integer}}$  { tabel integer }
   $d_1, d_2, \dots, d_n : \underline{\text{integer}}$  { tabel integer }
   $i, j, k : \underline{\text{integer}}$ 

Algoritma
  { Langkah 0 (inisialisasi: )
  for  $i \leftarrow 1$  to  $n$  do
     $s_i \leftarrow 0$ 
     $d_i \leftarrow m_{ai}$ 
  endfor

  { Langkah 1: )
   $s_a \leftarrow 1$  (karena simpul a adalah simpul asal lintasan terpendek, jadi
   simpul a sudah pasti terpilih dalam lintasan terpendek)
   $d_a \leftarrow \infty$  (tidak ada lintasan terpendek dari simpul a ke a)

  { Langkah 2, 3, ..., n - 1: )
  for  $k \leftarrow 2$  to  $n - 1$  do
     $j \leftarrow \text{simpul dengan } s_j = 0 \text{ dan } d_j \text{ minimal}$ 
     $s_j \leftarrow 1$  (simpul j sudah terpilih ke dalam lintasan terpendek)
```

```

{ perbarui tabel d }
for semua simpul i dengan  $s_i = 0$  do
    if  $d_j + m_{ji} < d_i$  then
         $d_i \leftarrow d_j + m_{ji}$ 
    endif
endfor
endfor

```

Algoritma 8.1 Algoritma Dijkstra untuk mencari lintasan terpendek

Tinjau kembali graf berarah pada Gambar 8.66, dengan matriks ketetanggaan M sebagai berikut:

		$j = 1$	2	3	4	5	6
$i = 1$		0	50	10	40	45	∞
2		∞	0	15	∞	10	∞
3		20	∞	0	15	∞	∞
4		∞	20	∞	0	35	∞
5		∞	∞	∞	30	0	∞
6		∞	∞	∞	3	∞	0

Perhitungan lintasan terpendek dari simpul awal $a = 1$ ke semua simpul lainnya ditabulasikan sebagai berikut.

Lelaran	Simpul yang dipilih	Lintasan	S						D					
			1	2	3	4	5	6	1	2	3	4	5	6
Inisial	-	-	0	0	0	0	0	0	0	50	10	40	45	∞
										(1,2)	(1,3)	(1,4)	(1,5)	(1,6)
1	1	1	1	0	0	0	0	0	∞	50	10	40	45	∞
										(1,2)	(1,3)	(1,4)	(1,5)	(1,6)
2	3	1, 3	1	0	1	0	0	0	∞	50	10	25	45	∞
										(1,2)	(1,3)	(1,3,4)	(1,5)	(1,6)
3	4	1, 3, 4	1	0	1	1	0	0	∞	45	10	25	45	∞
										(1,3,4,2)	(1,3)	(1,3,4)	(1,5)	(1,6)
4	2	1, 3, 4, 2	1	1	1	1	0	0	∞	45	10	25	45	∞
										(1,3,4,2)	(1,3)	(1,3,4)	(1,5)	(1,6)
5	5	1, 5	1	1	1	1	1	0	∞	45	10	25	45	∞
										(1,3,4,2)	(1,3)	(1,3,4)	(1,5)	(1,6)

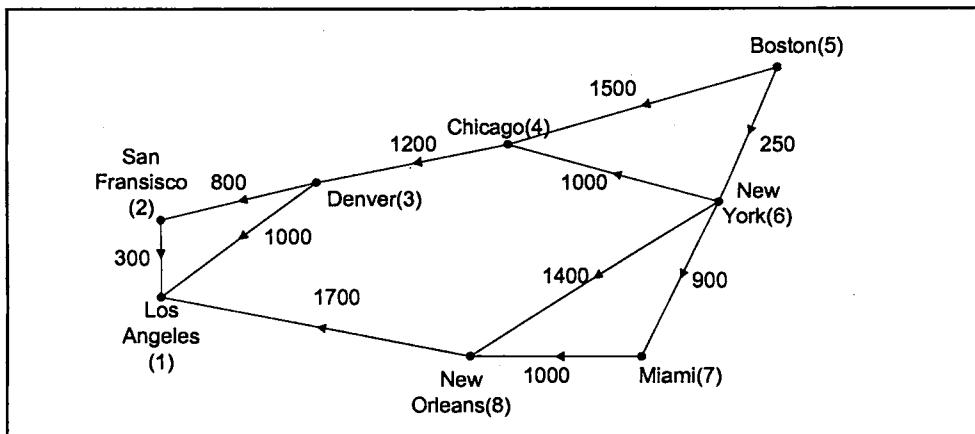
(Keterangan: Angka-angka di dalam tanda kurung menyatakan lintasan terpendek dari 1 ke simpul tersebut)

Jadi, lintasan terpendek dari:

- 1 ke 3 adalah 1, 3 dengan panjang = 10
- 1 ke 4 adalah 1, 3, 4 dengan jarak = 25
- 1 ke 2 adalah 1, 3, 4, 2 dengan jarak = 45
- 1 ke 5 adalah 1, 5 dengan jarak = 45
- 1 ke 6 tidak ada lintasan

Contoh 8.43

Tinjau graf berarah pada Gambar 8.65 yang menyatakan jarak beberapa kota di Amerika Serikat.



Gambar 8.65 Graf berarah dari sebuah peta AS

Penyelesaian:

Matriks ketetanggaan M sebagai berikut:

	$j = 1$	2	3	4	5	6	7	8
$i = 1$	0	∞						
2	300	∞						
3	1000	800	0	∞	∞	∞	∞	∞
4	∞	∞	1200	0	∞	∞	∞	∞
5	∞	∞	∞	1500	0	250	∞	∞
6	∞	∞	∞	1000	∞	0	900	1400
7	∞	∞	∞	∞	∞	∞	0	1000
8	1700	∞	∞	∞	∞	∞	∞	0

Perhitungan lintasan terpendek dari simpul awal $a = 5$ ke semua simpul lainnya ditabulasikan sebagai berikut.

Lelaran	Simpul yang dipilih	Lintasan	<i>S</i>								<i>D</i>							
			1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8
Inisial	-	-	0	0	0	0	0	0	0	0	∞	∞	∞	1500	0	250	∞	∞
1	5	5	0	0	0	1	0	0	∞	∞	1500	∞	∞	250	∞	∞		
2	6	5, 6	0	0	0	1	1	0	∞	∞	1250	∞	∞	250	1150	1650		
3	7	5, 6, 7	0	0	0	1	1	1	∞	∞	1250	∞	∞	250	1150	1650		
4	4	5, 6, 4	0	0	0	1	1	1	0	∞	∞	2450	1250	∞	250	1150	1650	
5	8	5, 6, 8	0	0	0	1	1	1	1	3350	∞	2450	1250	∞	250	1150	1650	
6	3	5, 6, 4, 3	0	0	1	1	1	1	1	3350	∞	2450	1250	∞	250	1150	1650	
7	2	5, 6, 4, 3, 2	0	1	1	1	1	1	1	3350	3250	2450	1250	∞	250	1150	1650	

Jadi, lintasan terpendek dari:

- 5 ke 6 adalah 5, 6 dengan panjang = 250
- 5 ke 7 adalah 5, 6, 7 dengan jarak = 1150
- 5 ke 4 adalah 5, 6, 4 dengan jarak = 1250
- 5 ke 8 adalah 5, 6, 8 dengan jarak = 1650
- 5 ke 3 adalah 5, 6, 4, 3 dengan jarak = 2450
- 5 ke 2 adalah 5, 6, 4, 3, 2 dengan jarak = 3250
- 5 ke 1 adalah 5, 6, 8, 1 dengan jarak = 3350

Perhatikan bahwa jumlah lelaran hanya 7 (atau $n - 1$). Jarak dari simpul 5 ke 1 (Los Angeles) sudah benar karena satu-satunya nilai s_i yang nol adalah s_1 , sehingga nilai di dalam d_1 sudah otomatis menjadi jarak terpendek dari simpul 5 ke 1. ■

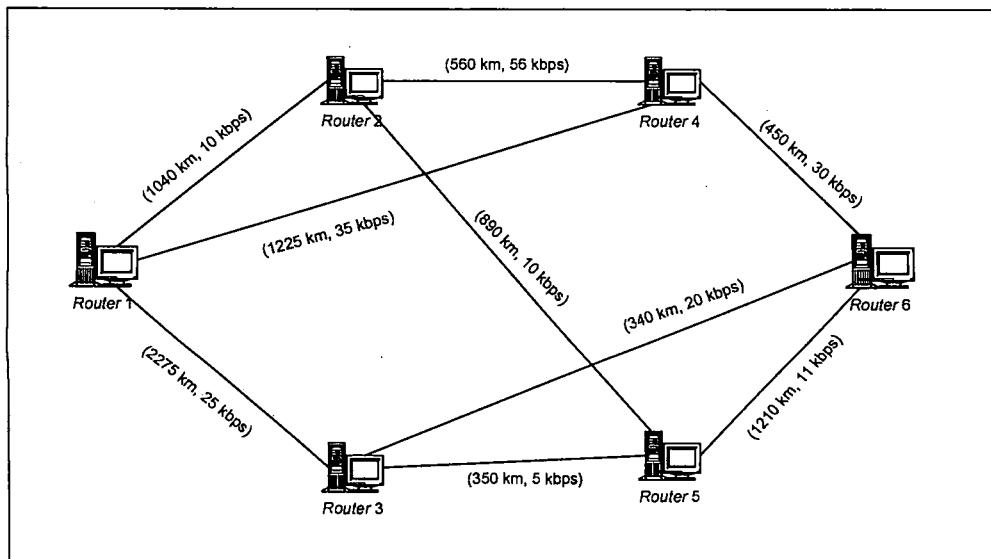
Penerapan Algoritma Dijkstra pada Jaringan Komputer

Jaringan komputer terdiri dari sejumlah komputer yang terhubung satu sama lain melalui saluran komunikasi (misalnya kabel, serat optik, gelombang mikro, gelombang radio). Antara satu komputer dengan komputer lain seringkali terpisah jauh secara geografis (antar kota atau antar negara), atau berada dalam wilayah geografis yang sama (dalam satu gedung, dalam satu area, dsb). Pesan yang dikirim dari satu komputer ke komputer lainnya umumnya dipecah menjadi sejumlah paket (*packet*). data yang berukuran lebih kecil. Saluran komunikasi yang digunakan untuk mengantarkan paket mempunyai keterbatasan dalam hal kecepatan transmisi (umumnya kecepatan transmisi dinyatakan dalam satuan *kbps – kilobit per second*). Untuk menyampaikan paket data dari satu komputer ke komputer lainnya, sistem jaringan komputer harus dapat melakukan pemilihan rute yang tepat agar paket dapat sampai ke komputer tujuan dalam waktu yang cepat. Yang dimaksud dengan **perutean** (*routing*) adalah menentukan lintasan yang dilalui oleh paket dari komputer pengirim (asal) ke komputer penerima (tujuan).

Permasalahannya adalah bagaimana menentukan rute yang tepat sehingga paket data dapat sampai ke komputer penerima dalam waktu yang sesingkat mungkin. Dengan menggunakan rute tersebut, paket data yang sampai ke suatu komputer dapat diarahkan ke komputer tetangga yang tepat sehingga paket menuju komputer penerima dengan delay (delay) waktu yang minimum. Dengan kata lain, kita harus menentukan lintasan terpendek yang akan dilalui oleh paket tersebut dari komputer pengirim ke komputer penerima.

Perutean yang diharapkan adalah **perutean adaptif** (*adaptive routing*). Perutean adaptif berarti sistem jaringan komputer dapat menentukan rute baru apabila terjadi perubahan topologi jaringan (misalnya ada penambahan *router* baru, kerusakan pada suatu *router* sehingga *router* tersebut tidak bisa dilalui, atau perubahan kecepatan transmisi antar *router*).

Jaringan komputer dapat dimodelkan sebagai sebuah graf terhubung, dengan setiap simpul menyatakan sebuah komputer (bisa berupa terminal komputer atau sebuah *router*. *Router* adalah komputer yang didedikasikan untuk mengarahkan pesan. Di dalam tugas ini, kita mengasumsikan simpul menyatakan *router*, sedangkan terminal komputer – yang merupakan komputer *end user* – dianggap cabang dari *router*). Sisi di dalam graf menyatakan saluran komunikasi (sering disebut *link*). Setiap sisi di-assign dengan sebuah label nilai (yang disebut bobot atau *weight*). Bobot tersebut dapat menyatakan jarak geografis (dalam kilometer), kecepatan transfer data, atau delay transmisi (waktu pengiriman) (lihat Gambar 8.66). Setiap *router* memelihara sebuah tabel yang disebut tabel rute (*routing table*). Tabel rute berisi *router* asal, *router* tujuan, dan simpul antara (via) yang dilalui (lihat Gambar 8.67).

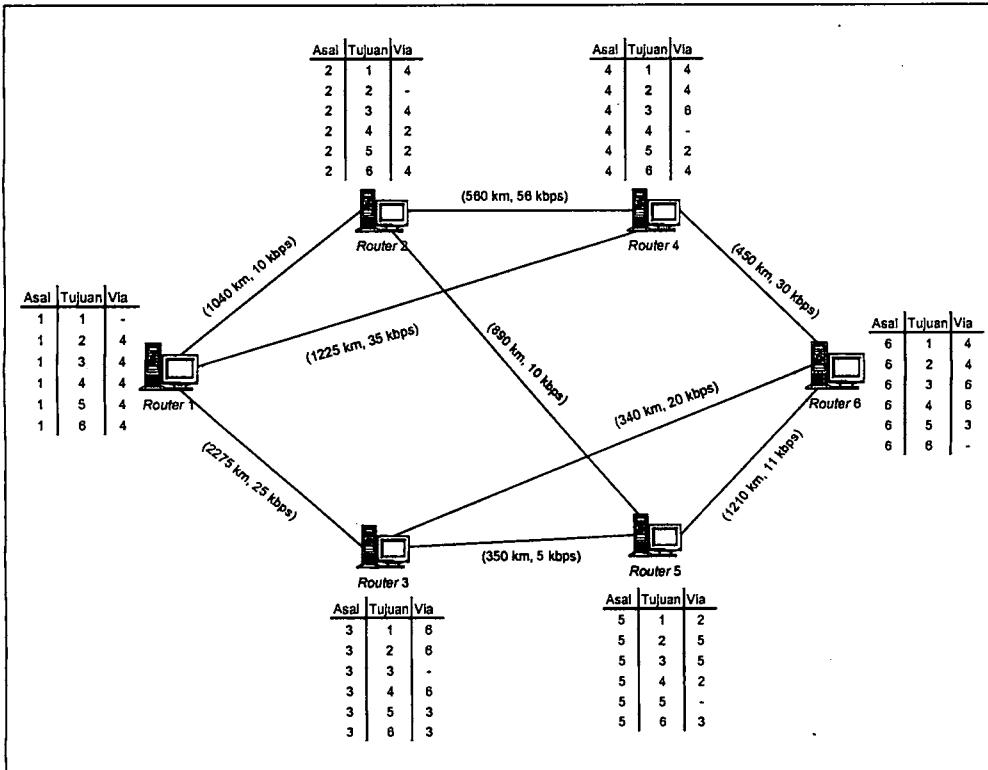


Gambar 8.66 Jaringan komputer (simpul-simpul terminal komputer *end user* tidak digambarkan):

Mencari lintasan terpendek dari *router* asal ke *router* tujuan dapat diartikan sebagai menentukan lintasan terpendek dari simpul asal ke simpul tujuan di dalam graf yang merepresentasikan jaringan komputer tersebut. Algoritma Dijkstra adalah algoritma yang banyak digunakan untuk mencari lintasan terpendek. Lintasan terpendek yang dihasilkan dari algoritma Dijkstra (berdasarkan delai) untuk jaringan komputer pada Gambar 8.66 ditabulasikan di dalam Tabel 8.1. Hasil pembentukan tabel rute (berdasarkan hasil algoritam Dijkstra) diperlihatkan pada Gambar 8.67.

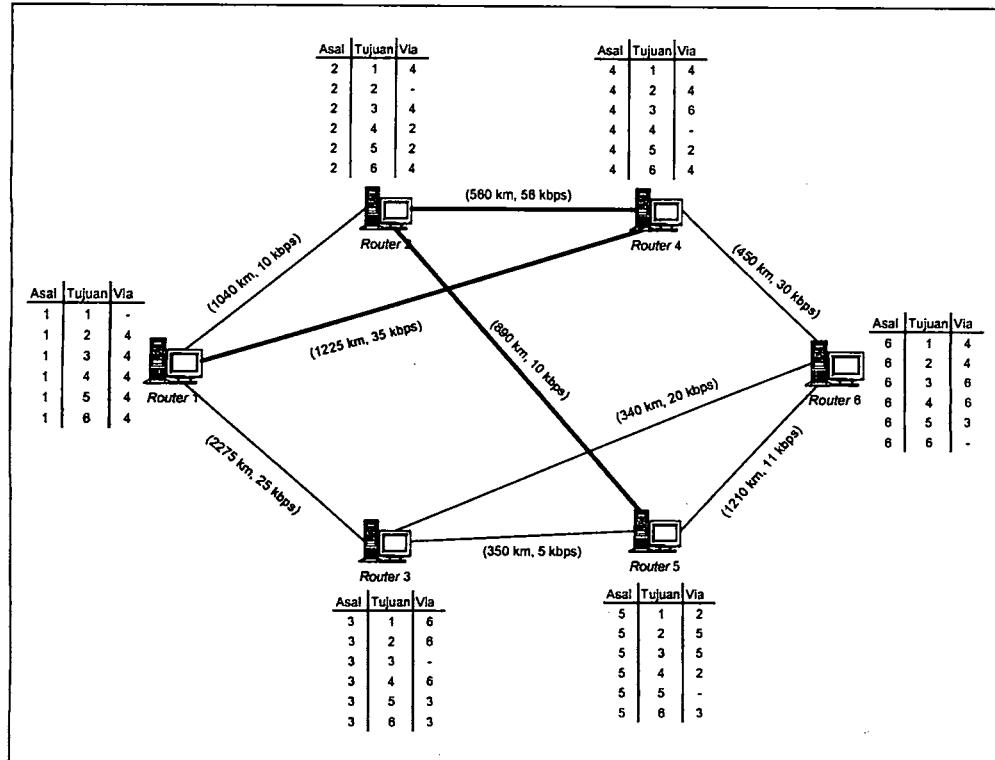
Tabel 8.1 Lintasan terpendek yang dihasilkan dengan Algoritam Dijkstra untuk jaringan komputer pada Gambar 8.66.

<i>Router</i> Asal	<i>Router</i> Tujuan	Lintasan Terpendek
1	1	-
	2	1, 4, 2
	3	1, 4, 6, 3
	4	1, 4
	5	1, 4, 2, 5
	6	1, 4, 6
2	1	2, 4, 1
	2	-
	3	2, 4, 6, 3
	4	2, 4
	5	2, 5
	6	2, 4, 6
3	1	3, 6, 4, 1
	2	3, 6, 4, 2
	3	-
	4	3, 6, 4
	5	3, 5
	6	3, 6
4	1	4, 1
	2	4, 2
	3	4, 6, 2
	4	4, 6, 3
	5	4, 2, 5
	6	4, 6
5	1	5, 2, 4, 1
	2	5, 2
	3	5, 3
	4	5, 2, 4
	5	-
	6	5, 3, 6
6	1	6, 4, 1
	2	6, 4, 2
	3	6, 3
	4	6, 4
	5	6, 3, 5
	6	-



Gambar 8.67 Jaringan komputer dengan tabel rute pada setiap router.

Sebuah pesan selalu mengandung informasi asal (*source*) dan tujuan (*destination*). Misalkan sebuah pesan dikirim dari *router 1* ke *router 5*, maka *router 1* membaca tujuan pesan (yaitu ke *router 5*), lalu berdasarkan tabel rute yang dipeliharanya, *router 1* mengarahkan pesan ke *router 4*. Setelah sampai di *router 4*, *router 4* membaca tujuan pesan (yaitu ke *router 5*), lalu berdasarkan tabel rute yang dipeliharanya, *router 4* mengarahkan pesan ke *router 2*. Begitu seterusnya sehingga pesan sampai ke *router tujuan*. Lintasan terpendek yang dilalui oleh pesan diperlihatkan dengan garis tebal pada Gambar 8.68.



Gambar 8.68 Lintasan terpendek yang dilalui oleh pesan dari router 1 ke router 5.

8.14 Persoalan Pedagang Keliling

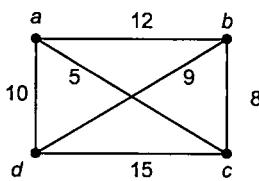
Persoalan Pedagang Keliling (*Travelling Salesperson Problem - TSP*) termasuk ke dalam persoalan yang sangat terkenal di dalam teori graf. Nama persoalan ini dilihami oleh masalah seorang pedagang yang berkeliling mengunjungi sejumlah kota. Deskripsi persoalannya adalah sebagai berikut: diberikan sejumlah kota dan jarak antar kota. Tentukan sirkuit terpendek yang harus dilalui oleh seorang pedagang bila pedagang itu berangkat dari sebuah kota asal dan menyinggahi setiap kota tepat satu kali dan kembali lagi ke kota asal keberangkatan.

Kota dapat dinyatakan sebagai simpul graf, sedangkan sisi menyatakan jalan yang menghubungkan antar dua buah kota. Bobot pada sisi menyatakan jarak antara dua buah kota. Persoalan perjalanan pedagang tidak lain adalah menentukan sirkuit Hamilton yang memiliki bobot minimum pada sebuah graf terhubung.

Meskipun persoalan ini bernama perjalanan pedagang, namun penerapannya tidak hanya pada kasus yang berhubungan dengan pedagang. Banyak terapan TSP yang muncul dalam kehidupan sehari-hari maupun dalam bidang teknik, antara lain:

1. Misalkan sebuah mobil pos ditugaskan mengambil surat dari kotak pos yang tersebar pada n buah lokasi di berbagai sudut kota. Graf dengan $n + 1$ simpul dapat digunakan untuk menyajikan persoalan. Satu simpul menyatakan kantor pos tempat mobil pos mulai berangkat. Sisi (i, j) diberi bobot yang sama dengan jarak dari kotak pos i ke kotak pos j . Rute yang dilalui mobil pos adalah sebuah perjalanan (*tour*) yang mengunjungi setiap kotak pos hanya satu kali dan kembali lagi ke kantor pos asal. Kita harus menentukan rute perjalanan yang mempunyai total jarak terpendek.
2. Misalkan kita ingin menggunakan lengan robot untuk mengencangkan mur pada beberapa buah peralatan mesin dalam sebuah jalur perakitan. Lengan robot mulai berada dari posisi awalnya (yaitu di atas mur pertama, kemudian mengencangkannya), lalu berturut-turut pindah ke mur-mur berikutnya dan kembali lagi ke posisi awalnya. Siklus yang dibentuk jelaslah perjalanan mengunjungi simpul-simpul sebuah graf. Tiap simpul menyatakan mur, sisi menyatakan perpindahan, dan bobot setiap sisi menyatakan waktu yang diperlukan lengan robot untuk berpindah di antara dua simpul. Perjalanan dengan biaya minimum berarti meminimumkan waktu yang dibutuhkan robot untuk menyelesaikan tugasnya (yaitu total waktu perpindahan dari sebuah mur ke mur lainnya, sedangkan waktu pengencangan mur tidak dimasukkan dalam perhitungan).
3. Dalam lingkungan produksi terdapat beberapa komoditi yang dihasilkan oleh sekumpulan mesin. Proses fabrikasi merupakan sebuah siklus. Tiap-tiap siklus produksi menghasilkan n komoditi berbeda. Bila pekerjaan mesin diubah dari produksi komoditas i ke komoditas j , perubahan tersebut mendatangkan biaya sebesar c_{ij} . Diiinginkan untuk menemukan runtunan produksi yang menghasilkan n komoditas ini. Runtunan tersebut harus meminimumkan total biaya akibat perubahan urutan produksi (biaya lainnya tidak bergantung pada runtunan). Karena komoditas diproses secara siklus, adalah perlu memasukkan biaya untuk memulai siklus berikutnya. Ini adalah biaya akibat perubahan produksi komoditi terakhir ke komoditi pertama. Masalah ini dapat dianggap sebagai TSP pada graf n simpul dengan sisi yang bobotnya c_{ij} .

Pada persoalan TSP ini, jika setiap simpul mempunyai sisi ke simpul yang lain, maka graf yang merepresentasikannya adalah graf lengkap berbobot. Pada sembarang graf lengkap dengan n buah simpul ($n > 2$), jumlah sirkuit Hamilton yang berbeda adalah $(n - 1)!/2$. Rumus ini dihasilkan dari kenyataan bahwa dimulai dari sembarang simpul kita mempunyai $n - 1$ buah sisi untuk dipilih dari simpul pertama, $n - 2$ sisi dari simpul kedua, $n - 3$ dari simpul ketiga, dan seterusnya. Ini adalah pilihan yang independen, sehingga kita memperoleh $(n - 1)!$ pilihan. Jumlah itu harus dibagi dengan 2, karena tiap sirkuit Hamilton terhitung dua kali, sehingga semuanya ada $(n - 1)!/2$ buah sirkuit Hamilton.

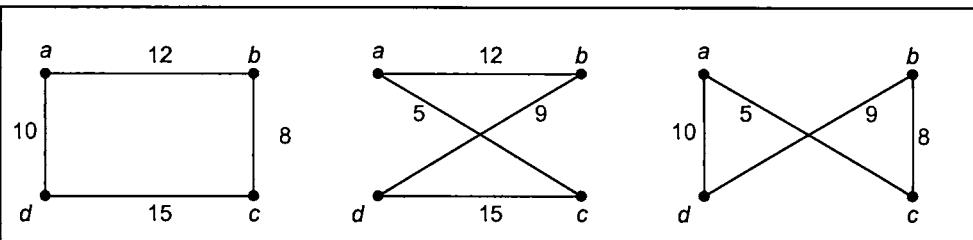


Gambar 8.69 Graf lengkap dengan 4 simpul. Tiap sisi diberi bobot

Tinjau graf lengkap dengan $n = 4$ simpul seperti yang ditunjukkan pada Gambar 8.69. Graf tersebut memiliki $(4 - 1)!/2 = 3$ sirkuit Hamilton, yaitu:

- $$S_1 = (a, b, c, d, a)$$
- atau
- (a, d, c, b, a)
- dengan panjang rute
- $= 10 + 12 + 8 + 15 = 45$
-
- $$S_2 = (a, c, d, b, a)$$
- atau
- (a, b, d, c, a)
- dengan panjang rute
- $= 12 + 5 + 9 + 15 = 41$
-
- $$S_3 = (a, c, b, d, a)$$
- atau
- (a, d, b, c, a)
- dengan panjang sirkuit
- $= 10 + 5 + 9 + 8 = 32$

Gambar sirkuit Hamilton-nya diperlihatkan pada Gambar 8.70.



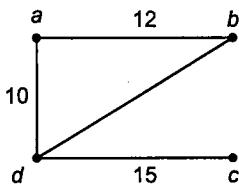
Gambar 8.70 Tiga buah sirkuit Hamilton berbeda dari graf lengkap pada Gambar 8.67.

Jadi, sirkuit Hamilton terpendek adalah $S_3 = (a, c, b, d, a)$ atau (a, d, b, c, a) dengan panjang sirkuit $= 10 + 5 + 9 + 8 = 32$. Ini adalah solusi persoalan TSP untuk graf berbobot graf pada Gambar 8.69.

Persoalan TSP adalah persoalan yang sulit (*hard problem*) dipandang dari sudut komputasinya. Artinya, secara teoritis, TSP dapat dipecahkan dengan mengenumerasi $(n - 1)!/2$ buah sirkuit Hamilton, menghitung panjang rute masing-masing sirkuit, dan kemudian memilih sirkuit yang memiliki panjang ruter terpendek. Untuk n yang besar, jumlah sirkuit Hamilton yang harus diperiksa tentu saja sangat banyak. Misalnya untuk jumlah simpul $n = 20$ akan terdapat $(19!)/2$ sirkuit Hamilton atau sekitar 6×10^{16} penyelesaian.

Catatlah bahwa persoalan TSP tidak hanya berlaku pada graf lengkap, tetapi juga pada graf tidak lengkap sekalipun asalkan sirkuit Hamiltonnya ada. Graf pada Gambar 8.71 bukan graf lengkap namun memiliki sirkuit Hamilton.

Belum ada algoritma yang mangkus untuk memecahkan persoalan TSP untuk n sembarang, meskipun banyak usaha-usaha yang telah dibuat. Beberapa metode heuristik seperti *branch and bound* untuk memecahkan persoalan TSP dapat dibaca pada buku [HOR78].



Gambar 8.71 Graf tidak lengkap tetapi memiliki sirkuit Hamilton

8.15 Persoalan Tukang Pos Cina

Persoalan Tukang Pos Cina (*Chinese Postman Problem*) pertama kali dikemukakan oleh Mei Gan (berasal dari Cina) pada tahun 1962. Ia mengemukakan masalah yang disebut persoalan tukang pos Cina. Masalahnya tukang pos adalah sebagai berikut:

Seorang tukang pos akan mengantar surat ke alamat-alamat sepanjang jalan di suatu daerah. Bagaimana ia merencanakan rute perjalanannya supaya ia melewati setiap jalan tepat sekali dan kembali lagi ke tempat awal keberangkatan.

Persoalan tukang pos Cina tidak lain menentukan sirkuit Euler di dalam graf. Jika peta jalan tempat tukang pos mengantarkan surat merupakan graf Euler, maka sirkuit Eulernya mudah ditemukan. Tetapi jika grafnya bukan graf Euler maka beberapa sisi di dalam graf harus dilalui lebih dari sekali. Karena itu, kita harus menemukan sirkuit yang terpendek yang harus dilalui oleh tukang pos yang mengunjungi setiap jalan paling sedikit satu kali. Oleh karena itu, masalah tukang pos Cina dirumuskan kembali sebagai berikut:

Seorang tukang pos akan mengantar surat ke alamat-alamat sepanjang jalan di suatu daerah. Bagaimana ia merencanakan rute perjalanannya yang mempunyai jarak terpendek supaya ia melewati setiap jalan paling sedikit sekali dan kembali lagi ke tempat awal keberangkatan.

Algoritma pemecahan masalah tukang pos Cina dapat dipelajari di dalam [GIB85].

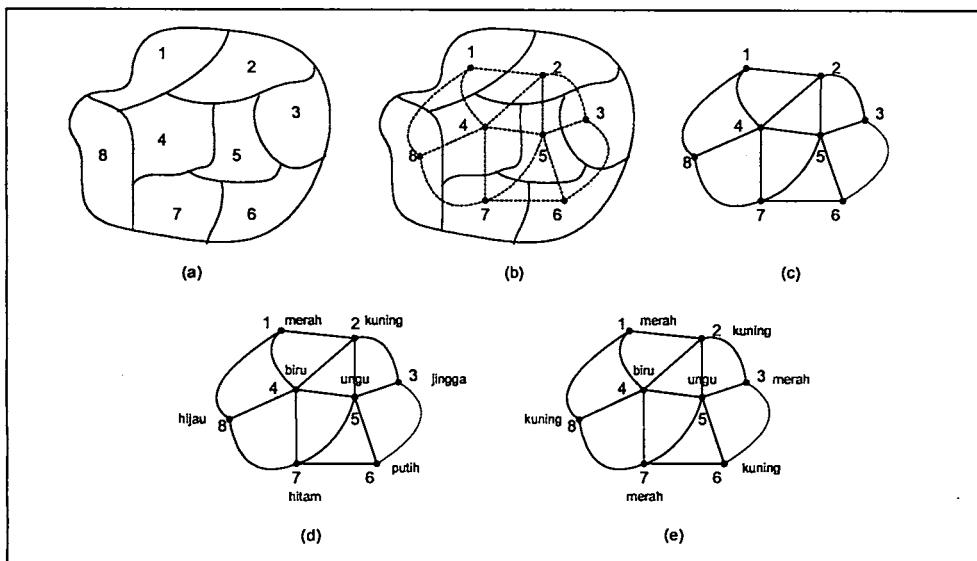
8.16 Pewarnaan Graf

Ada tiga macam persoalan pewarnaan graf (*graph colouring*), yaitu pewarnaan simpul, pewarnaan sisi, dan pewarnaan wilayah (*region*). Buku ini hanya membahas pewarnaan simpul saja.

DEFINISI 8.20. Pewarnaan simpul adalah memberi warna pada simpul-simpul di dalam graf sedemikian sehingga setiap dua simpul bertetangga mempunyai warna yang berbeda.

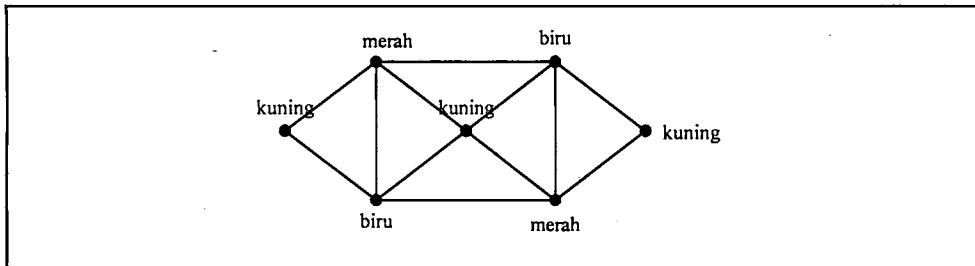
Salah satu terapan penting pewarnaan graf adalah mewarnai peta (*colouring of map*). Misalkan kita diminta mewarnai sebuah peta yang terdiri atas sejumlah wilayah. Wilayah pada peta dapat menyatakan provinsi, kabupaten, negara, dan lain-lain (untuk mudahnya, perhatikanlah peta negara AS yang terdiri atas 50 buah negara bagian). Kita diminta mewarnai setiap wilayah di dalam peta sedemikian sehingga tidak ada dua wilayah bertetangga yang mempunyai warna sama. Satu cara untuk menjamin bahwa dua buah wilayah bertetangga tidak mempunyai warna yang sama adalah menggunakan warna yang berbeda untuk setiap wilayah.

Di dalam persoalan pewarnaan graf, kita tidak hanya sekedar mewarnai simpul-simpul dengan warna berbeda dari warna simpul tetangganya saja, namun kita juga menginginkan jumlah macam warna yang digunakan sesedikit mungkin. Dihubungkan dengan masalah pewarnaan peta di atas, 8 warna yang berbeda untuk setiap simpul di dalam Gambar 8.72 jelas tidak mangkus. Sebenarnya kita dapat menggunakan 4 warna saja untuk mewarnai peta tersebut (Gambar 8.72 (e)).



Gambar 8.72 (a) Peta, (b) Peta dan graf yang merepresentasikannya, (c) Graf yang merepresentasikan peta, (d) Pewarnaan simpul, setiap simpul mempunai warna berbeda, (e) Empat warna sudah cukup untuk mewarnai 8 simpul

Jumlah warna minimum yang dapat digunakan untuk mewarnai simpul disebut **bilangan kromatik** graf G , disimbolkan dengan $\chi(G)$. Suatu graf G yang mempunyai bilangan kromatis k dilambangkan dengan $\chi(G) = k$. Jadi, graf pada Gambar 8.72(c) memiliki $\chi(G) = 4$, sedangkan graf pada Gambar 8.73 memiliki $\chi(G) = 3$.



Gambar 8.73 Tiga buah warna sudah cukup untuk mewarnai graf ini ($\chi(G) = 3$).

Beberapa graf tertentu dapat langsung ditentukan bilangan kromatiknya. Graf kosong N_n memiliki $\chi(G) = 1$, karena semua simpul tidak terhubung, jadi untuk mewarnai semua simpul cukup dibutuhkan satu warna saja. Graf lengkap K_n memiliki $\chi(G) = n$ sebab semua simpul saling terhubung sehingga diperlukan n buah warna. Graf bipartit $K_{m,n}$ mempunyai $\chi(G) = 2$, satu untuk simpul-simpul di himpunan V_1 dan satu lagi untuk simpul-simpul di V_2 . Graf lingkaran dengan n ganjil memiliki $\chi(G) = 3$, sedangkan jika n genap maka $\chi(G) = 2$. Sembarang pohon T memiliki $\chi(T) = 2$. Untuk graf-graf yang lain tidak dapat dinyatakan secara umum bilangan kromatiknya.

Masalah menentukan bilangan kromatik graf planar yang direpresentasikan sebagai graf bidang sudah banyak diteliti oleh para ilmuwan. Perkembangan hasil penelitian untuk menemukan bilangan kromatik itu dapat disajikan dalam teorema yang beruntun di bawah ini (pembuktianya tidak diberikan di sini).

Mula-mula, hasil penelitian tersebut dirangkum dalam teorema berikut:

TEOREMA 8.10. Bilangan kromatik graf planar tidak lebih dari 6.

Teorema 8.10 ini kemudian diperbaiki menjadi Teorema 8.11:

TEOREMA 8.11. Bilangan kromatik graf planar tidak lebih dari 5.

Puncak dari persoalan pewarnaan graf ini muncul pada tahun 1976 sebagai hasil dari pemecahan **persoalan 4-warna** (*four colour problem*), yaitu salah satu persoalan yang sangat terkenal dalam teori graf (persoalan 4-warna diajukan pada tahun 1852). Persoalan 4-warna berbunyi: dapatkah sembarang graf planar diwarnai hanya dengan 4 warna saja? Jawaban dari persoalan ini ditemukan oleh

Appel dan Haken yang menggunakan komputer untuk menganalisis hampir 2000 graf yang melibatkan jutaan kasus [LIP92]. Jawaban ini dinyatakan dalam Teorema 8.12 berikut:

TEOREMA 8.12. Bilangan kromatik graf planar tidak lebih dari 4.

Pembuktian Teorema 8.12 ini sangat rumit dan menurut catatan sejarah pembuktianya membutuhkan ratusan lembar kertas untuk menuliskannya.

Pemecahan persoalan pewarnaan graf sangat berjasa dalam menentukan jumlah minimum warna yang dibutuhkan untuk mewarnai sembarang peta. Selama bertahun-tahun, lima buah warna adalah jumlah yang cukup untuk mewarnai sembarang peta. Setelah beberapa ratus tahun, persoalan ini berhasil dipecahkan oleh K. Appel dan W. Haken seperti yang dikemukakan pada Teorema 8.12 di atas. Mereka berhasil memperlihatkan bahwa empat buah warna sudah cukup untuk mewarnai sembarang graf planar.

Contoh 8.44

Persoalan yang mempunyai karakteristik seperti pewarnaan graf adalah persoalan menentukan jadwal ujian. Misalkan terdapat delapan orang mahasiswa ($1, 2, \dots, 8$) dan lima buah mata kuliah yang dapat dipilihnya (A, B, C, D, E). Tabel berikut memperlihatkan matriks lima mata kuliah dan delapan orang mahasiswa. Angka 1 pada elemen (i, j) berarti mahasiswa i memilih mata kuliah j , sedangkan angka 0 menyatakan mahasiswa i tidak memilih mata kuliah j .

	A	B	C	D	E
1	0	1	0	0	1
2	0	1	0	1	0
3	0	0	1	1	0
4	1	1	0	0	0
5	0	1	0	1	0
6	0	0	1	1	0
7	1	0	1	0	0
8	0	0	1	1	0

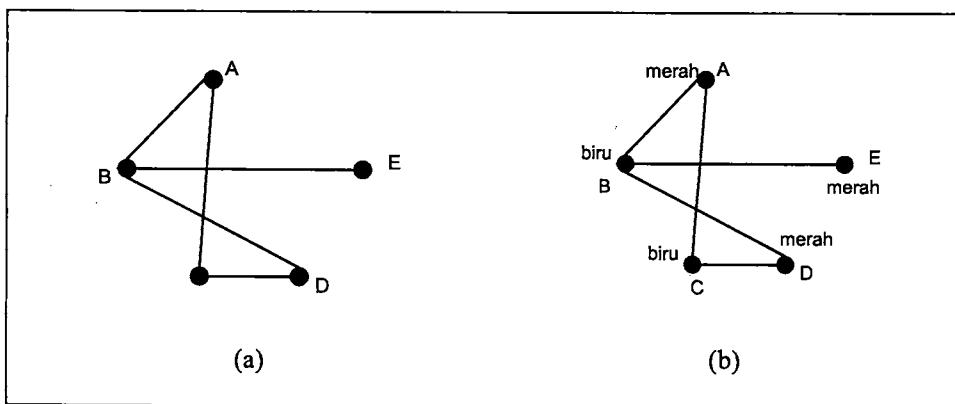
Berdasarkan tabel tadi, administratur mata kuliah ingin menentukan jadwal ujian sedemikian sehingga semua mahasiswa dapat mengikuti ujian mata kuliah yang diambilnya tanpa bertabrakan waktunya dengan jadwal ujian mata kuliah lain yang juga diambilnya. Pendek kata, jika ada mahasiswa yang mengambil dua buah mata kuliah atau lebih, jadwal ujian mata kuliah tersebut harus pada waktu yang tidak bersamaan. Ujian dua buah mata kuliah dapat dijadwalkan pada waktu yang sama jika tidak ada mahasiswa yang sama yang mengikuti ujian dua buah mata kuliah itu. Berapa paling sedikit jumlah hari yang dibutuhkan untuk jadwal ujian tersebut?

Penyelesaian:

Penyelesaian persoalan menentukan jadwal ujian semua mata kuliah sama dengan menentukan bilangan kromatis graf. Kita dapat menggambarkan graf yang menyatakan penjadwalan ujian. Simpul-simpul pada graf menyatakan mata kuliah. Sisi yang menghubungkan dua buah simpul menyatakan ada mahasiswa yang memilih kedua mata kuliah itu. Graf yang merepresentasikan persoalan di atas dibuat ditunjukkan pada Gambar 8.74(a).

Berdasarkan graf tersebut kita menyimpulkan, bahwa apabila terdapat dua buah simpul dihubungkan oleh sisi, maka ujian kedua mata kuliah itu tidak dapat dibuat pada waktu yang sama. Warna-warna yang berbeda dapat diberikan pada simpul graf yang menunjukkan bahwa waktu ujiannya berbeda. Diinginkan jadwal ujiannya sesedikit mungkin untuk memudahkan pelaksanaanya. Jadi kita harus menentukan bilangan kromatis graf.

Bilangan kromatik graf pada Gambar 8.75 adalah 2. Jadi, ujian mata kuliah A , E , dan D dapat dilaksanakan bersamaan, sedangkan ujian mata kuliah B dan C dilakukan bersamaan tetapi pada waktu yang berbeda dengan mata kuliah A , E , dan D . Gambar 8.74(b) memperlihatkan graf yang telah diwarnai. ■



Gambar 8.74. (a) Graf persoalan penjadwalan ujian 5 mata kuliah untuk 8 orang mahasiswa
 (b) Hasil pewarnaan pada simpul-simpul graf

Untuk graf dengan jumlah simpul sedikit, kita dapat menentukan bilangan kromatiknya dengan mudah. Tetapi, untuk graf yang besar, kita perlu membuat program komputer untuk menentukan bilangan kromatik tersebut.

Algoritma Pewarnaan Graf

Algoritma Welch-Powell dapat digunakan untuk mewarnai sebuah graf G secara mangkus. Algoritma ini hanya memberikan batas atas untuk $\chi(G)$, yaitu bahwa

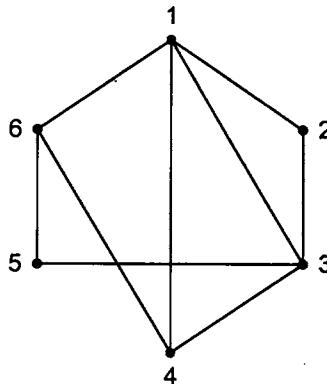
algoritma tidak selalu memberikan jumlah warna minimum yang diperlukan untuk mewarnai G [LIP92]. Algoritma Welch-Powell adalah sebagai berikut:

ALGORITMA Welch-Powell

1. Urutkan simpul-simpul dari G dalam derajat yang menurun (urutan seperti ini mungkin tidak unik karena beberapa simpul mungkin berderajat sama).
2. Gunakan satu warna untuk mewarnai simpul pertama (yang mempunyai derajat tertinggi) dan simpul-simpul lain (dalam urutan yang berurut) yang tidak bertetangga dengan simpul pertama ini.
3. Mulai lagi dengan simpul derajat tertinggi berikutnya di dalam daftar terurut yang belum diwarnai dan ulangi proses pewarnaan simpul dengan menggunakan warna kedua.
4. Ulangi penambahan warna-warna sampai semua simpul telah diwarnai.

Contoh 8.45

Gunakan algoritma Welch-Powell untuk mewarnai graf di bawah ini.



Penyelesaian:

Urutkan simpul-simpul di dalam graf berdasarkan derajat yang menurun. Warnai simpul 1 dengan warna a . Simpul yang tidak bertetangga dengan 1 adalah simpul 5, warnai simpul 5 ini dengan warna a . Simpul berikutnya di dalam daftar yang belum diwarnai adalah simpul 3. Warnai simpul 3 dengan warna b , dan simpul yang tidak bertetangga dengan 3 adalah simpul 6, warna simpul 6 ini juga dengan warna b . Simpul berikutnya yang belum diwarnai adalah simpul 4. Warnai simpul 4 dengan warna c , begitu juga simpul yang tidak bertetangga dengan simpul 4 diberi warna c . Sekarang semua simpul sudah diwarnai.

Simpul	1	3	6	4	2	5
Derajat	4	4	3	3	2	2
Warna	a	b	b	c	c	a

Perhatikan bahwa simpul 1, 2, dan 3 terhubung satu sama lain, sehingga paling sedikit dibutuhkan 3 warna berbeda untuk mewarnai graf G . Jadi, $\chi(G) = 3$. ■

8.17 Ragam Soal dan Penyelesaian

Contoh 8.46

Dapatkan kita menggambar graf teratur berderajat 3 dengan 7 buah simpul? Mengapa?

Penyelesaian:

Tidak, karena menurut aturan Lemma jabat tangan, jumlah derajat semua simpul pada suatu graf adalah genap, yaitu 2 kali jumlah sisi di dalam graf tersebut. Pada graf tersebut:

$$\begin{aligned}e &= nr / 2 \\ \Leftrightarrow 2e &= nr \\ \Leftrightarrow 2e &= 3 \cdot 7 \\ \Leftrightarrow 2e &= 21 \quad \text{jelas tidak memenuhi syarat karena 2 kali jumlah sisi pada graf tersebut ganjil.}\end{aligned}$$

Pendekatan lain:

$e = 21 / 2$ Jelas bahwa jumlah sisi dari suatu graf tidak mungkin berupa pecahan, maka tidak mungkin menggambar graf teratur berderajat 3 dengan 7 buah simpul. ■

Contoh 8.47

Tentukan jumlah simpul pada graf sederhana bila mempunyai 20 buah sisi dan tiap simpul berderajat sama.

Penyelesaian:

Pada graf teratur derajat r dengan n buah simpul, jumlah sisinya adalah

$$e = nr/2$$

sehingga

$$n = 2e/r = 2 \cdot 20/r = 40/r$$

Tinjau kasus-kasus nilai r sebagai berikut:

- Untuk $r = 1$, maka $n = 40$; akan terbentuk graf tidak terhubung yang masing-masing simpulnya berderajat 1, jumlah sisinya adalah $40/2 = 20$ (memenuhi)
- Untuk $r = 2$, maka $n = 20$, akan terbentuk graf lingkaran dengan sisi 20 (memenuhi)
- Untuk $r = 3, 6, 7, 9$ tidak mungkin sebab hasil pembagian $(40/r)$ tidak bulat.
- Untuk r yang lebih besar lagi tidak akan mungkin lagi terbentuk graf sederhana sebab jumlah simpulnya akan lebih kecil sehingga maksimum sisi yang diizinkan juga semakin kecil.

Jadi r yang memenuhi adalah $\{1, 2, 4, 5\}$, dan jumlah simpul di dalam graf adalah $\{40, 20, 10, 8\}$. ■

Contoh 8.48

Berapa jumlah minimum simpul yang diperlukan agar sebuah graf dengan 6 buah sisi menjadi planar? Ulangi soal yang sama untuk 11 buah sisi.

Penyelesaian:

Gunakan ketidaksamaan Euler $e \leq 3n - 6$.

Untuk $e = 6$,

$$e \leq 3n - 6$$

$$6 \leq 3n - 6$$

$$12 \leq 3n$$

$$4 \leq n$$

berarti jumlah minimum simpul adalah 4.

Untuk $e = 11$,

$$e \leq 3n - 6$$

$$11 \leq 3n - 6$$

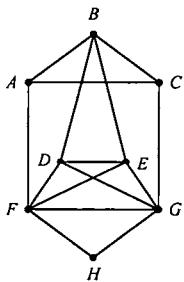
$$17 \leq 3n$$

$$17/3 \leq n$$

berarti jumlah minimum simpul adalah 6. ■

Contoh 8.49

Diberikan gambar sebuah graf G seperti di bawah ini.



- Tunjukkan dengan ketidaksamaan Euler bahwa graf G tidak planar.
- Tunjukkan dengan Teorema Kuratowski bahwa graf G tidak planar.

Penyelesaian:

- (a) Jika menggunakan rumus ketidaksamaan Euler $e \leq 3n - 6$ maka akan terlihat bahwa graf memenuhi ketidaksamaan tersebut (padahal graf tidak planar):

$$e \leq 3n - 6$$

$$15 \leq 3 \cdot 8 - 6$$

$$15 \leq 24 - 6$$

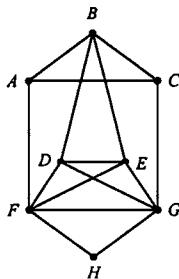
$$15 \leq 18$$

Untuk menunjukkan bahwa graf tidak planar kita membuat asumsi bahwa setiap daerah pada graf planar dibatasi oleh paling sedikit 4 buah sisi. Ketidaksamaan yang dihasilkan (lihat upa-bab 8.9) adalah:

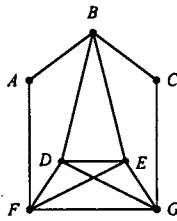
$$\begin{aligned}e &\leq 2n - 4 \\15 &\leq 2 \cdot 8 - 4 \\15 &\leq 16 - 4 \\15 &\leq 12 \text{ (benar)}\end{aligned}$$

Jadi, memang benar graf G tidak planar.

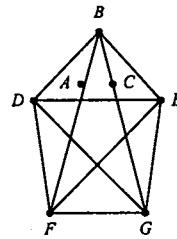
- (b) Dengan menggunakan teorema Kuratowski kita dapat membuktikan bahwa graf G mengandung upagraph yang homeomorfik dengan graf $K_{3,3}$ atau K_5 .



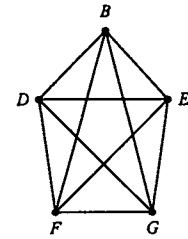
G



G_1 (upagraph dari G)



G_2 (isomorfik dengan G_1)



G_2 homeomorfik dengan K_5 (dengan membuang simpul A dan C yang berderajat 2)

Contoh 8.50

Berapa bilangan kromatis graf pada soal 8.49 di atas?

Penyelesaian:

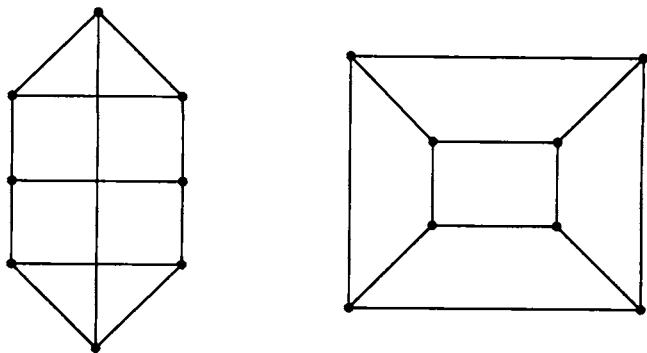
Bilangan kromatis graf tersebut adalah 4. Sebagai contoh simpul dapat dibagi menjadi 4 warna berbeda: (D, A, H) , (C, E) , (B, G) , (F) . ■

Contoh 8.50

Gambarkan 2 buah graf yang isomorfik dengan graf teratur berderajat 3 yang mempunyai 8 buah simpul.

Penyelesaian:

Jawaban untuk soal ini tidak unik. Ada banyak graf teratur dengan 8 simpul yang saling isomorfik satu sama lain. Sepasang di antaranya adalah seperti berikut ini:

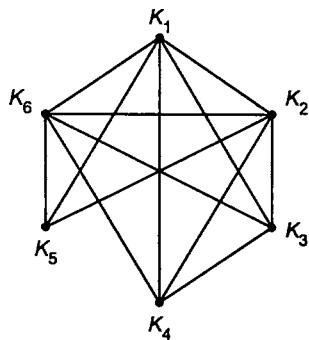


Contoh 8.5]

Sebuah departemen mempunyai 6 kelompok kerja yang setiap bulannya masing-masing selalu mengadakan rapat satu kali. Keenam kelompok kerja dengan masing-masing anggotanya adalah: $K_1 = \{Amir, Budi, Yanti\}$, $K_2 = \{Budi, Hasan, Tommy\}$, $K_3 = \{Amir, Tommy, Yanti\}$, $K_4 = \{Hasan, Tommy, Yanti\}$, $K_5 = \{Amir, Budi\}$, $K_6 = \{Budi, Tommy, Yanti\}$. Berapa banyak waktu rapat berbeda yang harus direncanakan sehingga tidak ada anggota kelompok kerja yang dijadwalkan rapat pada waktu yang sama. Gambarkan graf yang merepresentasikan persoalan ini lalu (jelaskan sisi menyatakan apa, simpul menyatakan apa) tentukan jumlah waktu rapat ini.

Penyelesaian:

Graf yang memodelkan persoalan di atas adalah seperti di bawah ini, yang dalam hal ini simpul menyatakan kelompok sedangkan sisi menyatakan adanya anggota kelompok yang sama.



Jika ada sisi yang menghubungkan 2 kelompok berarti kelompok tersebut tidak boleh rapat pada waktu yang sama. Untuk mencari jumlah minimum waktu rapat yang harus disediakan kita dapat menggunakan cara yang sama seperti mencari bilangan kromatis dari graf tersebut. Setiap warna yang berbeda mewakili satu waktu rapat yang dibutuhkan. Bilangan kromatis graf tersebut adalah 5, sehingga waktu rapat yang harus disediakan adalah 5.

Contoh 8.52

- (a) Apakah K_{13} memiliki sirkuit Euler? Sirkuit Hamilton?
(b) Ulangi pertanyaan (a) untuk K_{14}

Penyelesaian:

Sirkuit Euler hanya ada jika semua simpul di dalam graf berderajat genap. Untuk sirkuit Hamilton, setiap graf lengkap adalah graf Hamilton (memiliki sirkuit Hamilton).

- (a) K_{13} memiliki sirkuit Euler sebab setiap simpul pada K_{13} berderajat 12 (genap); K_{13} juga memiliki Sirkuit Hamilton sebab K_{13} adalah graf lengkap (setiap graf lengkap adalah graf Hamilton).
(b) K_{14} tidak memiliki sirkuit Euler sebab setiap simpul pada K_{14} berderajat 13 (ganjil). K_{14} memiliki Sirkuit Hamilton sebab K_{14} adalah graf lengkap.
-

Contoh 8.53

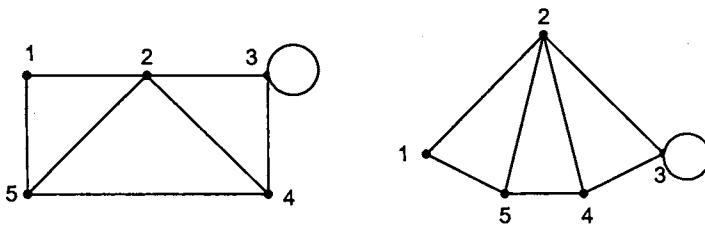
Diketahui matriks ketetanggaan (*adjacency matrices*) dari sebuah graf tidak berarah:

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

Gambarkan dua buah graf yang isomorfik yang bersesuaian dengan matriks ketetanggaan di atas.

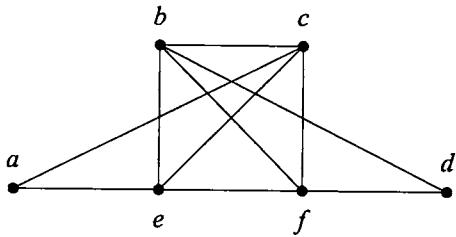
Penyelesaian:

Jawaban tidak tunggal, salah satunya seperti di bawah ini:



Contoh 8.54

Perlihatkan dengan ketidaksamaan Euler bahwa graf berikut planar, lalu gambarkan graf planar tersebut sebagai graf bidang.



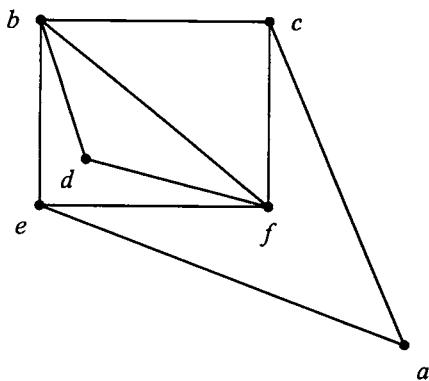
Penyelesaian:

Diketahui $e = 10$ dan $n = 6$, maka:

$$\begin{aligned} e &\leq 3n - 6 \\ 10 &\leq 3 \cdot 6 - 6 \\ 10 &\leq 12 \quad (\text{benar}) \end{aligned}$$

Jadi graf tersebut adalah graf planar.

Salah satu gambar graf bidangnya (mungkin ada cara penggambaran lain):



■

Contoh 8.55

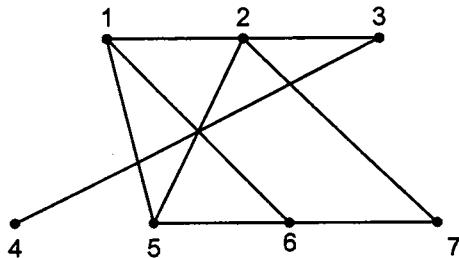
Di suatu negara terdapat 7 buah stasiun televisi. Pemerintah menetapkan aturan bahwa dua stasiun televisi yang berjarak ≤ 150 km tidak boleh beroperasi pada saluran frekuensi (UHF) yang sama. Tabel di bawah ini memperlihatkan jarak (km) antar stasiun televisi.

	1	2	3	4	5	6	7
1	-	85	175	200	50	100	230
2	-	-	125	175	100	160	145
3	-	-	-	100	200	250	160
4	-	-	-	-	210	220	180
5	-	-	-	-	-	100	235
6	-	-	-	-	-	-	120
7	-	-	-	-	-	-	-

- (a) Gambarkan graf yang memodelkan persoalan ini. Jelaskan pula arti setiap simpul dan sisi pada graf anda.
- (b) Berapa banyak frekuensi berbeda yang dibutuhkan bagi ketujuh stasiun TV tersebut sesuai dengan aturan Pemerintah? Termasuk kategori mana persolan ini?

Penyelesaian:

- (a) Graf yang memodelkan persoalan ini adalah sebagai berikut: simpul merepresentasikan stasiun televisi, sedangkan sisi merepresentasikan bahwa dua buah simpul yang terhubung oleh sisi tersebut tidak boleh beroperasi pada saluran frekuensi yang sama sebab jarak keduanya ≤ 150 km.



- (b) Banyak frekuensi yang berbeda adalah adalah minimum 3 (bilangan kromatis garf) dan maksimum 7 (jumlah warna maksimum). Persoalan ini termasuk ke dalam kategori Pewarnaan Graf. ■

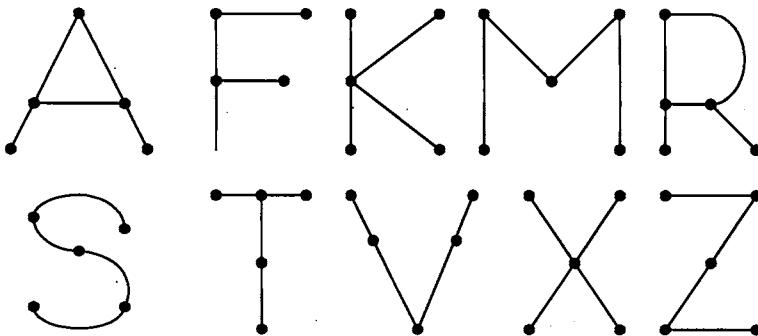
Soal Latihan

1. Dalam sebuah pesta, sepuluh orang saling berjabat tangan. Tiap orang hanya berjabat tangan satu kali dengan orang lainnya. Hitung jumlah jabat tangan yang terjadi (Petunjuk: modelkan persoalan ini ke dalam graf)
2. Tiga pasang suami istri yang sedang menempuh perjalanan sampai ke sebuah sungai. Di situ mereka menemukan sebuah perahu kecil yang hanya bisa membawa tidak lebih dari dua orang setiap kali menyeberang. Penyeberangan sungai dirumitkan oleh kenyataan bahwa para suami sangat pencemburu dan tidak mau meninggalkan istri-istri mereka jika ada lelaki lain. Buatlah sebuah graf untuk menunjukkan bagaimana penyeberangan itu bisa dilakukan.
3. Empat buah tim bola basket mengikuti kejuaraan antar Universitas. Pertandingan menggunakan sistem *round-robin*, yaitu setiap tim bertemu dengan tim lainnya satu kali. Misalkan empat tim tersebut dinamai A , B , C , dan D . Gambarkan graf berarah yang menyatakan satu set pertandingan (graf tersebut dinamakan graf turnamen – *tournament graph*).
4. Sebuah graf akan dibentuk dari 25 buah sisi. Berapa jumlah maksimum simpul di dalam graf sederhana yang dapat dibuat dari 25 buah sisi tersebut?
6. Ada n buah komputer yang akan dihubungkan dengan sejumlah kabel, baik secara langsung atau terhubung melalui komputer lainnya. Berapa jumlah minimum kabel yang dibutuhkan?
7. Tentukan jumlah simpul pada graf sederhana bila mempunyai 12 buah sisi dan tiap simpul berderajat dua.
8. Tentukan jumlah simpul pada graf sederhana bila mempunyai 20 buah sisi dan tiap simpul berderajat sama
9. Tunjukkan bahwa derajat maksimum sembarang simpul pada sembarang graf sederhana dengan n simpul adalah $n - 1$.
10. Berapa jumlah maksimum dan jumlah minimum simpul pada graf sederhana yang mempunyai 12 buah sisi dan tiap simpul berderajat ≥ 3 ?
11. Gambarkan dua buah graf teratur berderajat 3 dengan 6 buah simpul.
12. Dapatkah kita menggambar graf teratur derajat 3 dengan 7 buah simpul?

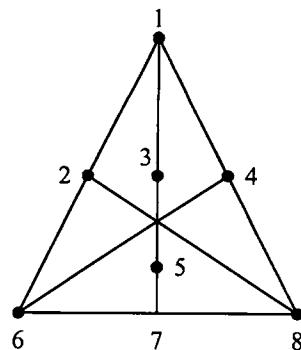
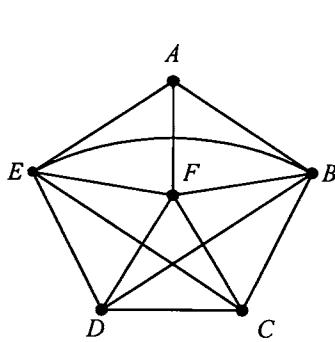
13. Ada n buah kota yang dihubungkan dengan sebuah jaringan k jalan raya (jalan raya dalam hal ini didefinisikan sebagai jalan antara dua buah kota yang tidak melalui kota-kota antara). Tunjukkan bahwa jika $k > (n - 1)(n - 2)/2$, maka kita selalu dapat menempuh perjalanan antara sembarang dua buah kota melalui jalan-jalan raya penghubungnya.
14. Dapatkah graf tidak berarah sederhana dengan 8 simpul memiliki 40 buah sisi?
15. Gambarkan dua buah graf dengan lima buah simpul yang isomorfik.
16. Diketahui matriks ketetanggaan (*adjacency matrices*) dari sebuah graf tidak berarah:

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

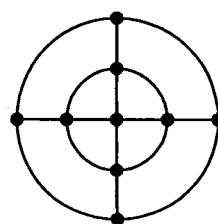
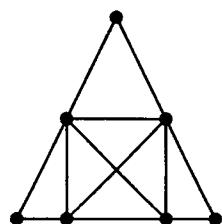
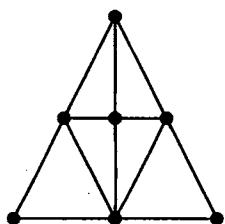
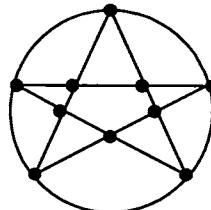
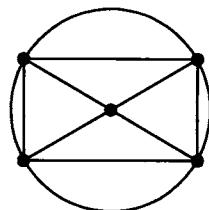
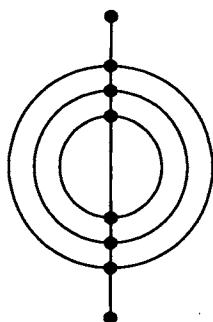
17. Gambarkan dua buah graf yang isomorfik yang bersesuaian dengan matriks ketetangan di atas.
18. Manakah di antara sepuluh graf karakter di bawah ini yang isomorfik dengan huruf M?



19. Perlihatkan dengan Teorema Kuratowski bahwa dua buah graf di bawah ini tidak planar!



20. Untuk n apakah graf lengkap K_n merupakan graf Euler?
21. Perusahaan kontraktor Euler dikontrak untuk membangun sebuah jembatan tambahan di Konigsberg sedemikian sehingga ada lintasan Euler yang melalui setiap jembatan. Di mana jembatan tambahan itu harus dibangun? Gambarkan grafnya.
22. Manakah di antara graf di bawah ini yang dapat dilukis tanpa mengangkat pensil sekalipun?



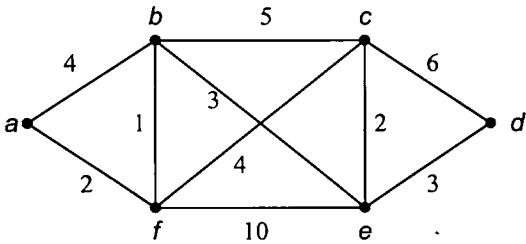
23. Gambarkan graf yang mempunyai lintasan Hamilton tetapi tidak memiliki sirkuit Hamilton.

24. (a) Gambarkan graf dengan lima simpul yang memiliki lintasan Euler dan juga sirkuit Hamilton.
(b) Gambarkan sebuah graf dengan 5 simpul yang mempunyai sirkuit Euler namun tidak mempunyai sirkuit Hamilton.
(c) Gambarkan sebuah graf dengan 5 simpul yang tidak mempunyai sirkuit Euler namun mempunyai sirkuit Hamilton.
(d) Gambarkan sebuah graf dengan 5 simpul yang tidak mempunyai sirkuit Euler maupun sirkuit Hamilton.
25. Di antara begitu banyak kamar/ruangan di sebuah rumah tua yang besar, ada hantu di setiap kamar/ruangan yang jumlah pintunya genap. Jika rumah tua itu hanya mempunyai satu pintu masuk, buktikan bahwa seseorang yang masuk dari luar selalu bisa mencapai sebuah kamar/ruangan yang tidak ada hantunya.
26. Misalkan G adalah graf dengan 11 buah simpul atau lebih. Tunjukkan bahwa G tidak planar.
27. Tunjukkan bahwa suatu graf planar terhubung dengan 6 simpul dan 12 buah sisi, setiap wilayahnya (*region*) dibatasi oleh 3 buah sisi.
28. Kubus- n adalah suatu graf tak-berarah dengan 2^n simpul yang diberi label dengan 2^n bilangan biner n -bit. Antara dua simpul ada sisi jika kedua label binernya berbeda hanya pada satu bit. Tunjukkan bahwa suatu kubus- n mempunyai sirkuit Hamilton untuk $n \geq 1$. (Suatu susunan 2^n bilangan biner n -bit sedemikian rupa sehingga dua bilangan yang berdekatan berbeda pada hanya satu bit dinamakan kode *Gray*)
29. Ada 6 jenis zat kimia yang perlu disimpan di dalam gudang. Beberapa pasang dari zat itu tidak dapat disimpan di dalam ruangan yang sama, karena campuran gasnya bersifat eksplosif (mudah meledak). Untuk zat yang semacam itu perlu dibangun ruang-ruang terpisah yang dilengkapi ventilasi dan penyedot udara keluar yang berlainan. Jika lebih banyak ruang yang dibutuhkan, berarti lebih banyak ongkos yang harus dikeluarkan. Karena itu perlu diketahui berapa banyak minimum ruangan yang diperlukan untuk dapat menyimpan semua zat kimia dengan aman. Berikut ini adalah daftar pasangan zat kimia yang tidak dapat disimpan di dalam ruangan yang sama:

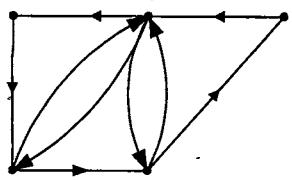
Zat kimia	Tidak dapat disimpan bersama zat kimia
A	B, D
B	A, D, E, F, G
C	E, G
D	A, F, B
E	B, C, G
F	B, D
G	C, E, G

Gambarkan graf yang menyatakan persoalan (jelaskan arti simpul dan sisi yang menghubungkan dua buah simpul). Pikirkan termasuk jenis manakah persoalan ini). Kemudian tentukan jumlah minimum ruangan yang dibutuhkan untuk menyimpan semua zat kimia di atas.

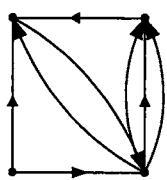
30. Tinjau graf berbobot di bawah ini. Simpul menyatakan kota, sisi menyatakan sarana transportasi yang menghubungkan kota, dan bobot menyatakan ongkos perjalanan antara dua kota bertetangga. Seorang pedagang berangkat dari kota a dan mengunjungi setiap kota lain tepat sekali dan kembali lagi ke kota a . Gambarkan semua kemungkinan lintasan perjalanan pedagang, lalu tentukan rute perjalanan yang termurah.



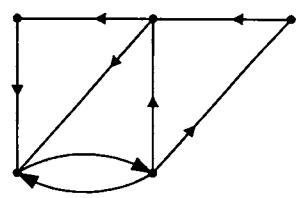
31. Dalam babak penyisihan kompetisi sepakbola yang menggunakan sistem kompetisi penuh, setiap tim bertanding dengan tim lainnya dua kali. Jika ada 20 tim, berapa banyak pertandingan yang harus diadakan? Graf apa yang terbentuk?
32. Ada tiga buah peta jalan di tiga buah komplek perumahan besar (Gambar a, b, dan c di bawah ini). Mana saja dari ketiga peta jalan tersebut yang dapat dilalui oleh Pak Pos sedemikian sehingga setiap ruas jalan hanya dilewati tepat sekali dan kembali lagi ke titik asal keberangkatan? Jelaskan alasan matematis atas jawaban anda tersebut.



(a)



(b)



(c)

BAB 9

Pohon

Jika berencana untuk satu tahun, tanamlah padi
Jika berencana untuk sepuluh tahun, tanamlah pohon
Namun jika berencana untuk seratus tahun, didiklah generasi penerus
(Confucius)

Graf terhubung yang tidak mengandung sirkuit disebut pohon. Di antara sekian banyak konsep dalam teori graf, konsep pohon (*tree*) mungkin merupakan konsep yang paling penting, karena terapannya yang luas dalam berbagai bidang ilmu. Banyak terapan, baik dalam bidang ilmu komputer maupun di luar bidang ilmu komputer, yang telah mengkaji pohon secara intensif sebagai objek matematika. Dalam kehidupan sehari-hari, orang telah lama menggunakan pohon untuk menggambarkan hirarkhi. Misalnya, pohon silsilah keluarga, struktur organisasi, organisasi pertandingan, dan lain-lain. Para ahli bahasa menggunakan pohon untuk menguraikan kalimat, yang disebut pohon parsing (*parse tree*).

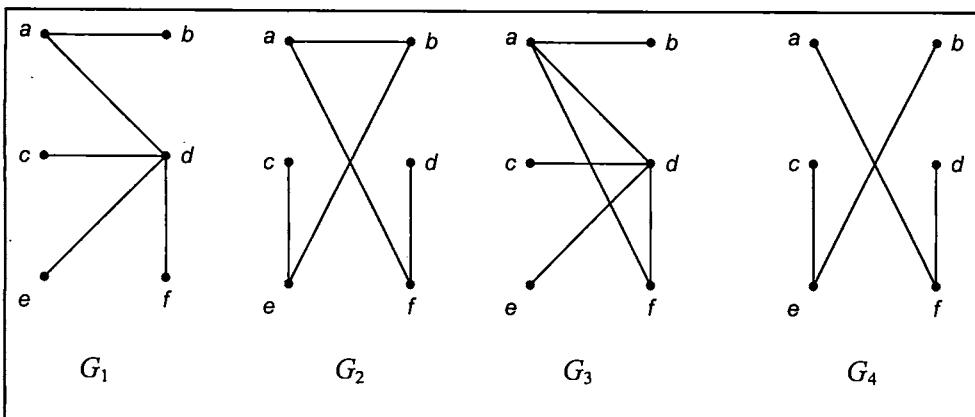
Pohon sudah lama digunakan sejak tahun 1857, ketika matematikawan Inggris Arthur Cayley menggunakan pohon untuk menghitung jumlah senyawa kimia. Bab ini membahas pohon dari sudut pandang teori graf. Pohon sebagai struktur data rekursif merupakan bagian dari perkuliahan Struktur Data.

9.1 Definisi Pohon

Pohon adalah graf yang khusus. Definisi pohon adalah sebagai berikut:

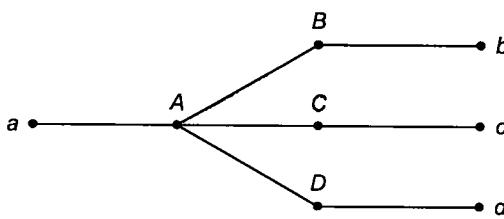
DEFINISI 9.1. Pohon adalah graf tak-berarah terhubung yang tidak mengandung sirkuit.

Menurut definisi 9.1 di atas, ada dua sifat penting pada pohon: terhubung dan tidak mengandung sirkuit. Pada Gambar 9.1, hanya G_1 dan G_2 yang pohon, sedangkan G_3 dan G_4 bukan pohon. G_3 bukan pohon karena ia mengandung sirkuit a, d, f, a , sedangkan G_4 bukan pohon karena ia tidak terhubung (anda jangan tertipu dengan persilangan dua buah sisi –dalam hal ini sisi (a, f) dan sisi (b, e) – karena titik silangnya bukan menyatakan simpul).



Gambar 9.1 G_1 dan G_2 adalah pohon, sedangkan G_3 dan G_4 bukan pohon

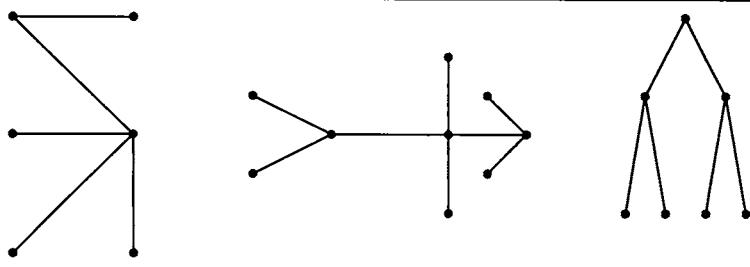
Berikut contoh pohon lainnya, diambil dari [LIU85], misalkan himpunan $V = \{a, b, B, c, C, d, D\}$ adalah empat pasangan suami-istri tukang gosip, dengan $a, b, c,$ dan d para suami, dan A, B, C, D para istri. Misalkan a menceritakan gosip lewat telpon kepada istrinya A , yang kemudian A menelepon para istri lainnya untuk menyebarkan gosip itu, dan setiap istri itu menelpon dan menceritakan gosip kepada suami masing-masing. Pohon pada Gambar 9.2 menunjukkan bagaimana gosip tersebut tersebar, dengan simpul menyatakan istri/suami dan sisi menyatakan panggilan telpon.



Gambar 9.2 Pohon penyebaran gosip

Karena definisi pohon diacu dari teori graf, maka sebuah pohon dapat mempunyai hanya sebuah simpul tanpa sebuah sisipun. Dengan kata lain, jika $G = (V, E)$ adalah pohon, maka V tidak boleh berupa himpunan kosong, namun E boleh kosong. Pada sebagian literatur, pohon yang dimaksudkan oleh Definisi 9.1 sering disebut juga **pohon bebas** (*free tree*) untuk membedakannya dengan **pohon berakar** (*rooted tree*). Pohon berakar akan kita bahas lebih lanjut di dalam upabab 9.4. Untuk selanjutnya, jika disebut pohon, maka yang dimaksudkan adalah pohon bebas.

Beberapa pohon dapat membentuk hutan. **Hutan** (*forest*) adalah kumpulan pohon yang saling lepas. Kita dapat juga menyatakan bahwa hutan adalah graf tak-terhubung yang tidak mengandung sirkuit, yang dalam hal ini setiap komponen di dalam graf terhubung tersebut adalah pohon. Gambar 9.3 berikut adalah hutan yang terdiri atas 3 buah pohon.



Gambar 9.3 Hutan yang terdiri dari tiga buah pohon

Pohon juga seringkali didefinisikan sebagai graf tak-berarah dengan sifat bahwa hanya terdapat sebuah lintasan unik antara setiap pasang simpul. Tinjau kembali graf G_1 pada Gambar 9.1. Setiap simpul di G_1 terhubung dengan lintasan tunggal. Sebagai contoh, dari b ke f hanya ada satu lintasan, yaitu b, a, d, f . Dari c ke a hanya ada satu lintasan, yaitu c, d, a . Demikian juga untuk setiap pasang simpul manapun di G_1 .

Selain itu, kita juga melihat bahwa di dalam pohon, jumlah sisinya adalah jumlah simpul dikurangi satu. Tinjau G_1 pada Gambar 9.1, jumlah simpul = 6 dan jumlah sisi = 5 (yaitu $6 - 1$). Sifat-sifat pohon lain yang dirangkum di dalam Teorema 9.1 pada upabab 9.2 di bawah ini.

9.2 Sifat-sifat Pohon

Sifat-sifat (*properties*) pohon dinyatakan dengan Teorema 9.1 di bawah ini.

TEOREMA 9.1. Misalkan $G = (V, E)$ adalah graf tak-berarah sederhana dan jumlah simpulnya n . Maka, semua pernyataan di bawah ini adalah ekivalen:

1. G adalah pohon.
2. Setiap pasang simpul di dalam G terhubung dengan lintasan tunggal.
3. G terhubung dan memiliki $m = n - 1$ buah sisi.
4. G tidak mengandung sirkuit dan memiliki $m = n - 1$ buah sisi.
5. G tidak mengandung sirkuit dan penambahan satu sisi pada graf akan membuat hanya satu sirkuit.
6. G terhubung dan semua sisinya adalah jembatan (jembatan adalah sisi yang bila dihapus menyebabkan graf terpecah menjadi dua komponen).

Semua butir pernyataan di atas juga dapat dianggap sebagai definisi lain dari pohon. Kita juga dapat membuktikan bahwa hutan F dengan k komponen mempunyai $m = n - k$ buah sisi.

Contoh 9.1

Sebuah pohon mempunyai $2n$ buah simpul berderajat 1, $3n$ buah simpul berderajat 2, dan n buah simpul berderajat 3. Tentukan banyaknya simpul dan sisi di dalam pohon itu.

Penyelesaian:

Menurut Lemma Jabat tangan, jumlah derajat semua simpul di dalam graf adalah 2 kali jumlah sisi di dalam graf tersebut. Jadi,

$$(2n \times 1) + (3n \times 2) + (n \times 3) = 2 |E| \Leftrightarrow 11n = 2 |E|$$

Menurut Teorema 9.1, jumlah sisi pada sebuah pohon adalah jumlah simpul minus satu, jadi

$$|E| = (2n + 3n + n) - 1 = 6n - 1$$

Dengan menyulihkan persamaan terakhir ke persamaan pertama,

$$11n = 2(6n - 1) = 12n - 2 \Leftrightarrow n = 2$$

Jadi, jumlah simpul pada pohon = $6n = 6 \times 2 = 12$ dan jumlah sisi = $6n - 1 = 11$.

9.3 Pewarnaan Pohon

Ditinjau dari teori pewarnaan graf, maka pohon mempunyai bilangan kromatik 2. Dengan kata lain, dua buah warna sudah cukup mewarnai simpul-simpul di pohon sedemikian sehingga tidak ada dua buah simpul bertetangga mempunyai warna sama.

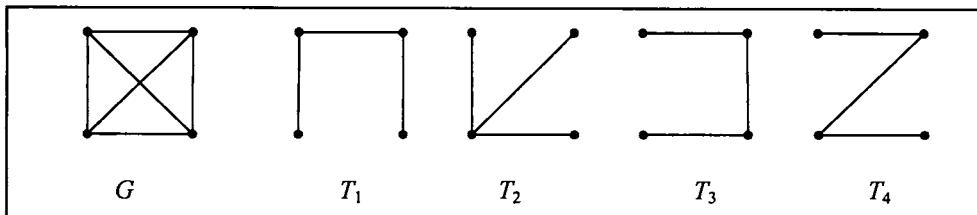
Pewarnaan pada pohon T dilakukan dengan cara berikut: petakan warna pertama pada sembarang sebuah simpul. Kemudian, petakan warna kedua pada simpul-simpul yang bertetangga dengan simpul pertama tadi. Selanjutnya, petakan warna pertama ke semua simpul yang bertetangga dengan simpul-simpul yang telah diberi warna kedua. Ulangi proses ini sampai semua simpul telah diwarnai. Sebagai contoh, tinjau G_1 pada Gambar 9.1. Simpul-simpul pada G_1 akan diwarnai dengan warna kuning dan biru. Simpul a dipilih pertama kali untuk diberi warna kuning. Kemudian simpul-simpul tetangga a , yaitu b dan d , diberi warna biru. Selanjutnya simpul-simpul yang beretangga dengan d , yaitu c , e , dan f , diberi warna kuning.

Kuning: a, c, e, f

Biru: b, d

9.4 Pohon Merentang

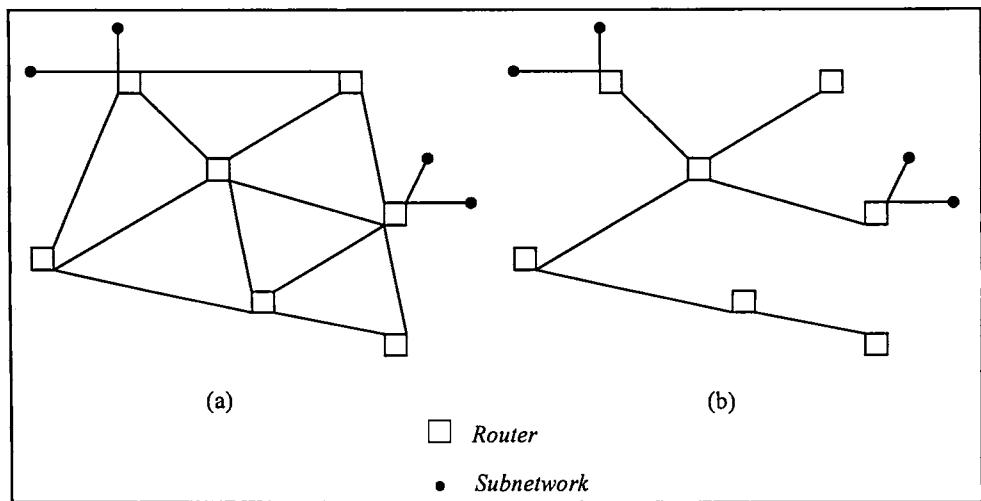
Misalkan $G = (V, E)$ adalah graf tak-berarah terhubung yang bukan pohon, yang berarti di G terdapat beberapa sirkuit. G dapat diubah menjadi pohon $T = (V_1, E_1)$ dengan cara memutuskan sirkuit-sirkuit yang ada. Caranya, mula-mula dipilih sebuah sirkuit, lalu hapus satu buah sisi dari sirkuit ini. G akan tetap terhubung dan jumlah sirkuitnya berkurang satu. Bila proses ini dilakukan berulang-ulang sampai semua sirkuit di G hilang, maka G menjadi sebuah pohon T , yang dinamakan **pohon merentang** (*spanning tree*). Disebut pohon merentang karena semua simpul pada pohon T sama dengan semua simpul pada graf G , dan sisi-sisi pada pohon $T \subseteq$ sisi-sisi pada graf G . Dengan kata lain, $V_1 = V$ dan $E_1 \subseteq E$ (Bandingkan dengan definisi upagraf rentang di Bab 8. Dengan kata lain, jika upagraf dari graf terhubung berbentuk pohon, maka upagraf merentang tersebut dinamakan pohon merentang). Gambar 9.4 adalah graf lengkap dengan empat buah pohon merentangnya (coba anda temukan seluruh pohon merentang lainnya).



Gambar 9.4 Graf lengkap G dan empat buah pohon merentangnya, T_1, T_2, T_3 , dan T_4

Aplikasi pohon merentang misalnya pada pemeliharaan jalan raya. Misalkan graf G pada Gambar 9.4 adalah peta jaringan jalan raya yang menghubungkan empat buah kota. Karena dana pemeliharaan yang terbatas, pemerintah daerah mempertimbangkan hanya memelihara jalan-jalan sesedikit mungkin sedemikian sehingga keempat kota masih tetap terhubung satu sama lain. Masalah ini dapat dipecahkan dengan membuat upagraf yang mengandung jumlah sisi minimum dan mengandung semua simpul di dalam graf. Graf semacam itu haruslah pohon merentang.

Pohon merentang juga memainkan peranan penting dalam jaringan komputer. Jaringan komputer dapat dimodelkan sebagai sebuah graf. Simpul pada graf dapat menyatakan suatu terminal komputer (*work station*) atau suatu *router*. (*router* adalah komputer yang difungsikan untuk meneruskan data dari suatu simpul komunikasi ke simpul komunikasi lain). Jika sebuah komputer mengirim pesan (atau data) ke komputer lain (melalui *router*), maka komputer tersebut mengirimkannya ke seluruh simpul-simpul di jaringan. Setiap pesan yang sampai ke suatu *router* antara akan diteruskan ke satu atau lebih *router* lainnya. Dengan cara seperti ini, maka pesan akan sampai ke komputer penerima. Pesan yang telah sampai ke suatu *router* diharapkan tidak pernah kembali diterima oleh *router* tersebut. Tetapi karena *router-router* pada jaringan umumnya membentuk sirkuit (atau *cycle* atau *loop*), maka menerima pesan yang sama lebih dari sekali pasti terjadi. Untuk mengatasi hal ini, maka algoritma jaringan membentuk pohon merentang di dalam graf sehingga antara sepasang simpul *router* hanya ada satu lintasan tunggal dan simpul-simpul *router* tidak pernah menerima pesan yang sama lebih dari sekali. Metode penyebaran pesan (*routing*) seperti ini dinamakan *IP Multicasting*. Gambar 9.5 memperlihatkan contoh sebuah jaringan komputer dan *router-router* yang membentuk pohon merentang.



Gambar 9.5 (a) Jaringan komputer, (b) Pohon merentang *multicast*

Harus diingat bahwa pohon merentang didefinisikan hanya untuk graf terhubung, karena pohon selalu terhubung. Pada graf tak-terhubung dengan n buah simpul kita tidak dapat menemukan upagraf terhubung dengan n buah simpul. Tiap komponen dari graf tak-terhubung mempunyai satu buah pohon merentang. Dengan demikian, graf tak-terhubung dengan k komponen mempunyai **hutan merentang (spanning forest)** yang terdiri dari k buah pohon merentang.

Pada graf terhubung terdapat setidaknya satu buah pohon merentang. Sifat ini dinyatakan dengan Teorema 9.2 berikut ini.

TEOREMA 9.2. Setiap graf terhubung mempunyai paling sedikit satu buah pohon merentang.

Teorema 9.2 menyatakan bahwa graf yang tidak mengandung sirkuit adalah pohon merentang itu sendiri. Pada graf yang mengandung sirkuit, pohon merentangnya diperoleh dengan cara memutuskan sirkuit yang ada.

Sisi pada pohon merentang – disebut **cabang (branch)** – adalah sisi dari graf semula, sedangkan **tali-hubung (chord atau link)** dari pohon adalah sisi dari graf yang tidak terdapat di dalam pohon merentang. Pada graf terhubung dengan m buah sisi dan n buah simpul terdapat $n - 1$ buah cabang dan $m - n + 1$ buah tali. Himpunan tali-hubung beserta simpul yang bersisian dengannya disebut **komplemen pohon**. Untuk graf terhubung G dengan n buah simpul dan m buah sisi, kita dapat menghitung jumlah cabang dan tali-hubung dengan rumus

$$\begin{aligned} \text{jumlah cabang} &= n - 1 \\ \text{jumlah tali-hubung} &= m - n + 1 \end{aligned}$$

dan pada graf tidak terhubung dengan k komponen, m buah sisi dan n buah simpul,

$$\begin{aligned} \text{jumlah cabang} &= n - k \\ \text{jumlah tali-hubung} &= m - n + k \end{aligned}$$

Jumlah cabang pada pohon merentang dari sebuah graf G disebut **rank graf G** , dan jumlah tali-hubung pada graf G disebut **nullity graf G** . Dapat dilihat bahwa

$$\text{rank} + \text{nullity} = \text{jumlah sisi graf } G$$

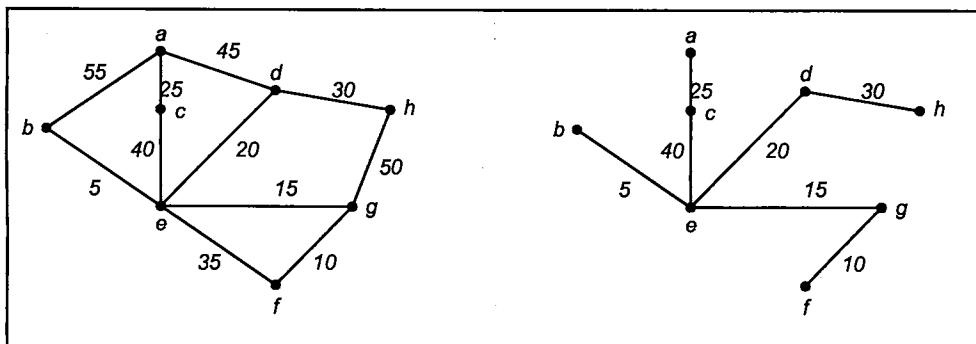
Nullity graf sering diacu sebagai **bilangan siklomatik**, atau **bilangan Betti pertama** [DEO74].

Ingatlah kembali bahwa jika kita menambahkan sebuah sisi antara dua buah simpul pada pohon maka akan terbentuk sirkuit. Tinjau kembali pohon merentang T pada graf terhubung G . Dengan menambahkan sebuah tali-hubung pada T akan terbentuk sirkuit. Sirkuit yang terbentuk dengan penambahan sebuah tali-hubung

pada pohon merentang disebut **sirkuit fundamental** (*fundamental circuit*). Berapa jumlah sirkuit fundamental pada sebuah graf? Tentu saja sebanyak jumlah tali-hubung.

Pohon Merentang Minimum

Jika G adalah graf berbobot, maka maka bobot pohon merentang T dari G didefinisikan sebagai jumlah bobot semua sisi di T . Pohon merentang yang berbeda mempunyai bobot yang berbeda pula. Di antara semua pohon merentang di G , pohon merentang yang berbobot minimum – dinamakan **pohon merentang minimum** (*minimum spanning tree*) – merupakan pohon merentang yang paling penting. Pohon merentang minimum mempunyai terapan yang luas dalam praktik. Misalkan Pemerintah akan membangun jalur rel kereta api yang menghubungkan sejumlah kota seperti yang digambarkan oleh graf pada Gambar 9.6. Membangun jalur rel kereta api biayanya mahal, karena itu pembangunan jalur ini tidak perlu menghubungkan langsung dua buah kota; tetapi cukup membangun jalur kereta seperti pohon merentang. Karena di dalam sebuah graf mungkin saja terdapat lebih dari satu pohon merentang, harus dicari pohon merentang yang mempunyai jumlah jarak terpendek, dengan kata lain harus dicari pohon merentang minimum.



Gambar 9.6 (a) Graf yang menyatakan jaringan jalur rel kereta api. Bobot pada tiap sisi menyatakan panjang rel kereta api ($\times 100$ km)

(b) Pohon merentang yang mempunyai jumlah jarak minimum

Terdapat dua buah algoritma membangun pohon merentang minimum. Yang pertama adalah algoritma Prim, dan yang kedua adalah algoritma Kruskal

Algoritma Prim

Misalkan T adalah pohon merentang yang sisi-sisinya diambil dari graf G . Algoritma Prim membentuk pohon merentang minimum langkah per langkah. Pada setiap langkah kita mengambil sisi e dari graf G yang mempunyai bobot

minimum dan bersisian dengan simpul-simpul di dalam T tetapi e tidak membentuk sirkuit di dalam T .

ALGORITMA PRIM

1. Ambil sisi dari graf G yang berbobot minimum, masukkan ke dalam T .
2. Pilih sisi e yang mempunyai bobot minimum dan bersisian dengan simpul di T , tetapi e tidak membentuk sirkuit di T . Masukkan e ke dalam T .
3. Ulangi 2 sebanyak $n - 2$ kali.

Jumlah langkah seluruhnya di dalam algoritma Prim adalah $1 + (n - 2) = n - 1$, yaitu sebanyak jumlah sisi di dalam pohon merentang dengan n buah simpul.

Dalam notasi *pseudo-code*, algoritma Prim kita tuliskan sebagai berikut:

```
procedure Prim(input G : graf, output T : pohon)
{ Membentuk pohon merentang minimum T dari graf terhubung G.
  Masukan: graf-berbobot terhubung G = (V, E), yang mana |V| = n
  Keluaran: pohon merentang minimum T = (V, E')
}

Deklarasi
e : sisi

Algoritma
T ← sisi e yang mempunyai bobot minimum di dalam E
E ← E - {e} { e sudah dipilih, jadi buang e dari E }
for i ← 1 to n - 2 do
  e ← sisi yang mempunyai bobot terkecil di dalam E dan bersisian
  dengan simpul di T
  T ← T ∪ {e} { masukkan e ke dalam T yang sudah terbentuk }
  E ← E - {e} { e sudah dipilih, jadi buang e dari E }
endfor
```

Algoritma 9.1 Algoritma Prim untuk membentuk pohon merentang minimum

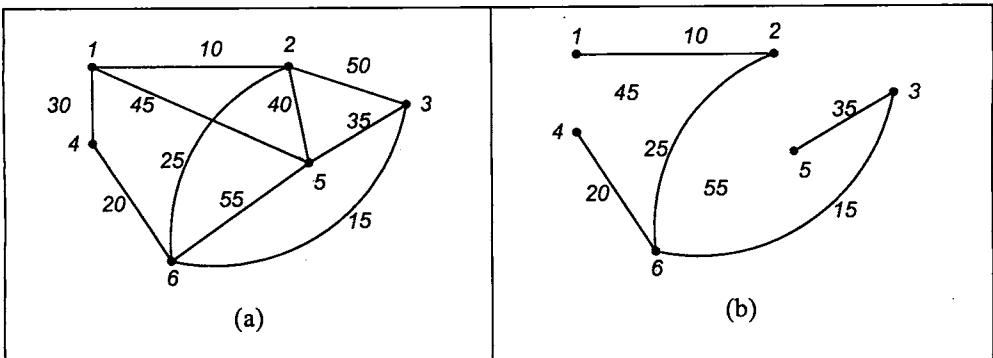
Contoh 9.2

Kita akan mencari pohon merentang minimum pada graf yang ditunjukkan pada Gambar 9.7 dengan algoritma Prim.

Penyelesaian:

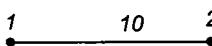
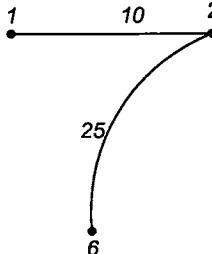
Langkah-langkah pembentukan pohon merentang minimum diperlihatkan pada Tabel 9.1. Pohon merentang minimum dari graf pada Gambar 9.7(a) adalah seperti yang ditunjukkan oleh Gambar 9.7(b). Bobot pohon merentang minimum ini adalah

$$10 + 25 + 15 + 20 + 35 = 105$$



Gambar 9.7 (a) Contoh graf untuk algoritma Prim dan Kruskal
(b) Pohon merentang minimum dari graf pada Gambar

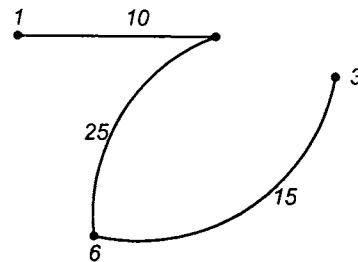
Tabel 9.1 Tabel pembentukan pohon merentang minimum dengan algoritma Prim

Langkah	Sisi	Bobot	Pohon merentang
1	(1, 2)	10	
2	(2, 6)	25	

3

(3, 6)

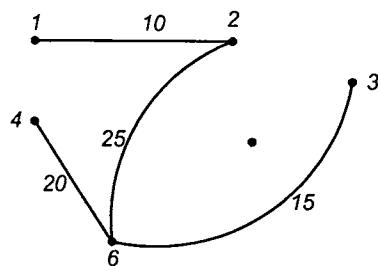
15



4

(4, 6)

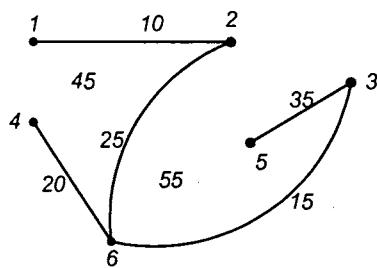
20



5

(3, 5)

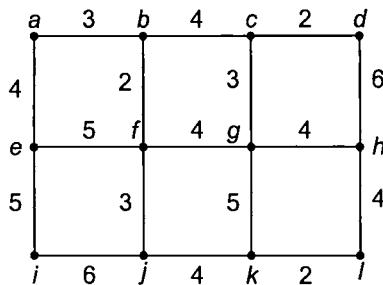
35



Perhatikanlah bahwa algoritma Prim tidak menentukan sisi mana yang dipilih jika terdapat lebih dari satu buah sisi yang berbobot sama. Satu cara untuk mengatasi hal ini adalah dengan mengurutkan sisi-sisi itu berdasarkan bobotnya dari kecil ke besar. Lagi pula, pohon merentang minimum yang dihasilkan tidak selalu unik. Graf sederhana terhubung dan berbobot dapat memiliki lebih dari satu buah pohon merentang minimum yang berbeda tetapi jumlah bobot minimumnya tetap sama. Hal ini ditunjukkan pada Contoh 9.3 berikut.

Contoh 9.3

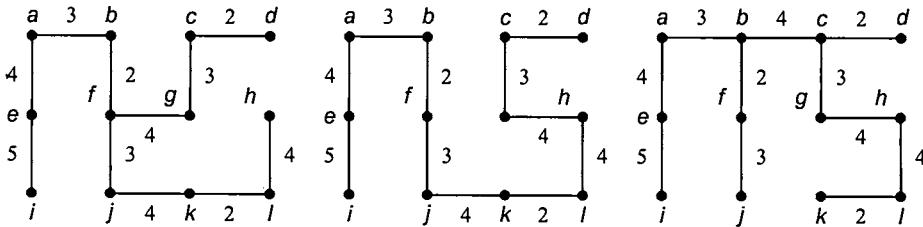
Gambarkan 3 buah pohon merentang minimum yang berbeda beserta bobotnya untuk graf pada Gambar 9.8 dengan menggunakan algoritma Prim.



Gambar 9.8 Graf untuk persoalan pada Contoh 9.3

Penyelesaian:

Graf di atas memiliki beberapa sisi yang berbobot sama, maka ada kemungkinan pohon merentang minimumnya lebih dari satu. Tiga buah di antaranya di adalah seperti di bawah ini:



Ketiga buah pohon merentang minimum di atas sama hanya bentuknya yang berbeda, namun jumlah bobot seluruh sisinya tetap sama, yaitu 36. ■

Algoritma Kruskal

Pada algoritma Kruskal, sisi-sisi di dalam graf diurut terlebih dahulu berdasarkan bobotnya dari kecil ke besar. Sisi yang dimasukkan ke dalam himpunan T adalah sisi graf G sedemikian sehingga T adalah pohon. Pada keadaan awal, sisi-sisi sudah diurut berdasarkan bobot membentuk hutan (*forest*), masing-masing pohon di hutan hanya berupa satu buah simpul. Hutan tersebut dinamakan **hutan merentang** (*spanning forest*). Sisi dari graf G ditambahkan ke T jika ia tidak membentuk siklus di T .

Perbedaan prinsip antara algoritma Prim dan Kruskal adalah: jika pada algoritma Prim sisi yang dimasukkan ke dalam T harus bersisian dengan sebuah simpul di T , maka pada algoritma Kruskal sisi yang dipilih tidak perlu bersisian dengan sebuah simpul di T asalkan penambahan sisi tersebut tidak membentuk sirkuit (siklus).

ALGORITMA KRUSKAL

(Asumsi: sisi-sisi dari graf sudah diurut menaik berdasarkan bobotnya)

1. T masih kosong
2. Pilih sisi e dengan bobot minimum yang tidak membentuk sirkuit di T . Masukkan e ke dalam T .
3. Ulangi langkah 2 sebanyak $n - 1$ kali.

Dalam notasi *pseudo-code*, algoritma Prim kita tuliskan sebagai berikut:

```
procedure Kruskal(input G : graf, output T : pohon)
{ Membentuk pohon merentang minimum T dari graf terhubung G.
  Masukan: graf-berbobot terhubung  $G = (V, E)$ , yang mana  $|V| = n$ 
  Keluaran: pohon rentang minimum  $T = (V, E')$ 
}

Deklarasi
  i, p, q, u, v : integer

Algoritma
  ( Asumsi: sisi-sisi dari graf sudah diurut menaik berdasarkan bobotnya )
  T  $\leftarrow \{\}$ 
  while jumlah sisi  $T < n - 1$  do
    e  $\leftarrow$  sisi di dalam  $E$  yang bobotnya terkecil
     $E \leftarrow E - \{e\}$       { e sudah dipilih, jadi buang e dari  $E$  }
    if e tidak membentuk siklus di  $T$  then
       $T \leftarrow T \cup \{e\}$   { masukkan e ke dalam  $T$  yang sudah terbentuk }
    endif
  endwhile
```

Algoritma 9.2 Algoritma Kruskal untuk membentuk pohon merentang minimum

Contoh 9.4

Tinjau kembali graf pada Gambar 9.7(a) di atas. Carilah pohon merentang minimumnya dengan algoritma Kruskal.

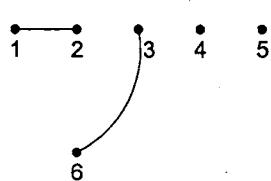
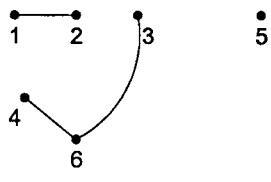
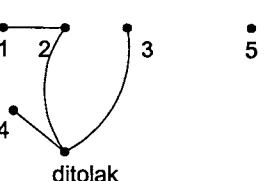
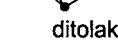
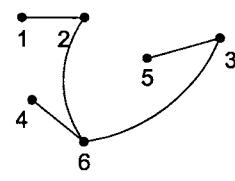
Penyelesaian:

Sisi-sisi graf diurut menaik menaik berdasarkan bobotnya:

Sisi	(1,2)	(3,6)	(4,6)	(2,6)	(1,4)	(3,5)	(2,5)	(1,5)	(2,3)	(5,6)
Bobot	10	15	20	25	30	35	40	45	50	55

Langkah-langkah pembentukan pohon merentang minimum diperlihatkan pada Tabel 9.2. Pohon merentang minimum dari graf pada Gambar 9.7(a) adalah seperti yang ditunjukkan oleh Gambar 9.7(b). Bobot pohon merentang minimum ini adalah $10 + 25 + 15 + 20 + 35 = 105$. ■

Tabel 9.2 Tabel pembentukan pohon merentang minimum dengan algoritma Kuskal

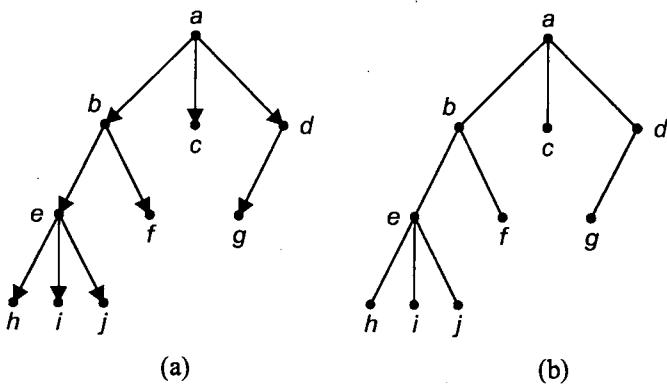
Langkah	Sisi	Bobot	Hutan merentang
0			• 1 2 3 4 5 6
1	(1, 2)	10	
2	(3, 6)	15	
3	(4, 6)	20	
4	(2, 6)	25	
5	(1, 4)	30	
6	(3, 5)	35	

9.5 Pohon Berakar

Pada kebanyakan aplikasi pohon, simpul tertentu diperlakukan sebagai akar (*root*). Sekali sebuah simpul ditetapkan sebagai akar, maka simpul-simpul lainnya dapat dicapai dari akar dengan memberi arah pada sisi-sisinya mengikutinya.

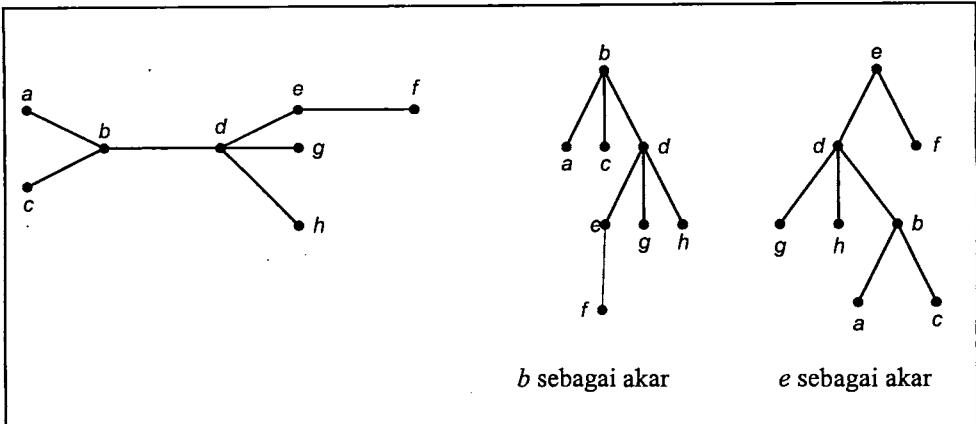
DEFINISI 9.2. Pohon yang sebuah simpulnya diperlakukan sebagai akar dan sisi-sisinya diberi arah menjauh dari akar dinamakan **pohon berakar** (*rooted tree*).

Akar mempunyai derajat-masuk sama dengan nol dan simpul-simpul lainnya berderajat-masuk sama dengan satu. Simpul yang mempunyai derajat-keluar sama dengan nol disebut **daun** atau simpul terminal. Simpul yang mempunyai derajat-keluar tidak sama dengan nol disebut simpul dalam atau simpul cabang. Setiap simpul di pohon dapat dicapai dari akar dengan sebuah lintasan tunggal (unik). Gambar 9.9(a) adalah contoh pohon berakar dengan *a* adalah simpul akarnya. Sebagai konvensi, arah sisi di dalam pohon tidak perlu digambar, karena setiap simpul di pohon harus dicapai dari akar, maka lintasan di dalam pohon berakar selalu dari “atas” ke “bawah”. Gambar 9.9(b) menunjukkan hal ini.



Gambar 9.9 (a) Pohon berakar, (b) sebagai konvensi, arah panah pada sisi dapat dibuang

Sembarang pohon tak-berakar dapat diubah menjadi pohon berakar dengan memilih sebuah simpul sebagai akar. Pemilihan simpul yang berbeda menjadi akar menghasilkan pohon berakar yang berbeda pula. Gambar 9.10 memperlihatkan dua pohon akar yang berbeda dari pengubahan sebuah pohon tak-berakar. Pohon berakar pertama memilih *b* sebagai akar sedangkan pohon akar kedua memilih *e* sebagai akar.



Gambar 9.10 (kiri) Pohon dan (kanan) dua buah pohon berakar yang dihasilkan dari pemilihan dua simpul berbeda sebagai akar

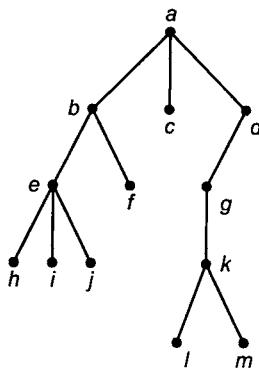
9.6 Terminologi pada Pohon Berakar

Keseluruhan sisa bab ini akan menggunakan istilah “pohon berakar” untuk membedakannya dengan “pohon” tidak berakar (*free tree*) (sebagaimana yang telah disinggung di dalam upabab 9.1).

Sebagaimana pada graf, kita akan sering menggunakan terminologi yang berhubungan dengan pohon. Di bawah ini didaftarkan beberapa terminologi yang penting pada pohon berakar. Untuk ilustrasi, pohon pada Gambar 9.12 dipakai sebagai contoh untuk menjelaskan terminologi yang dimaksudkan. Simpul-simpul pada pohon diberi label untuk mengacu simpul mana yang dimaksudkan. Kebanyakan terminologi pohon yang ditulis di bawah ini diadopsi dari terminologi botani dan silsilah keluarga.

Anak (*child* atau *children*) dan Orangtua (*parent*)

Misalkan Xx adalah sebuah simpul di dalam pohon berakar. Simpul y dikatakan **anak** simpul x jika ada sisi dari simpul x ke y . Dalam hal demikian, x disebut **orangtua (parent)** y . Pada Gambar 9.11, b, c , dan d adalah anak-anak simpul a , dan a adalah orangtua dari anak-anak itu. e dan f adalah anak-anak simpul b , dan b adalah orangtua dari e dan f . g adalah anak simpul d , dan d adalah orangtua g . Simpul h, i, j, l , dan m tidak mempunyai anak.



Gambar 9.11 Pohon berakar yang digunakan untuk menjelaskan terminologi pohon

Lintasan (*path*)

Lintasan dari simpul v_1 ke simpul v_k adalah runtunan simpul-simpul v_1, v_2, \dots, v_k sedemikian sehingga v_i adalah orangtua dari v_{i+1} untuk $1 \leq i < k$. Dari pohon pada Gambar 9.11, lintasan dari a ke j adalah a, b, e, j . **Panjang lintasan** adalah jumlah sisi yang dilalui dalam suatu lintasan, yaitu $k - 1$. Panjang lintasan dari a ke j adalah 3.

Keturunan (*descendant*) dan Leluhur (*ancestor*)

Jika terdapat lintasan dari simpul x ke simpul y di dalam pohon, maka x adalah **leluhur** dari simpul y , dan y adalah keturunan simpul x . Pada Gambar 9.11, b adalah leluhur simpul h , dan dengan demikian h adalah keturunan b .

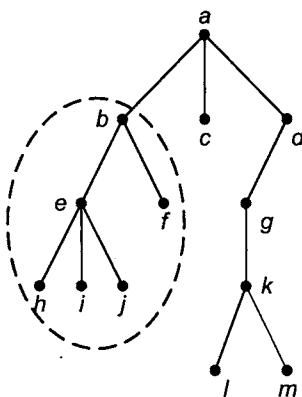
Saudara kandung (*sibling*)

Simpul yang berorangtua sama adalah **saudara kandung** satu sama lain. Pada Gambar 9.11, f adalah saudara kandung e . Tetapi, g bukan saudara kandung e , karena orangtua mereka berbeda.

Upapohon (*subtree*)

Misalkan x adalah simpul di dalam pohon T . Yang dimaksud dengan **upapohon** dengan x sebagai akarnya ialah upagraf $T' = (V', E')$ sedemikian sehingga V' mengandung x dan semua keturunannya dan E' mengandung sisi-sisi dalam semua lintasan yang berasal dari x . Sebagai contoh, $T' = (V', E')$ adalah upapohon dari pohon pada Gambar 9.12 dengan $V' = \{b, e, f, h, i, j\}$ dan $E' = \{$

$(b, e), (b, f), (e, h), (e, i), (e, j) \}$ dan b adalah simpul akarnya. Terdapat banyak upapohon di dalam pohon T . Dengan pengertian di atas, jika x adalah simpul, maka akar tiap-tiap upapohon dari x disebut anak, dan x adalah orangtua setiap akar upapohon.



Gambar 9.12 Upapohon $T' = (V, E)$ dengan b sebagai akarnya

Derajat (degree)

Ada perbedaan definisi derajat pada pohon berakar dengan definisi derajat pada graf (termasuk pohon tidak berakar). **Derajat** sebuah simpul pada pohon berakar adalah jumlah upapohon (atau jumlah anak) pada simpul tersebut. Pada Gambar 9.11, derajat a adalah 3, derajat b adalah 2, derajat d adalah satu dan derajat c adalah 0. Jadi, derajat yang dimaksudkan di sini adalah derajat-keluar.

Derajat maksimum dari semua simpul merupakan derajat pohon itu sendiri. Pohon pada Gambar 9.11 berderajat 3, karena derajat tertinggi dari seluruh simpulnya adalah 3.

Daun (leaf)

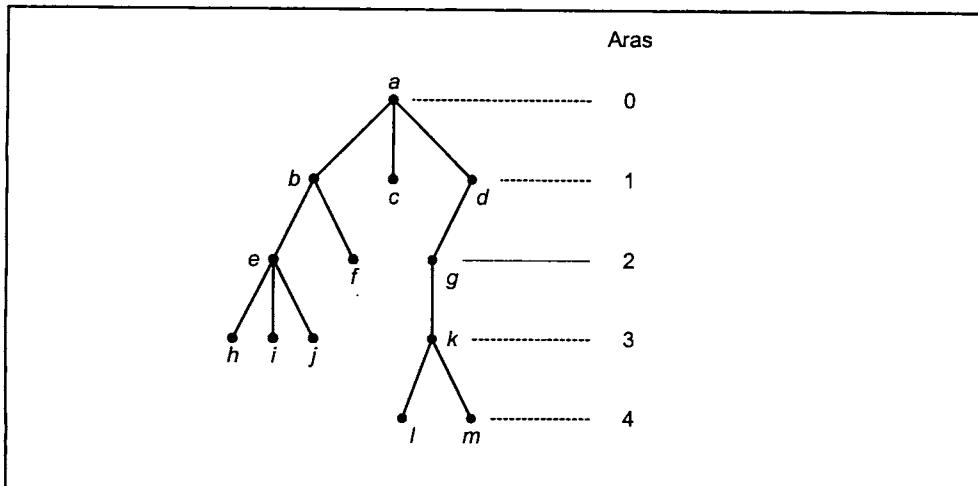
Simpul yang berderajat nol (atau tidak mempunyai anak) disebut **daun**. Simpul h , i , j , f , c , l , dan m adalah daun.

Simpul Dalam (internal nodes)

Simpul yang mempunyai anak disebut **simpul dalam**. Simpul d , e , g , dan k pada Gambar 9.11 adalah simpul dalam.

Aras (*level*) atau Tingkat

Akar mempunyai aras = 0, sedangkan aras simpul lainnya = 1 + panjang lintasan dari akar ke simpul tersebut. Beberapa literatur memulai nomor aras dari 0, literatur lainnya dari 1. Sebagai konvensi, kita memulai penomoran aras dari 0. Lihat Gambar 9.13.



Gambar 9.13 Pendefinisiaranas tiap simpul

Tinggi (*height*) atau Kedalaman (*depth*)

Aras maksimum dari suatu pohon disebut **tinggi** atau **kedalaman** pohon tersebut. Atau, dapat juga dikatakan, tinggi pohon adalah panjang maksimum lintasan dari akar ke daun. Pohon pada Gambar 9.13 mempunyai tinggi 4.

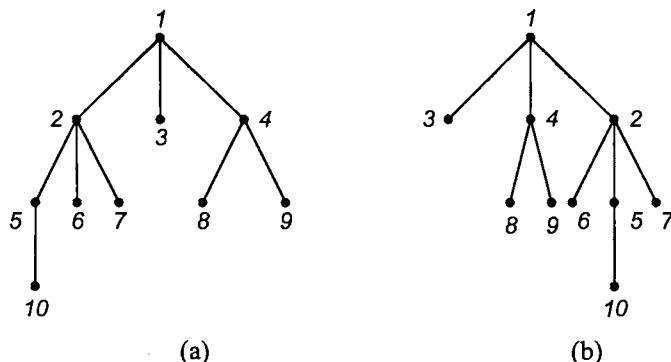
9.7 Pohon Berakar Terurut

DEFINISI 9.3. Pohon berakar yang urutan anak-anaknya penting disebut **pohon terurut** (*ordered tree*).

Pada pohon terurut, urutan anak-anak dari simpul dalam dispesifikasi dari kiri ke kanan. Sebagai contoh, dua buah pohon pada Gambar 9.14 adalah pohon berakar yang sama, tetapi sebagai pohon terurut, keduanya berbeda. Misalnya urutan anak-anak dari simpul 1 pada Gambar 9.14(a) adalah 2, 3, 4, sedangkan urutan anak-anak dari simpul 1 pada Gambar 9.14(b) adalah 3, 4, 2.

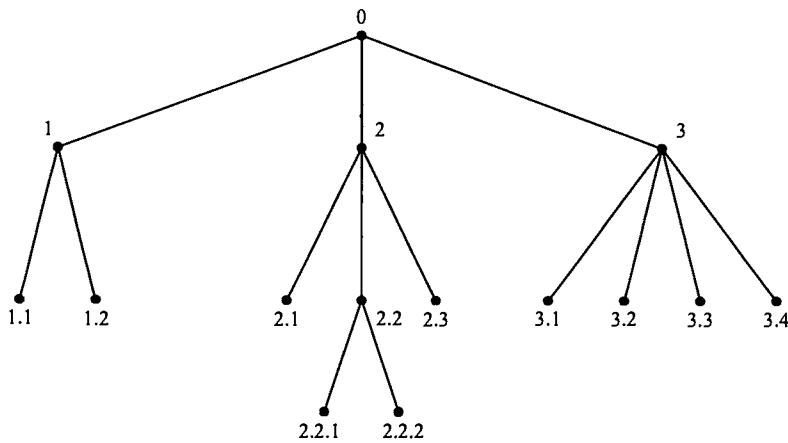
Perbedaan ini menjadi penting bila kita merepresentasikan pohon di dalam komputer, karena penelusuran dua buah pohon terurut yang berbeda akan menghasilkan urutan simpul yang berbeda pula. Jika pohon berakar terurut pada

simpul x mempunyai p buah upapohon, kita akan mengacunya sebagai upapohon pertama, upapohon kedua, ..., upapohon ke- p .



Gambar 9.14 Dua buah pohon terurut yang berbeda

Sistem yang universal dalam pengalamatan simpul-simpul pada pohon terurut adalah dengan memberi nomor setiap simpulnya seperti penomoran bab (beserta upababnya) di dalam sebuah buku. Simpul akar diberi nomor 0, simpul lain yang segera mengikuti akar diberi nomor 1, 2, 3, dan seterusnya. Anak-anak simpul 1 diberi nomor 1.1, 1.2, ...; anak-anak simpul 2 diberi nomor 2.1, 2.2, ...; demikian seterusnya (Gambar 9.15). Semua penomoran dimulai dengan aturan dari kiri ke kanan.



Gambar 9.15 Sistem pengalamatan universal pada pohon terurut.

9.8 Pohon *m*-ary

DEFINISI 9.5. Pohon berakar yang setiap simpul cabangnya mempunyai paling banyak n buah anak disebut **pohon *m*-ary**.

Jika $m = 2$, pohon disebut **pohon biner** (*binary tree*). Pohon pada Gambar 9.15 adalah pohon *3-ary*. Pohon *m*-ary dikatakan **pohon penuh** (*full*) atau **pohon teratur** jika setiap simpul cabangnya mempunyai tepat m buah anak.

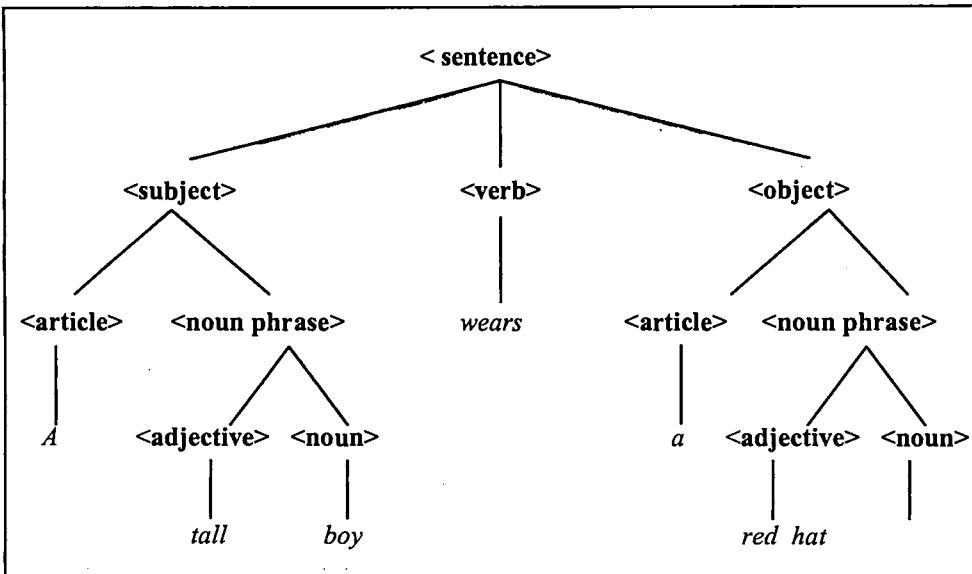
Pohon *m*-ary banyak digunakan di berbagai bidang ilmu maupun dalam kehidupan sehari-hari. Dalam terapannya, pohon *m*-ary digunakan sebagai model yang merepresentasikan suatu struktur. Dua contoh penggunaan pohon *m*-ary dikemukakan di bawah ini, yaitu penurunan kalimat (dalam bidang bahasa) dan direktori arsip di dalam komputer. Contoh penggunaan pohon *m*-ary lainnya adalah struktur organisasi, silsilah keluarga (dalam bidang genetika), struktur bab atau daftar isi di dalam buku, bagan pertandingan antara beberapa tim sepakbola, dan sebagainya.

Contoh 9.5

Pohon *m*-ary digunakan untuk merepresentasikan struktur kalimat dalam bahasa alami (*natural language*) maupun dalam bahasa pemrograman. Pohon penurunan kalimat itu disebut pohon *parsing* (*parse tree*). Gambar 9.16 memperlihatkan cara penurunan kalimat dalam Bahasa Inggris yang berbunyi

A tall boy wears a red hat

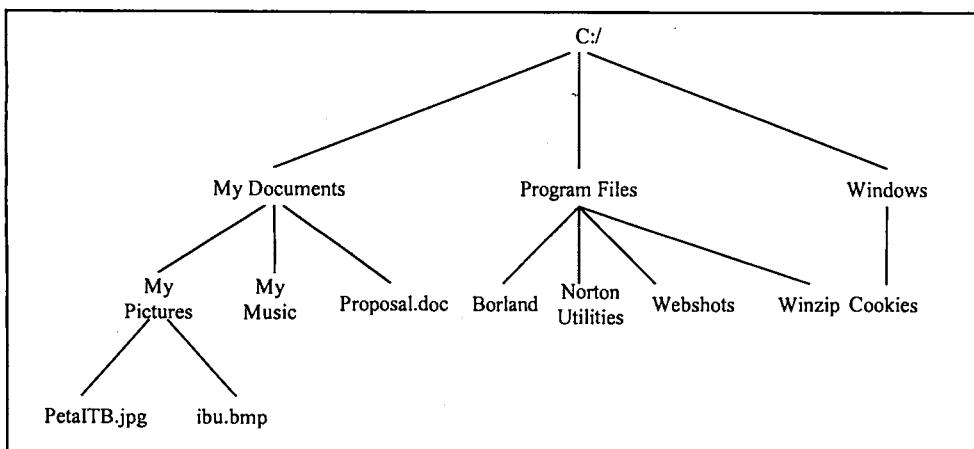
Akar menyatakan kalimat (*sentence*), daun menyatakan setiap kata-kata yang menyusun kalimat, sedangkan simpul dalam menyatakan cara pembagian kalimat menjadi unsur-unsur pembentuknya. Sebuah kalimat (*sentence*) dalam Bahasa Inggris disusun oleh *subject*, *verb* dan *object*. *Subject* dapat terdiri dari sebuah *article* dan *noun phrase*. Sebuah *noun phrase* dapat terdiri atas *adjective* dan *noun*. Begitu juga *object* dapat terdiri atas sebuah *article* dan *noun phrase*. ■



Gambar 9.16 Pohon parsing dari kalimat *A tall boy wears a red hat*

Contoh 9.6

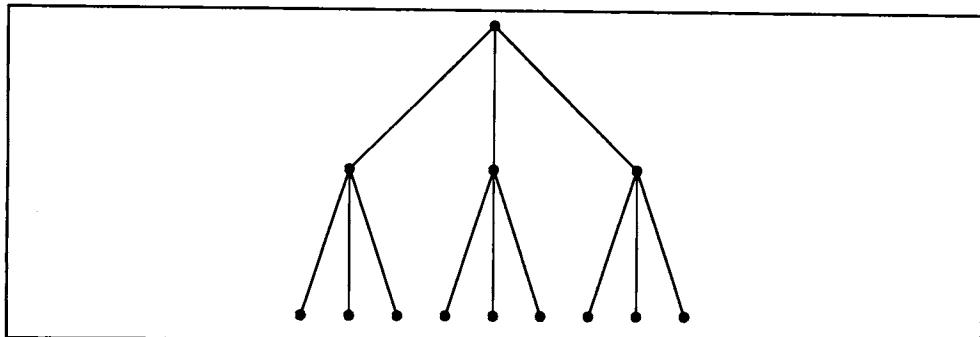
Dalam sistem pengarsipan komputer (*computer file system*), arsip-arsip di dalam media penyimpanan sekunder (seperti *floppy disk*, *compact disk*, *hard disk*), disusun dalam bentuk direktori. Suatu direktori dapat mengandung beberapa upa-direktori (*sub-directory*). Struktur direktori dimodelkan dalam pohon *m-ary*. Di sini akar menyatakan sistem arsip keseluruhan, simpul dalam menyatakan upa-direktori, dan daun menyatakan arsip atau direktori kosong. Gambar 9.17 memperlihatkan struktur direktori dalam *platform* sistem operasi *Windows*. ■



Gambar 9.17 Struktur direktori arsip di dalam sistem operasi *Windows*

Jumlah Daun pada Pohon m -ary Penuh

Pohon m -ary penuh adalah pohon yang setiap simpulnya tepat mempunyai m buah anak. Pada pohon m -ary penuh dengan tinggi h , jumlah daun adalah m^h . Perhatikan bahwa jika T bukan pohon m -ary penuh, maka jumlah daun $\leq m^h$. Gambar 9.18 adalah pohon 3-ary penuh dengan jumlah daun = $3^2 = 9$.



Gambar 9.18 Pohon 3-ary penuh dengan jumlah daun = $3^2 = 9$.

Jumlah Seluruh Simpul pada Pohon m -ary Penuh

Pada pohon m -ary penuh dengan tinggi h ,

$$\begin{aligned} \text{aras 0} &\rightarrow \text{jumlah simpul} = m^0 = 1 \\ \text{aras 1} &\rightarrow \text{jumlah simpul} = m^1 \\ \text{aras 2} &\rightarrow \text{jumlah simpul} = m^2 \\ \dots & \\ \text{aras } h &\rightarrow \text{jumlah simpul} = m^h \end{aligned}$$

maka jumlah seluruh simpul adalah

$$S = m^0 + m^1 + m^2 + \dots + m^h = \frac{m^{h+1} - 1}{m - 1} \quad (9.1)$$

Perhatikan bahwa jika T bukan pohon m -ary penuh, maka $S \leq \frac{m^{h+1} - 1}{m - 1}$.

Hubungan Jumlah Daun dan Simpul Dalam pada Pohon m -ary Penuh

Contoh 9.7

[LIU85] Misalkan sebuah turnamen tenis diikuti oleh t orang tim dengan sistem pertandingan gugur (pemain yang kalah langsung tersingkir, pemain yang menang melawan pemenang pertandingan lainnya). Berapa banyak pertandingan yang terjadi?

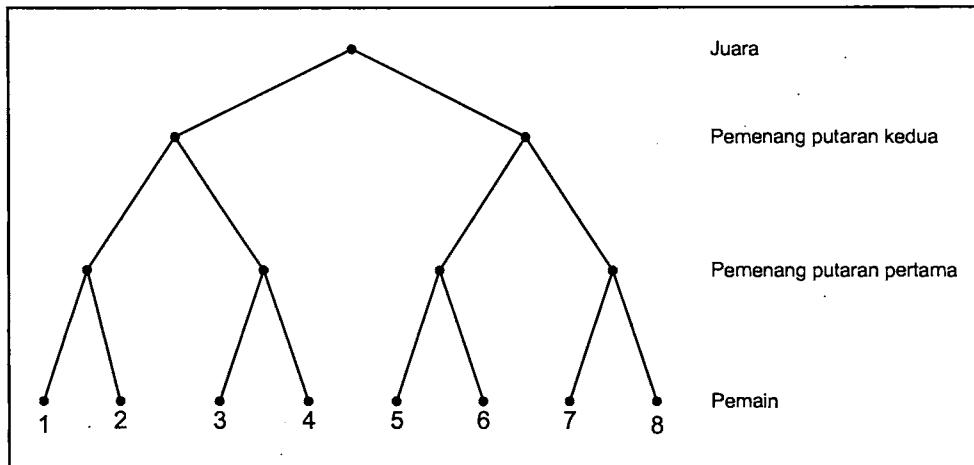
Penyelesaian:

Persoalan ini dapat dimodelkan dengan pohon biner penuh. Daun menyatakan pemain, simpul dalam (termasuk akar) menyatakan para pemenang pertandingan (jumlah seluruh simpul dalam berarti juga total banyak pertandingan yang dilakukan sampai mencapai juara).

Misalkan i adalah banyaknya simpul dalam dan t adalah banyaknya simpul daun di dalam pohon biner penuh. Karena di dalam sistem gugur setiap pertandingan menggugurkan seorang pemain, dan pada akhir pertandingan yang dimainkan semua peserta telah gugur kecuali sang juara, maka banyaknya pertandingan yang terjadi satu lebih sedikit daripada jumlah pemain. Jadi, diperoleh hubungan:

$$i = t - 1 \quad (9.2)$$

Gambar 9.19 adalah pohon biner teratur yang memodelkan sistem gugur untuk $t = 8$ orang pemain. Banyaknya pertandingan yang terjadi adalah $i = 8 - 1 = 7$ kali. ■



Gambar 9.19 Pohon pertandingan turnamen tenis dengan sistem gugur

Persamaan di atas dapat dirampatkan untuk pohon m -ary penuh. Misalkan setiap pertandingan diikuti oleh m orang pemain dan hanya satu yang keluar sebagai pemenang. Jadi, setiap pertandingan menggugurkan $m - 1$ orang pemain. Kita peroleh hubungan:

$$(m - 1)i = t - 1 \quad (9.3)$$

Contoh 9.6

[LIU85] Misalkan kita akan menyambungkan 19 buah lampu pada satu stop kontak dengan menggunakan sejumlah kabel ekstensi yang masing-masing mempunyai empat

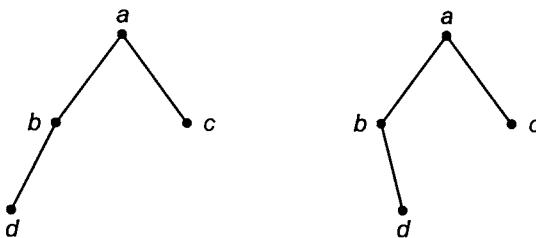
outlet. Karena penyambungan semacam itu merupakan pohon 4-ary dengan stop kontak sebagai akar pohon, maka

$$(4 - 1) i = 19 - 1$$

diperoleh $i = 6$. Jadi, dibutuhkan enam buah kabel ekstensi. ■

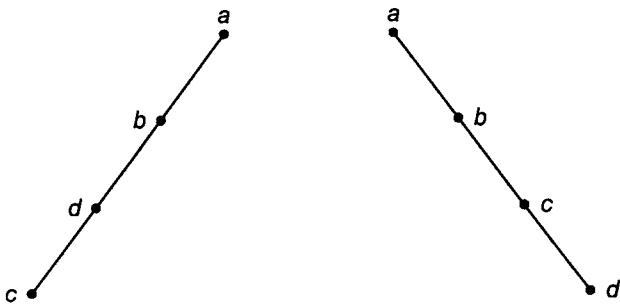
9.9 Pohon Biner

Pohon m -ary yang paling penting adalah pohon biner (*binary tree*). Pohon biner merupakan pohon m -ary jika $m = 2$. Pohon biner adalah pohon yang setiap simpul cabangnya mempunyai paling banyak dua buah anak. Alih-alih menyebutnya anak pertama dan anak kedua dari suatu simpul dalam, kita menyebutnya **anak kiri** (*left child*) dan **anak kanan** (*right child*). Pohon yang akarnya adalah anak kiri disebut **upapohon kiri** (*left subtree*), sedangkan pohon yang akarnya adalah anak kanan disebut **upapohon kanan** (*right subtree*). Karena adanya perbedaan anak/upapohon kiri dan anak/upapohon kanan, maka pohon biner adalah pohon terurut. Dua buah pohon pada Gambar 9.20 adalah dua buah pohon biner berbeda.



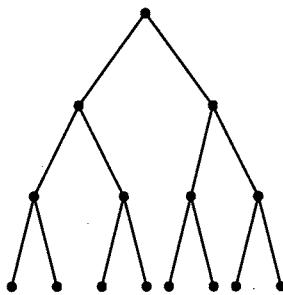
Gambar 9.20 Dua buah pohon biner yang berbeda

Pohon yang semua simpulnya terletak di bagian kiri saja atau di bagian kanan saja disebut **pohon condong** (*skewed tree*). Pohon yang condong ke kiri disebut **pohon condong-kiri** (*skew left*), pohon yang condong ke kanan disebut **pohon condong-kanan** (*skew right*). Lihat Gambar 9.21.



Gambar 9.21 (a) Pohon condong-kiri, dan (b) pohon condong kanan

Pohon biner penuh (*full binary tree*) adalah pohon biner yang setiap simpulnya mempunyai tepat dua buah anak, kiri dan kanan, kecuali simpul pada aras bawah (Gambar 9.22).



Gambar 9.22 Pohon biner penuh

Pohon biner penuh dengan tinggi h memiliki jumlah daun sebanyak 2^h , sedangkan jumlah seluruh simpulnya adalah:

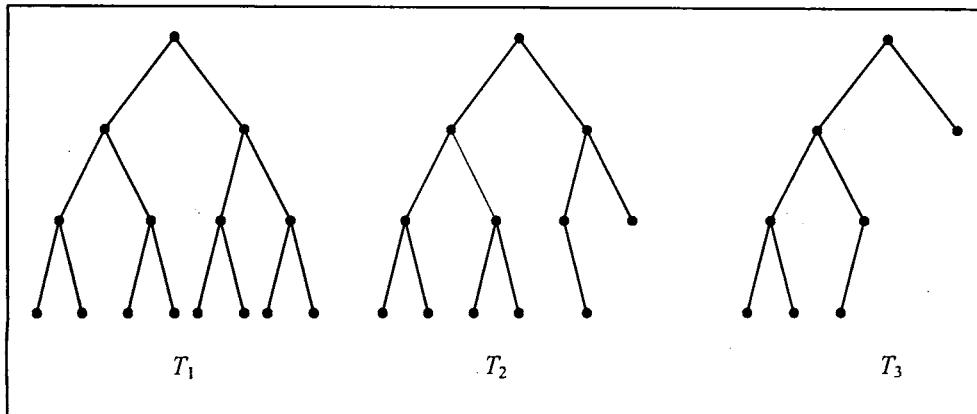
$$S = 2^0 + 2^1 + 2^2 + \dots + 2^h = 2^{h+1} - 1 \quad (9.4)$$

Pohon Biner Seimbang

Pohon biner seimbang (*balanced binary tree*) adalah pohon biner yang perbedaan tinggi antara upapohon kiri dan upapohon maksimal 1. Pada pohon biner seimbang dengan tinggi h , semua daun berada pada aras h atau $h - 1$. Untuk membuat pohon seimbang, tinggi pohon secara keseluruhan harus dibuat

seminimal mungkin. Untuk memperoleh tinggi minimum, setiap aras harus mengandung jumlah simpul sebanyak mungkin. Hal ini dapat dibuat dengan menyebarkan setengah dari jumlah simpul di upapohon kiri dan setengah dari jumlah simpul di upapohon kanan.

Pohon T_1 dan T_2 pada Gambar 9.23 adalah pohon seimbang, sedangkan T_3 bukan pohon seimbang karena perbedaan upapohon kiri dan upapohon kanan tidak maksimal satu (periksa!).



Gambar 9.23 T_1 dan T_2 adalah pohon seimbang, sedangkan T_3 bukan pohon seimbang.

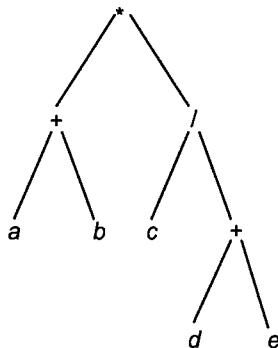
Pohon biner merupakan struktur yang penting dalam ilmu komputer. Terapan pohon biner di dalam ilmu komputer sangat banyak, di antaranya yang disebutkan di sini adalah pohon ekspresi (*expression tree*), pohon keputusan (*decision tree*), kode prefiks (*prefix code*), kode Huffman (*Huffman code*), dan pohon pencarian biner (*binary search tree*). Masing-masing terapan pohon biner di atas diuraikan pada masing-masing upabab di bawah ini.

9.10 Pohon Ekspresi

1. Pohon Ekspresi

Pohon ekspresi ialah pohon biner dengan daun menyatakan *operand* dan simpul dalam (termasuk akar) menyatakan *operator*. Perhatikan bahwa tanda kurung tidak lagi diperlukan bila suatu ekspresi aritmetik direpresentasikan sebagai pohon biner.

Sebagai contoh, ekspresi $(a + b)*(c/(d + e))$ dinyatakan dalam pohon biner pada Gambar 9.24. Daun menyatakan *operand* a , b , c , d , dan e , sedangkan simpul dalam –termasuk akar– menyatakan *operator* $+$, $*$, dan $/$.



Gambar 9.24 Pohon ekspresi dari $(a + b)^*(c/(d + e))$

Pohon ekspresi digunakan oleh *compiler* bahasa tingkat tinggi untuk mengevaluasi ekspresi yang ditulis dalam notasi *infix*, *prefix (polish notation)*, dan *prefix (inverse Polish notation)*.

Dalam notasi *infix*, operator berada di antara dua buah *operand*, pada notasi *prefix*, operator mendahului dua buah *operand*-nya, dan pada notasi *postfix* kedua *operand* mendahului operatornya.

Ekspresi $(a + b)^*(c/(d + e))$ adalah dalam bentuk *infix*, sedangkan ekspresi *prefix*-nya adalah

$$* + a b / c + d e$$

dan ekspresi *postfix*-nya adalah

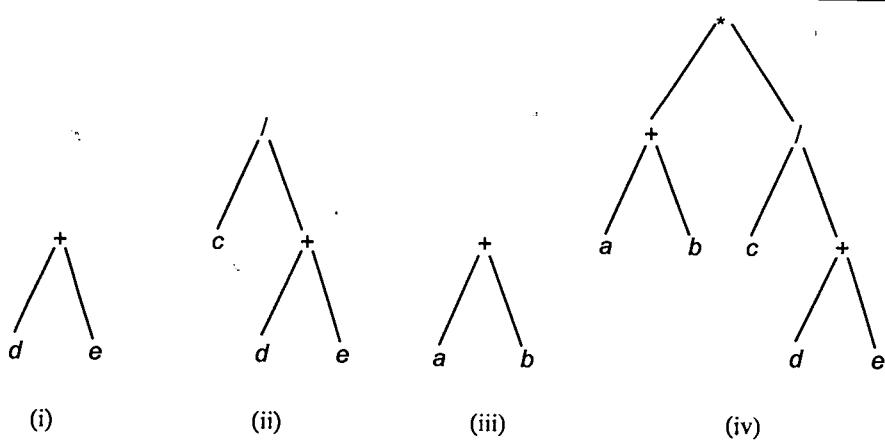
$$a b + c d e + / *$$

Contoh 9.7

Gambarkan pembentukan pohon ekspresi dari ekspresi $(a + b)^*(c/(d + e))$ di atas.

Penyelesaian:

Pohon ekspresi dari notasi *infix* dibangun dari bawah ke atas dengan memperhatikan urutan prioritas penggerjaan operator. Operator $/$ dan $*$ mempunyai prioritas lebih tinggi daripada operator $+$ dan $-$. Mula-mula dibentuk upapohon untuk $(d + e)$, kemudian upapohon untuk $c/(d+e)$, upapohon untuk $(a + b)$ dan akhirnya penggabungan upapohon $(a + b)$ dengan upapohon $d/(d+e)$. Pohon ekspresi diperlihatkan pada Gambar 9.25. ■



Gambar 9.25 Pembentukan pohon ekspresi dari $(a+b)*(d(d+e))$

Pembentukan Pohon Ekspresi dari Notasi Postfix

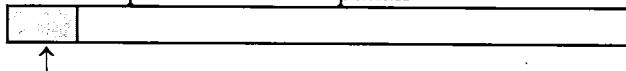
Jika diberikan ekspresi dalam notasi *postfix*, kita dapat membangun pohon ekspresinya dengan algoritma di bawah ini. Untuk itu, kita membutuhkan sebuah tabel dan sebuah tumpukan (*stack*).

- Setiap elemen (*operand* dan operator) dari notasi *postfix* yang panjangnya n disimpan di dalam tabel sebagai elemen P_1, P_2, \dots, P_n .

1	2	3	4	5	6	7	8	$n = 9$
a	b	+	c	d	e	+	/	*

- Tumpukan S menyimpan *pointer* ke simpul pohon biner (bayangkanlah tumpukan tumbuh dari “kiri” ke “kanan”).

→ arah pertumbuhan tumpukan



```

procedure BangunPohonEkspresiDariPostfix(input  $P_1, P_2, \dots, P_n$  : elemen
                                            postfix, output  $T$  : pohon)
{ Membangun pohon ekspresi dari notasi postfix
  Masukan: notasi postfix, setiap elemennya di simpan di dalam tabel  $P$ 
  Keluaran: pohon ekspresi  $T$ 
}
Deklarasi
  i : integer
  S : stack
  T1, T2 : pohon

Algoritma
  for  $i \leftarrow 1$  to  $n$  do
    if  $P_i = \text{operand}$  then
      Buat (create) satu buah simpul untuk  $P_i$ 
      Masukkan (push) pointer-nya ke dalam tumpukan  $S$ 
    else {  $P_i = \text{operator}$  }
      Ambil (pop) pointer dua upapohon  $T_1$  dan  $T_2$  dari puncak tumpukan  $S$ 
      Buat pohon  $T$  yang akarnya adalah operator dan upapohon kiri
      dan upapohon kanannya masing-masing  $T_1$  dan  $T_2$ 
    endif
  endfor

```

Algoritma 9.3 Pembentukan pohon ekspresi dari notasi *postfix*

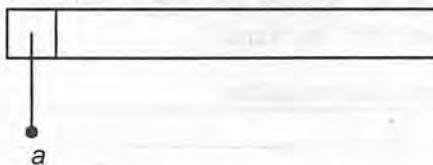
Algoritma pembentukan pohon ekspresi dari notasi *prefix* relatif sukar dipahami sehingga tidak dibicarakan di dalam bab ini.

Contoh 9.8

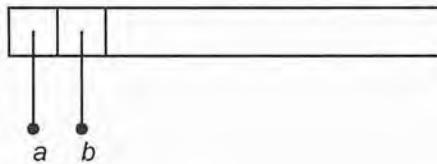
Terapkan algoritma *BangunPohonEkspresiDariPostfix* untuk membangun pohon ekspresi dari notasi *postfix* $a\ b\ +\ c\ d\ e\ +\ /*$

Penyelesaian:

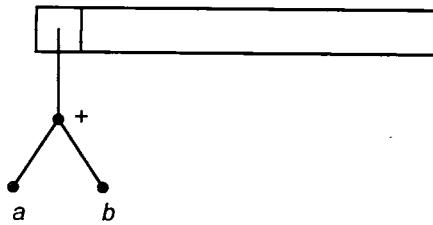
- Mulai dari elemen *postfix* pertama, P_1 . Karena $P_1 = 'a' = \text{operand}$, buat simpul untuk P_1 , *push* pointer-nya ke dalam tumpukan S .



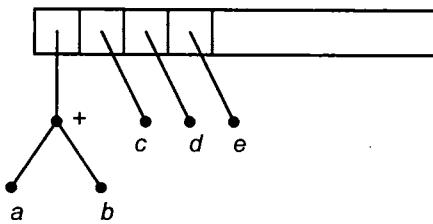
- Baca P_2 . Karena $P_2 = 'b' = \text{operand}$, buat simpul untuk P_2 , *push* pointer-nya ke tumpukan S .



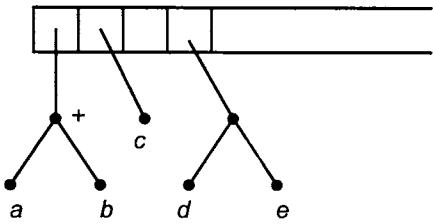
(iii) Baca P_3 . Karena $P_3 = '+' = operator$, buat pohon T dengan ‘ a ’ dan ‘ b ’ sebagai anak.



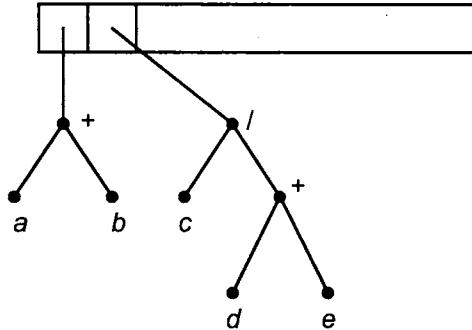
(iv) Baca P_4, P_5, P_6 . Karena $P_4, P_5, P_6 = operand$, buat simpul untuk P_4, P_5 , dan P_6 . Push pointer-nya ke dalam tumpukan S.



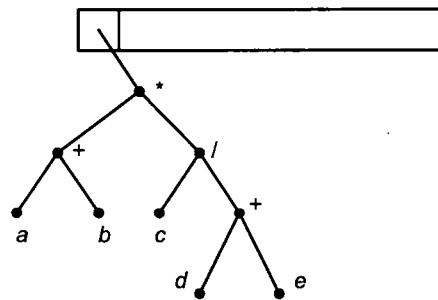
(v) Baca P_7 . Karena $P_7 = '+' = operator$, buat pohon T dengan ‘ d ’ dan ‘ e ’ sebagai anak.



(vi) Baca P_8 . Karena $P_8 = '/' = operator$, buat pohon T dengan ‘ c ’ dan ‘ $+$ ’ sebagai anak.



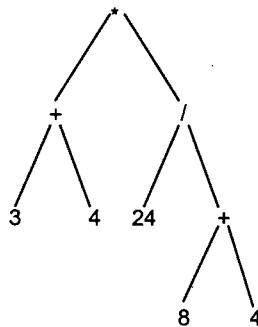
(vii) Baca P_9 . Karena $P_9 = '*' = \text{operator}$, buat pohon T dengan ‘+’ dan ‘*’ sebagai anak.



Karena semua elemen tabel P sudah habis dibaca, maka tumpukan S akan berisi *pointer* yang menunjuk ke akar pohon ekspresi. ■

Contoh 9.9

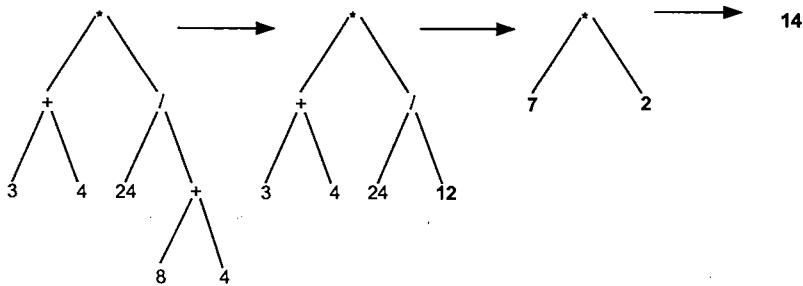
Evaluasi pohon ekspresi berikut:



Penyelesaian:

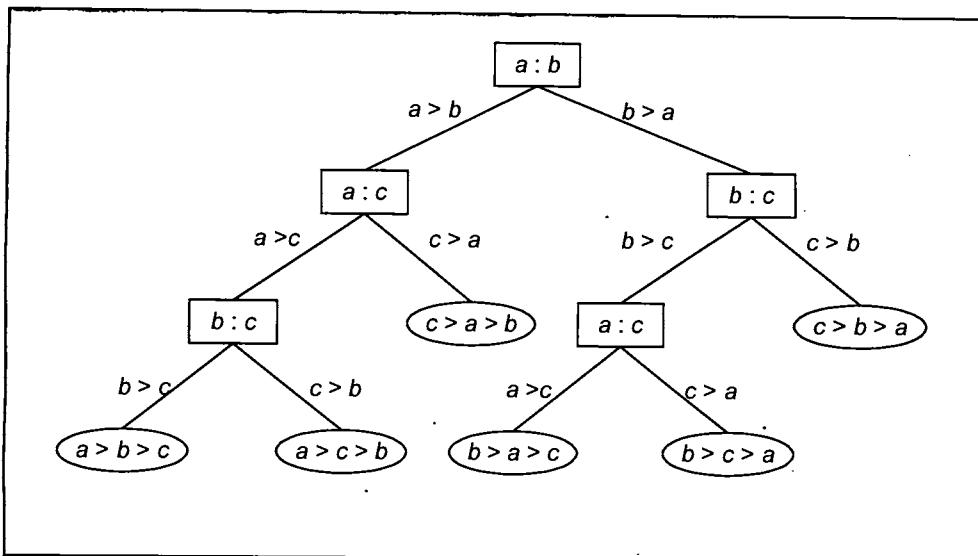
Pohon ekspresi dievaluasi mulai dari bawah ke atas. Dua buah daun (yang merepresentasikan *operand*) diperasikan dengan operator yang menjadi orangtuanya. Nilai evaluasi sementara (dicetak tebal) disimpan pada simpul orangtua tadi, dan kedua *operand* yang dioperasikan dihapus. Pada akhir evaluasi, simpul akar merepresentasikan nilai ekspresi keseluruhan (dalam hal ini 14).

Tahapan evaluasi pohon ekspresi diperlihatkan pada gambar berikut:



9.11 Pohon Keputusan

Pohon keputusan digunakan untuk memodelkan persoalan yang terdiri dari serangkaian keputusan yang mengarah ke solusi. Tiap simpul dalam menyatakan keputusan, sedangkan daun menyatakan solusi. Sebagai contoh, kita ingin mengurutkan tiga buah bilangan, a , b , dan c . Pohon keputusan untuk persoalan ini ditunjukkan pada Gambar 9.26.



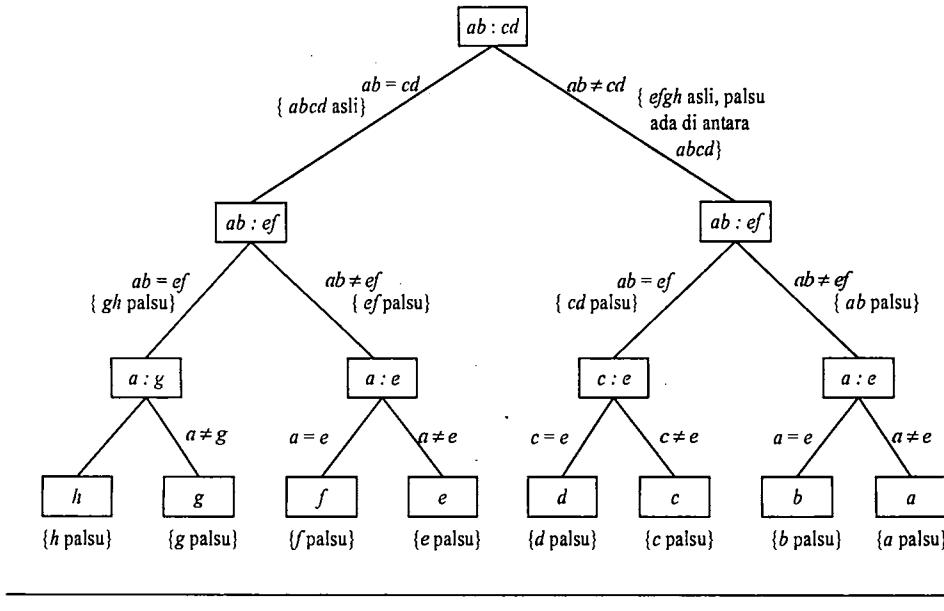
Gambar 9.26 Pohon keputusan untuk mengurutkan 3 buah elemen

Contoh 9.10

Diketahui 8 buah koin uang logam. Satu dari delapan koin itu ternyata palsu. Koin yang palsu mungkin lebih ringan atau lebih berat daripada koin yang asli. Misalkan tersedia sebuah timbangan neraca yang sangat teliti. Buatlah pohon keputusan untuk mencari uang palsu dengan cara menimbang paling banyak hanya 3 kali saja.

Penyelesaian:

Misalkan 8 koin itu dinamai a , b , c , d , e , f , g , h . Pohon keputusan untuk mencari koin yang palsu ditunjukkan di bawah ini. Daun menyatakan koin yang palsu.



9.12 Kode Awalan

Kode awalan (*prefix code*) adalah himpunan kode, misalnya kode biner, sedemikian sehingga tidak ada anggota kumpulan yang merupakan awalan dari anggota yang lain. Contohnya, himpunan

$$\{ 000, 001, 01, 10, 11 \}$$

adalah kode awalan, tetapi

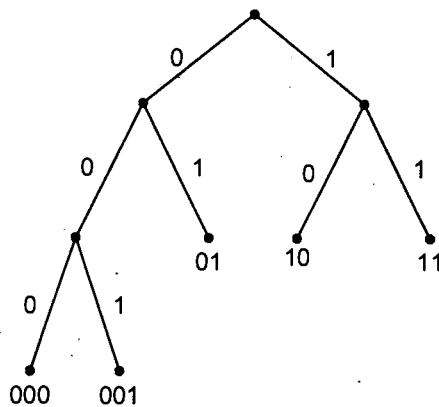
$$\{ 1, 00, 01, 000, 0001 \}$$

bukan kode awalan, sebab 00 adalah prefiks dari 0001.

Kode awalan mempunyai pohon biner yang bersesuaian. Sisi diberi label 0 atau 1. Pelabelan sisi harus taat-asas, yakni semua sisi kiri dilabeli 0 saja (atau 1 saja), sedangkan sisi kanan dilabeli 1 saja (atau 0 saja). Barisan sisi-sisi yang dilalui oleh lintasan dari akar ke daun menyatakan kode awalan. Kode awalan ini ditulis pada daun. Contoh himpunan yang pertama mempunyai pohon biner pada Gambar 9.27.

Kegunaan kode awalan adalah untuk mengirim pesan pada komunikasi data. Setiap karakter di dalam pesan direpresentasikan dengan barisan angka 0 dan 1. Untuk mengirim pesan, kita cukup mengirimkan *string* angka 0 dan 1 yang merepresentasikan karakter di dalam pesan. Oleh pihak penerima, *string* angka 0

dan 1 ini dikembalikan lagi ke karakter penyusun pesan semula. Agar tidak timbul ambigu dalam mengkonversikan kembali *string* 0 dan 1 menjadi karakter semula, maka setiap karakter tidak boleh mempunyai kode yang merupakan awalan bagi kode yang lain.



Gambar 9.27 Pohon biner dari kode prefiks { 000, 001, 01, 10, 11}

Terapan lain dari kode awalan adalah untuk pembentukan kode Huffman dalam pemampatan data (*data compression*). Kode Huffman dijelaskan di bawah ini.

9.13 Kode Huffman

Dalam komunikasi data, pesan (*message*) yang dikirim seringkali ukurannya sangat besar sehingga waktu pengirimannya lama. Begitu juga dalam penyimpanan data, arsip (*file*) yang berukuran besar memakan ruang penyimpanan yang besar. Kedua masalah ini dapat diatasi dengan mengkodekan pesan atau isi arsip sesingkat mungkin, sehingga waktu pengiriman pesan juga relatif cepat, dan ruang penyimpanan yang dibutuhkan juga sedikit. Cara pengkodean seperti ini disebut pemampatan (*compression*) data. Pemampatan data dilakukan dengan mengkodekan setiap karakter di dalam pesan atau di dalam arsip dikodekan dengan kode yang lebih pendek. Sistem kode yang banyak digunakan adalah kode ASCII. Dengan kode ASCII, setiap karakter dikodekan dalam 8 bit biner. Tabel 9.3 berikut adalah contoh kode ASCII untuk beberapa karakter:

Tabel 9.3 Kode ASCII

Simbol	Kode ASCII
A	01000001
B	01000010
C	01000011
D	01000100

Dengan mengikuti ketentuan pengkodean di atas, *string* ‘ABACCD’ direpresentasikan menjadi rangkaian bit:

010000010100000100100000101000001101000001101000100010000001

Jadi, dengan sistem pengkodean ASCII, representasi 7 huruf membutuhkan $7 \times 8 = 56$ bit (*7 byte*). Untuk meminimumkan jumlah bit yang dibutuhkan, panjang kode untuk setiap karakter sedapat mungkin diperpendek, terutama untuk karakter yang kekerapan (*frequency*) kemunculannya besar. Pemikiran seperti inilah yang mendasari munculnya kode Huffman. Misalnya pada pesan ‘ABACCD’, kekerapan kemunculan A adalah 3, kekerapan B adalah 1, kekerapan C adalah 2, dan kekerapan D adalah 1, sehingga dapat dibuat Tabel 9.4 berikut:

Tabel 9.4 Tabel kekerapan dan kode Huffman untuk *string* ‘ABACCD’

Simbol	Kekerapan	Peluang	Kode Huffman
A	3	3/7	0
B	1	1/7	110
C	2	2/7	10
D	1	1/7	111

Dengan menggunakan kode Huffman di dalam Tabel 9.4, pesan “ABACCD” direpresentasikan menjadi rangkaian bit:

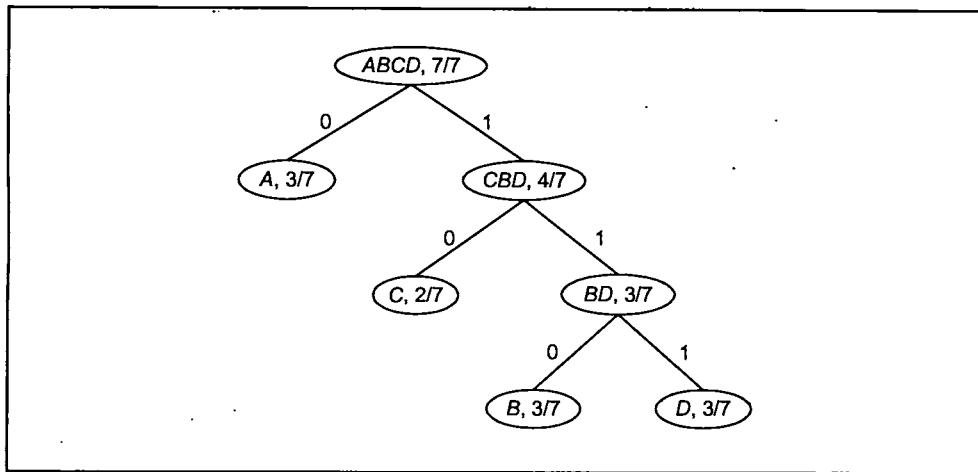
0110010101110

Jadi, dengan menggunakan kode Huffman ini, jumlah bit yang dibutuhkan untuk *string* ‘ABACCD’ hanya 13 bit. Simbol-simbol yang sering muncul di dalam *string* direpresentasikan dengan kode yang lebih pendek daripada kode untuk simbol yang jarang muncul. Kode untuk setiap simbol tidak boleh merupakan awalan dari kode yang lain sebab akan menimbulkan keraguan (*ambigu*) dalam proses pemulihannya (*decoding* - yaitu mengubah kembali kode biner ke simbol asalnya). Oleh karena itu, kode Huffman adalah kode prefiks.

Bagaimana cara mendapatkan kode Huffman?

Untuk mendapatkan kode Huffman, mula-mula kita harus menghitung dulu kekerapan kemunculan tiap simbol di dalam teks. Cara pembentukan kode Huffman adalah dengan membentuk pohon biner, yang dinamakan pohon Huffman, sebagai berikut:

- Pilih dua simbol dengan peluang (*probability*) paling kecil (pada contoh di atas simbol *B* dan *D*). Kedua simbol tadi dikombinasikan sebagai simpul orangtua dari simbol *B* dan *D* sehingga menjadi simbol *BD* dengan peluang $1/7 + 1/7 = 2/7$, yaitu jumlah peluang kedua anaknya. Simbol baru ini diperlakukan sebagai simpul baru dan diperhitungkan dalam mencari simbol selanjutnya yang memiliki peluang paling kecil.
- Selanjutnya, pilih dua simbol berikutnya, termasuk simbol baru, yang mempunyai peluang terkecil. Pada contoh ini, dua simbol tersebut adalah *C* (peluang = $2/7$) dan *BD* (peluang = $2/7$). Lakukan hal yang sama seperti langkah sebelumnya sehingga dihasilkan simbol baru *CBD* dengan kekerapan $2/7 + 2/7 = 4/7$.
- Prosedur yang sama dilakukan pada dua simbol berikutnya yang mempunyai peluang terkecil, yaitu *A* (peluang = $3/7$) dan *CBD* (peluang = $4/7$) sehingga menghasilkan simpul *ACBD*, yang merupakan akar pohon Huffman dengan peluang $3/7 + 4/7 = 7/7$.



Gambar 9.28 Pohon Huffman untuk pesan 'ABACCCDA'

Daun pada pohon Huffman menyatakan simbol yang digunakan dalam teks atau pesan. Kode setiap simbol dengan memberi label 0 pada setiap cabang (sisi) kiri dan label 1 untuk setiap canganan kanan. Pohon Huffman untuk string 'ABACCCDA' ditunjukkan pada Gambar 9.28.

Dengan membuat lintasan dari akar ke daun, akan dihasilkan kode untuk setiap simbol. Dari Gambar 9.28 diperoleh kode untuk masing-masing simbol asal sebagai berikut:

$$A = 0,$$

$$B = 110,$$

$$C = 10,$$

$$D = 111$$

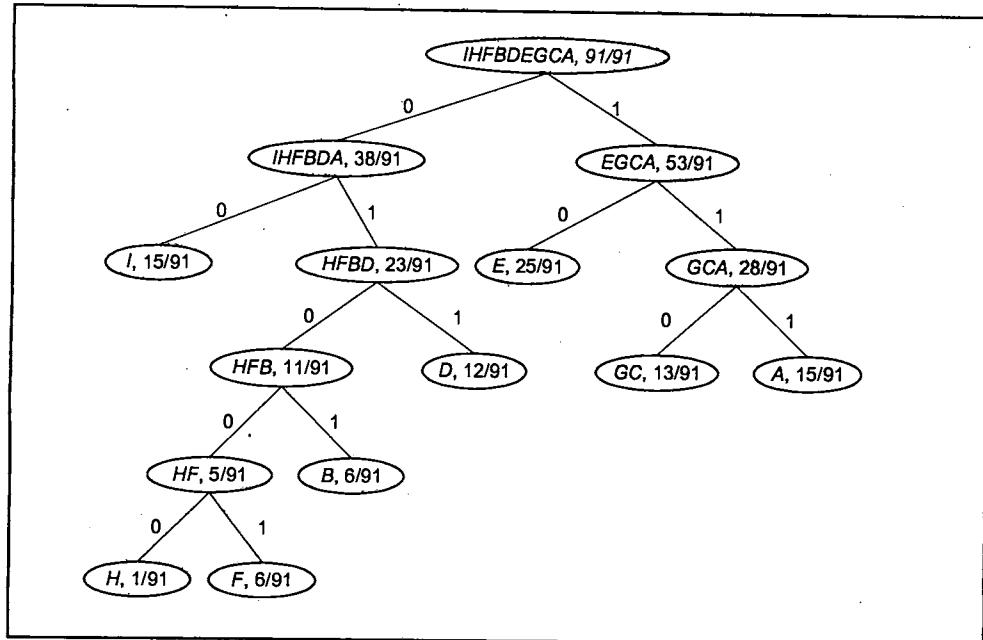
Perhatikan bahwa simbol yang paling sering muncul memiliki kode dengan jumlah bit paling sedikit. Perhatikan pula bahwa tidak ada simbol yang kodennya merupakan kode awalan untuk simbol yang lain. Dengan cara ini, kode yang diawali 0 dapat dipastikan adalah A , sedangkan kode yang diawali 1 mungkin B , C , atau D , yang harus diperiksa lagi dari bit-bit selanjutnya.

Perhatikanlah bahwa kode Huffman tidak bersifat unik, artinya kode untuk setiap karakter berbeda-beda pada setiap pesan bergantung pada kekerapan kemunculan karakter tersebut di dalam pesan. Selain itu, keputusan apakah suatu simpul pada pohon Huffman diletakkan di kiri atau di kanan juga menentukan kode yang dihasilkan (tetapi tidak mempengaruhi panjang kodennya)

Contoh 9.11

Misalkan dari sebuah pesan sudah dihitung kekerapan kemunculan setiap simbol karakter seperti diperlihatkan pada Tabel 9.5. Pohon Huffman-nya ditunjukkan pada Gambar 9.29.

Simbol	Kekerapan	Peluang	Kode Huffman
A	15	15/91	111
B	6	6/91	0101
C	7	7/91	1101
D	12	12/91	011
E	25	25/91	10
F	4	4/91	01001
G	6	6/91	1100
H	1	1/91	01000
I	15	15/91	00



Gambar 9.29 Pohon Huffman untuk Tabel 9.5

9.14 Pohon Pencarian

Pohon pencarian biner (*binary search tree* - BST) mungkin adalah pohon biner yang paling penting, khususnya pada persoalan yang banyak melakukan operasi pencarian, penyisipan, dan penghapusan elemen. Untuk operasi semacam itu, pohon pencarian biner memiliki kinerja yang lebih baik daripada struktur data lain, yang dalam hal ini waktu pencarian.

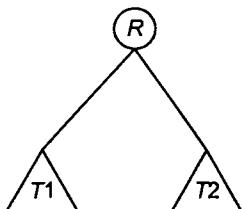
Simpul pada pohon pencarian dapat berupa *field kunci (kunci)* pada data *record*, atau data itu sendiri (dengan catatan setiap data adalah unik). Kunci adalah nilai yang membedakan setiap simpul dengan simpul yang lainnya. Kunci harus unik, karena itu tidak ada dua buah simpul atau lebih yang mempunyai kunci yang sama.

Pohon pencarian biner adalah pohon biner yang setiap kuncinya diatur dalam suatu urutan tertentu. Ketentuan pengaturan kunci adalah sebagai berikut:

Jika R adalah akar, dan semua kunci yang tersimpan pada setiap simpul tidak ada yang sama, maka (lihat Gambar 9.30):

- (a) semua simpul pada upapohon kiri mempunyai kunci lebih kecil dari Kunci(R)

- (b) semua simpul di upapohon kanan mempunyai kunci nilai lebih besar dari $\text{Kunci}(R)$



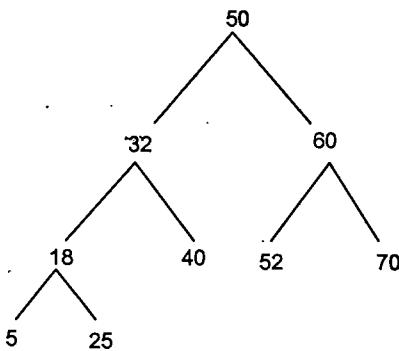
$\text{Kunci}(T1) < \text{Kunci}(R)$
 $\text{Kunci}(T2) > \text{Kunci}(R)$

Gambar 9.30 Skema pohon pencarian

Contoh 9.12

Gambar 9.31 adalah sebuah pohon pencarian untuk data masukan dengan urutan sebagai berikut:

50, 32, 18, 40, 60, 52, 5, 25, 70.



Gambar 9.31 Contoh pohon pencarian

Perhatikanlah dari Gambar 9.31 di atas, bahwa simpul di upapohon kiri 50 mempunyai kunci lebih kecil dari 50, dan simpul di upapohon kanan 50 mempunyai kunci lebih besar dari 50. Secara rekursif, simpul lainnya juga mempunyai pola demikian.

Sesuai namanya, struktur pohon pencarian dimaksudkan untuk memberikan pengaksesan yang cepat terhadap data di simpul. Pencarian data di pohon pencarian dilakukan melalui kunci. Pencarian selalu dimulai dari simpul akar. Kunci di simpul akar dibandingkan dengan nilai yang dicari (x). Jika kunci di simpul akar tidak sama dengan x , pencarian dilanjutkan di upapohon kiri atau di upapohon kanan, bergantung pada apakah x lebih kecil dari kunci di akar atau x lebih besar daripada kunci di akar. Pembandingan demikian diteruskan sampai x sama dengan nilai suatu kunci atau tercapai sebuah daun.

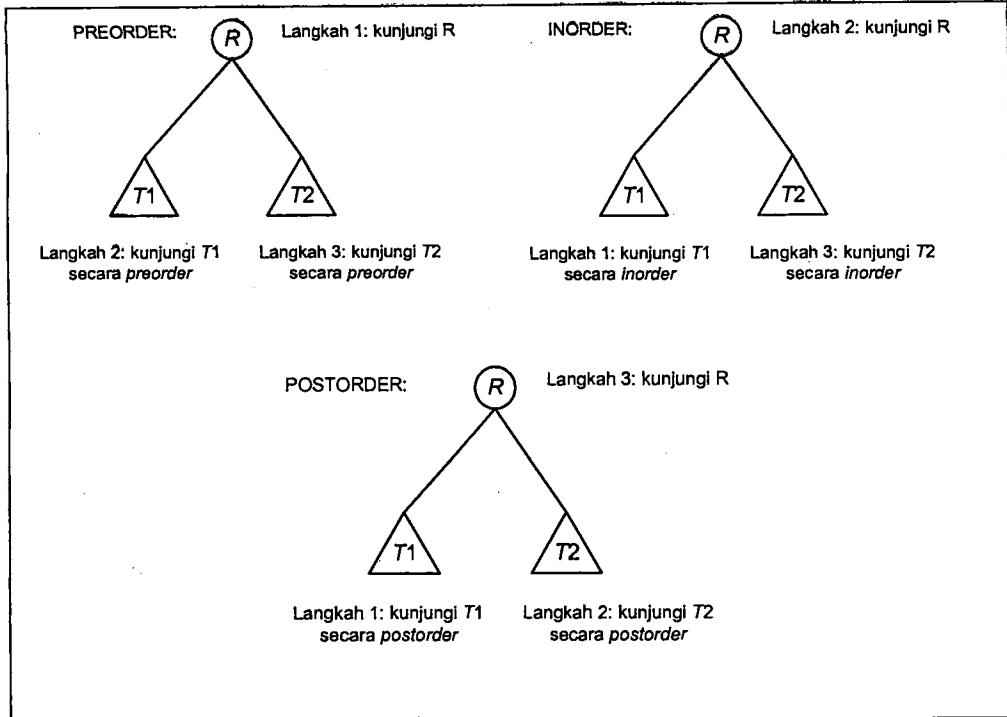
9.15 Traversal Pohon Biner

Operasi dasar yang sering dilakukan pada pohon biner ialah mengunjungi (*traversal*) setiap simpul tepat satu kali. Misalkan T adalah pohon biner, akarnya R , upapohon kiri T_1 dan upapohon kanan T_2 (Gambar 9.32).

Ada tiga macam skema mengunjungi simpul-simpul di dalam pohon biner T :

1. *Preorder*
 - (i) Kunjungi R (sekaligus memproses simpul R)
 - (ii) Kunjungi T_1 secara *preorder*
 - (iii) Kunjungi T_2 secara *preorder*
2. *Inorder*
 - (i) Kunjungi T_1 secara *inorder*
 - (ii) Kunjungi R (sekaligus memproses simpul R)
 - (iii) Kunjungi T_2 secara *inorder*
3. *Postorder*
 - (i) Kunjungi T_1 secara *postorder*
 - (ii) Kunjungi T_2 secara *postorder*
 - (iii) Kunjungi R (sekaligus memproses simpul R)

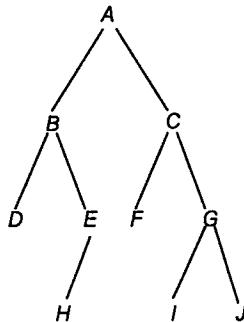
Proses yang dilakukan terhadap simpul yang dikunjungi misalnya mencetak informasi yang disimpan di dalam simpul, memanipulasi nilai, dan sebagainya.



Gambar 9.32 Skema mengunjungi pohon biner

Contoh 9.13

Tinjau pohon biner T di bawah ini:

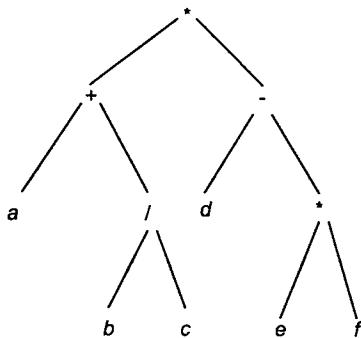


Lintasan *preorder*, *inorder*, dan *postorder* dari T adalah:

- preorder:* $A, B, D, E, F, C, F, G, I, J$
inorder: $D, B, H, E, A, F, C, I, G, J$
postorder: $D, H, E, B, F, I, J, G, C, A$

Ada korespondensi antara ketiga cara mengunjungi pohon biner dengan notasi *prefix*, *infix*, dan *postfix*. Misalkan pohon yang dikunjungi adalah pohon ekspresi pada Gambar 9.33. Penelusuran pohon secara *preorder*, *inorder*, dan *postorder* masing-masing menghasilkan ekspresi dalam notasi *prefix*, *infix*, dan *postfix* sebagai berikut:

<i>preorder</i>	: $* + a / b \ c - d * e f$	(<i>prefix</i>)
<i>inorder</i>	: $a + b / c * d - e * f$	(<i>infix</i>)
<i>postorder</i>	: $a \ b \ c \ / + \ d \ e \ f * \ - *$	(<i>postfix</i>)

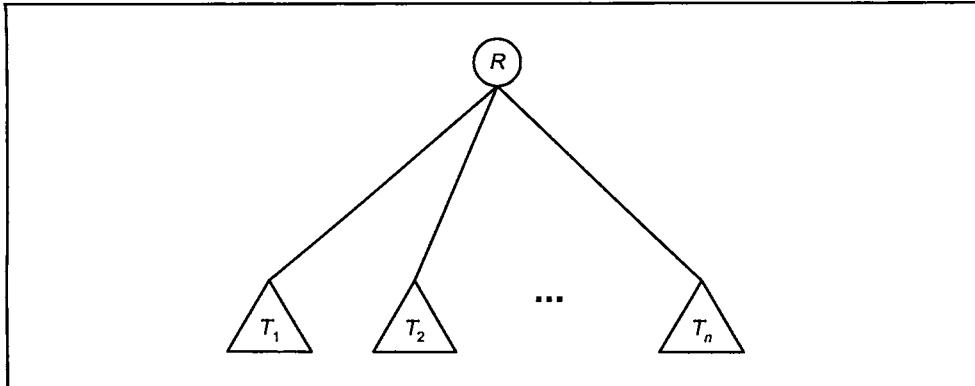


Gambar 9.33 Pohon ekspresi

Preorder, Inorder, dan Postorder pada Pohon m-ary

Skema *preorder*, *postorder*, dan *inorder* pada pohon biner dapat dirampatkan sehingga berlaku untuk sembarang pohon *m-ary*. Misalkan T adalah pohon *m-ary*, akarnya R , dan upapohnnya adalah T_1, T_2, \dots, T_n (Gambar 9.34). Traversal *preorder*, *inorder* dan *postorder* pada pohon *m-ary* didefinisikan sebagai berikut:

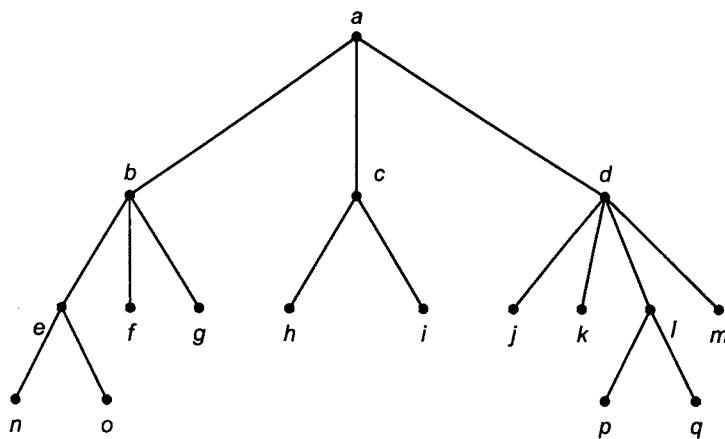
1. *Preorder*
 - (i) Kunjungi R
 - (ii) Kunjungi T_1, T_2, \dots, T_n secara *preorder*
2. *Inorder*
 - (i) Kunjungi T_1 secara *inorder*
 - (ii) Kunjungi R
 - (iii) Kunjungi T_2, T_3, \dots, T_n secara *inorder*
3. *Postorder*
 - (i) Kunjungi T_1, T_2, \dots, T_n secara *postorder*
 - (ii) Kunjungi R



Gambar 9.34 Skema pohon *m*-ary

Contoh 9.14

Tentukan hasil kunjungan *preorder*, *inorder*, dan *postorder* pada pohon 4-ary berikut ini:



Penyelesaian:

Lintasan hasil traversal:

preorder: $a, b, e, n, o, f, g, c, h, i, d, j, k, l, p, q, m$

inorder: $n, e, o, b, f, g, a, h, c, i, j, d, k, p, l, q, m$

postorder: $n, o, e, f, g, b, h, i, c, j, k, p, q, l, m, d, a$

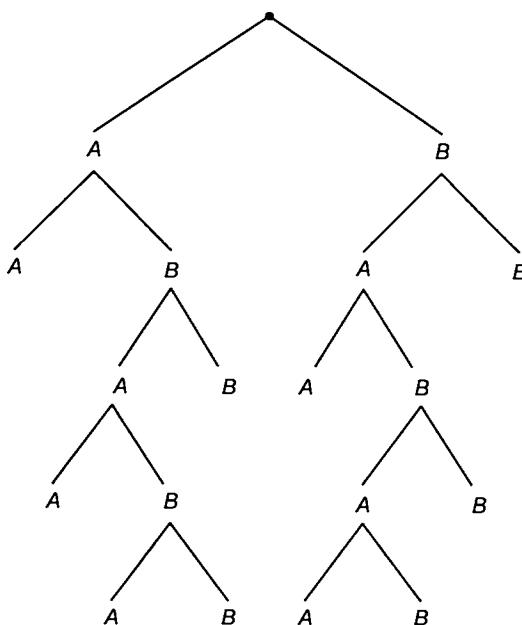
9.16 Ragam Soal dan Pembahasan

Contoh 9.15

[LIP92] Gunakan pohon berakar untuk menggambarkan semua kemungkinan hasil dari pertandingan tenis antara dua orang pemain, Anton dan Budi, yang dalam hal ini pemenangnya adalah pemain yang pertama memenangkan dua set berturut-turut atau pemain yang pertama memenangkan total tiga set.

Penyelesaian:

Pohon berakar yang menggambarkan semua kemungkinan hasil pertandingan tenis untuk mencari pemenang pertandingan tenis ditunjukkan di bawah ini ($A = \text{Anton}$, $B = \text{Budi}$):



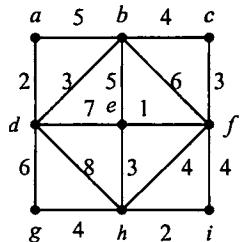
Setiap simpul, kecuali akar, menyatakan pemenang pertandingan. Setiap lintasan dari akar ke daun menyatakan kejadian (yaitu, siapa yang memenangkan set) yang menuju ke suatu hasil pertandingan. Ada 10 daun dan 10 lintasan yang menyatakan kemungkinan hasil pertandingan, yaitu

$AA, ABAA, ABB, ABABA, ABABB, BAA, BABAA, BABAB, BABBB, BB$

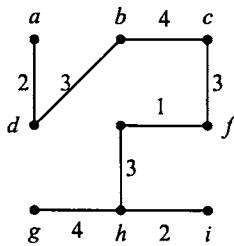
■

Contoh 9.16

Tentukan dan gambarkan pohon merentang minimum dari graf di bawah ini (tahapan pembentukannya tidak perlu ditulis).

**Penyelesaian:**

Pohon merentang minimumnya adalah seperti di bawah ini:



Bobot pohon merentang minimum: $1 + 3 + 3 + 2 + 4 + 4 + 3 + 2 = 22$. ■

Contoh 9.17

Sebuah pohon 4 -ary tingginya 5 . Tanpa menggambarkan pohnnya, hitung jumlah maksimum seluruh simpul di dalam pohon tersebut. Berapa jumlah maksimum seluruh daunnya?

Penyelesaian:

Pohon 4 -ary dengan tinggi 5 berarti $m = 4$ dan $h = 5$. Jumlah maksimum seluruh simpul

$$S \leq \frac{m^{h+1} - 1}{m - 1} = \frac{4^{5+1} - 1}{4 - 1} = 1365 \text{ simpul}$$

sedangkan jumlah maksimum seluruh daun $\leq m^h = 4^5 = 1024$ daun. ■

Contoh 9.18

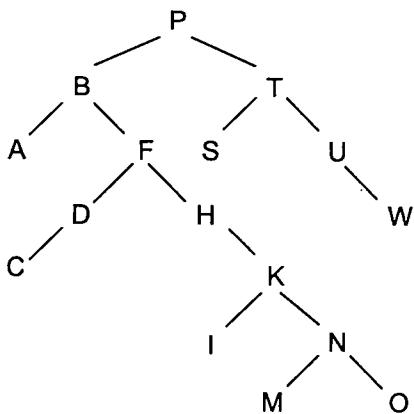
Diberikan masukan berupa rangkaian karakter dengan urutan sebagai berikut:

$P, T, B, F, H, K, N, S, A, U, M, I, D, C, W, O$

- (a) Gambarkan pohon pencarian (*search tree*) yang terbentuk.
 (b) Tentukan hasil penelusuran *preorder*, *inorder*, dan *postorder*, dari pohon jawaban (a) di atas.

Penyelesaian:

Pohon pencarian biner yang terbentuk adalah:



Preorder : P, B, A, F, D, C, H, K, I, N, M, O, T, S, U, W

Inorder : A, B, C, D, F, H, I, K, M, N, O, P, S, T, U, W

Postorder: A, C, D, I, M, O, N, K, H, F, B, S, W, U, T, P

■

Contoh 9.19

Sebuah turnamen catur diikuti oleh 5000 orang peserta. Berapa banyak pertandingan yang harus diadakan sampai ditemukan seorang juara jika turnamen menggunakan sistem gugur, yaitu peserta yang kalah tidak pernah bertanding lagi, dan peserta yang menang akan melawan pemenang pertandingan lainnya? Graf apa yang terbentuk?

Penyelesaian:

Persoalan dapat dimodelkan dengan pohon biner penuh. Daun menyatakan pemain, simpul dalam (termasuk akar) menyatakan pemenang pertandingan. Jumlah simpul dalam berarti total banyak pertandingan yang dilakukan sampai menjadi juara.

i = banyaknya simpul dalam

t = banyaknya simpul daun

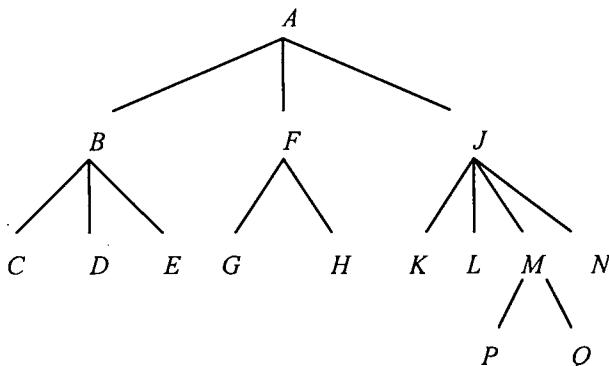
Setiap pertandingan menggugurkan seorang pemain, dan pada akhirnya hanya ada satu pemenang. Maka berlaku hubungan:

$$i = t - 1$$

Jadi banyaknya pertandingan adalah $5000 - 1 = 4999$ pertandingan. Graf yang terbentuk adalah pohon biner penuh.

Contoh 9.20

Tentukan hasil penelusuran *preorder* dan *postorder* dari pohon 4-ary berikut ini:



Penyelesaian:

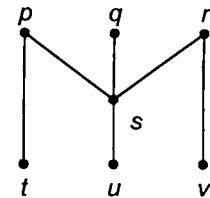
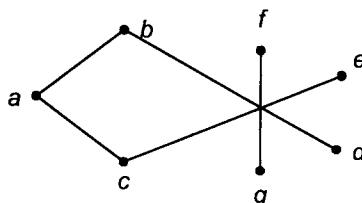
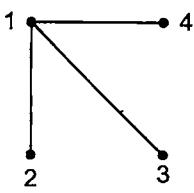
Preorder : $A, B, C, D, E, F, G, H, J, K, L, M, P, Q, N$

Postorder : $C, D, E, B, G, H, F, K, L, P, Q, M, N, J, A$

■

Soal Latihan

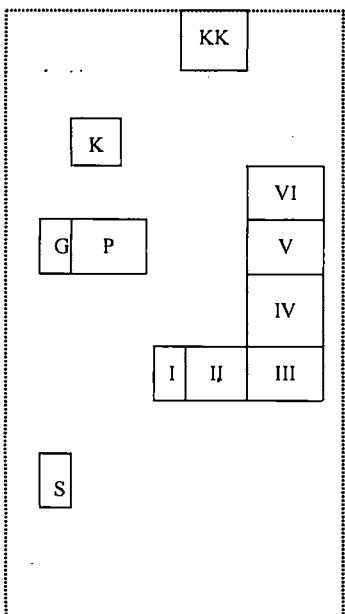
1. Manakah di antara ketiga buah graf di bawah ini yang merupakan pohon?



2. Misalkan T sebuah pohon dengan 50 buah sisi. Pembuangan suatu sisi tertentu dari T menghasilkan dua pohon T_1 dan T_2 yang terpisah satu sama lain. Jika banyaknya simpul di dalam T_1 sama banyak dengan banyaknya simpul di dalam T_2 , tentukan jumlah simpul dan sisi di dalam T_1 dan T_2 .
3. Tunjukkan bahwa jumlah derajat semua simpul di dalam pohon dengan n simpul adalah $2n - 2$.
4. Tunjukkan bahwa sebuah pohon biner teratur mempunyai sejumlah ganjil simpul.
5. Gambarkan semua pohon rentang dari graf lengkap dengan 4 buah simpul.
6. Berapa banyak sisi harus dibuang dari graf terhubung dengan n buah simpul dan m buah untuk menghasilkan pohon rentang?
7. Apakah lintasan Hamilton di dalam sebuah graf terhubung merupakan pohon merentang? Jelaskan jawaban anda.
8. Apakah semua pohon merentang T dari graf terhubung G harus mengandung jumlah sisi yang sama? Jelaskan alasannya (bukan dengan contoh).
9. Apakah semua pohon planar? Jika ya, jelaskan mengapa? Jika tidak, dapatkah anda temukan pohon yang tidak planar?
10. Buatlah pohon Huffman untuk string ‘*abaaccdeba*’ dengan ketentuan: simbol dengan peluang lebih kecil sebagai anak kiri dan simbol dengan peluang lebih besar sebagai anak kanan, sisi kiri dilabeli dengan 0 dan sisi kanan dengan 1. Tuliskan kode Huffman untuk setiap simbol pembentuk *string*, selanjutnya tuliskan rangkaian bit yang merepresentasikan *string* tersebut dengan kode Huffman.
11. Bangunlah pohon Huffman untuk string ‘*makan malam di mana*’ (tidak termasuk tanda petik) dengan ketentuan: simbol dengan peluang lebih kecil sebagai anak kiri dan simbol dengan peluang lebih besar sebagai anak kanan, sisi kiri

dilabeli dengan 0 dan sisi kanan dengan 1. Tentukan kode Huffman untuk setiap karakter, dan tentukan representasi biner dari string yang telah dimampatkan itu.

12. Gambarkan pohon ekspresi untuk (a) $b^2 - 4ac$ dan (b) $((a_3x + a_2)x + a_1)x + a_0$
13. (a) Gambarkan sebuah pohon biner dengan tujuh simpul dan hanya satu buah daun.
 (b) Gambarkan sebuah pohon biner dengan tujuh simpul
14. Pada alam hari harus dipasang lampu-lampu khusus di gedung-gedung utama sebuah sekolah dasar. Lampu-lampu itu dinyalakan dan dimatikan dari gardu Satpam sekolah. Diagram di sebelah kiri menunjukkan tata letak sekolah itu. Digunakan simbol-simbol I, II, III, IV, V, dan VI untuk kelas 1 sampai kelas 6, KK untuk kamar kecil, P untuk perpustakaan, G untuk ruang guru, K untuk kantin, dan S untuk gardu Satpam. Tabel di sebelah kanan menunjukkan panjang kabel listrik dalam satuan meter antar setiap lampu yang dipasang.
 (a) Gambarkan graf yang memodelkan persoalan.
 (b) Bagaimana cara menghubungkan lampu-lampu itu satu dengan yang lain dengan menggunakan kabel listrik yang panjangnya *minimum*? Berapa panjang minimum kabel yang diperlukan? (dalam anda menjawab soal ini, jelaskan pendekatan/metode yang anda gunakan).



Panjang kabel listrik antar ruangan

	I	II	III	IV	V	VI	G	P	K	KK	S
I	-	2 5	-	-	-	-	8 0	6 0	-	-	5 0
II	2 5	-	4 5	-	-	-	-	6 5	-	-	-
III	-	4 5	-	2 0	-	-	-	5 5	-	-	-
IV	-	-	2 0	-	4 7	-	-	6 0	-	-	-
V	-	-	-	4 7	-	4 0	-	4 5	-	-	-
VI	-	-	-	-	4 0	-	-	-	3 0	6 0	-
G	8 0	-	-	-	-	-	-	5 5	9 0	-	8 5
P	6 0	6 5	5 5	6 0	4 5	-	5 5	-	6 0	-	-
K	-	-	-	-	-	3 0	9 0	6 0	-	9 0	-
KK	-	-	-	-	-	6 0	-	-	9 0	-	-
S	5 0	-	-	-	-	-	8 5	-	-	-	-

15. Diberikan kode Huffman sebagai berikut: $a:001$, $b:0001$, $e:1$, $r:0000$, $s:0100$, $t:011$, $x:01010$.
 - (a) Gambarkan pohon Huffman yang merepresentasikan kode tersebut.
 - (b) *Decode* rangkaian bit berikut: 011110010100110001001
16. Sebuah surat berantai dimulai ketika seseorang mengirim sebuah surat kepada 5 orang lainnya. Tiap orang yang menerima surat mengirimkan surat tersebut kepada 5 orang lain yang belum pernah menerima surat tersebut atau kepada orang belum pernah mengirimkannya ke orang lain. Misalkan ada 10.000 orang yang mengirimkan surat tersebut sebelum rantai berakhir. Berapa banyak orang yang menerima surat tersebut dan berapa banyak yang tidak mengirimkannya?
17. Perhatikan situasi bisnis khusus ini. Sebuah perusahaan yang menjual produk-produknya di dua wilayah geografis utama merencanakan untuk memperkenalkan sebuah produk baru. Prosedur normal untuk pengenalan produk adalah sebagai berikut: Pertama, produk diperkenalkan dalam sebuah uji coba (pasar kecil) di wilayah I. Jika produk gagal, ini tidak dilanjutkan; jika ini sukses maka produk diperkenalkan ke seluruh wilayah I. Jika produk sukses di wilayah I, produk diperkenalkan ke seluruh wilayah II; jika tidak, produk diperkenalkan dalam sebuah uji coba (pasar kecil) di wilayah II. Lagi, jika ini berjalan suskes, produk akan diperkenalkan ke seluruh wilayah. Gunakan pohon berakar untuk menentukan semua kemungkinan hasil dari prosedur pengenalan produk.

Esensi dari matematika adalah kebebasannya.
(George Cantor)

BAB 10

Kompleksitas Algoritma

Karena kebodohan kita membuat kesalahan,
dan dari kesalahan kita belajar.
(Anonim)

Algoritma adalah urutan logis langkah-langkah penyelesaian masalah secara sistematis. Sebuah algoritma tidak saja 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. Bila 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. Maka, untuk ukuran masukan yang besar algoritma tersebut menjadi tidak mangkus. Masalah kemangkus (*efficiency*) algoritma merupakan pokok bahasan bab ini.

10.1 Kemangkus Algoritma

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

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, bila mengurutkan 1000 buah 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 . Bila n meningkat, maka sumberdaya waktu/ruang yang dibutuhkan juga meningkat. Seberapa besar peningkatan sumberdaya itu menentukan apakah algoritmanya mangkus atau tidak.

Kemangkus algoritma juga berguna dalam membanding-bandtingkan algoritma. Sebuah masalah dapat mempunyai 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 buah data, pertanyaan yang sering muncul adalah: bagaimana memilih algoritma yang terbaik untuk diimplementasikan? Untuk menjawabnya, kita memerlukan kriteria formal yang digunakan untuk menilai algoritma yang terbaik. Kriteria itu tidak lain adalah kemangkus algoritma.

10.2 Mengapa Kita Memerlukan Algoritma yang Mangkus?

Mana yang lebih baik: menggunakan algoritma yang waktu eksekusinya cepat dengan komputer standard ataukah menggunakan algoritma yang waktunya tidak cepat tetapi dengan komputer yang cepat? Pertanyaan semacam ini seringkali muncul ketika seseorang akan memecahkan suatu masalah dengan komputer. Sebagai ilustrasi [BRA88], misalkan untuk menyelesaikan sebuah masalah tertentu telah tersedia:

1. algorima yang waktu eksekusinya dalam orde eksponensial (2^n), dengan n adalah jumlah masukan yang diproses, dan
2. sebuah komputer yang mampu menjalankan program dengan masukan berukuran n dalam waktu $10^{-4} \times 2^n$ detik.

Dengan algoritma dan komputer tersebut, maka dapat dihitung bahwa untuk

$n = 10$, dibutuhkan waktu eksekusi kira-kira $1/10$ detik,

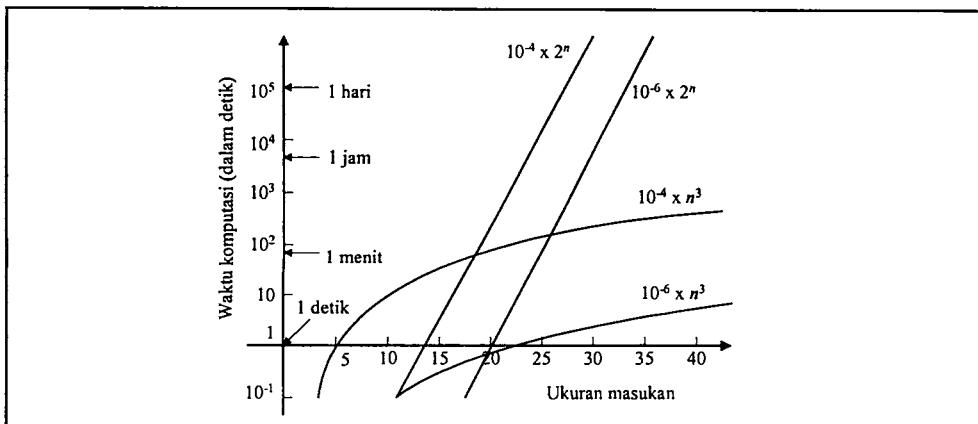
$n = 20$, dibutuhkan waktu eksekusi kira-kira 2 menit,

$n = 30$, dibutuhkan waktu eksekusi lebih dari satu hari.

Misalkan kita dapat menjalankan komputer tanpa gangguan selama satu tahun, maka waktu satu tahun itu hanya dapat menyelesaikan persoalan dengan masukan sebanyak 38.

Karena kita perlu menyelesaikan masalah dengan jumlah masukan yang lebih besar, mungkin kita harus membeli mesin baru yang 100 kali lebih cepat daripada komputer semula (menjadi 10^6). Dengan algoritma yang sama, kita sekarang dapat menyelesaikan masalah dengan masukan sebanyak n dalam waktu $10^{-6} \times 2^n$ detik. Bila kita menjalankan mesin baru itu selama satu tahun penuh, kita hanya dapat menyelesaikan persoalan dengan $n = 45$. Secara umum, jika sebelumnya kita dapat menyelesaikan persoalan untuk masukan sebanyak n selang waktu tertentu, komputer baru itu akan dapat menyelesaikan persoalan dengan masukan paling banyak $n + 7$ dalam waktu yang sama.

Sebagai gantinya, kita memutuskan untuk menaruh perhatian pada algoritmanya. Misalkan kita menemukan sebuah algoritma baru yang dapat menyelesaikan masalah semula dalam waktu orde kubik (n^3). Bayangkanlah, dengan menggunakan komputer pertama, algoritma baru ini dapat menyelesaikan masalah dengan masukan sebanyak n dalam waktu $10^{-4} \times n^3$ detik. Dalam waktu satu hari kita dapat menyelesaikan masalah dengan jumlah masukan lebih besar dari 900; dan dalam waktu satu tahun komputasi ukuran masukan yang dapat diselesaikan hampir mencapai 6800 lebih. Bahkan dengan komputer yang kedua, jumlah masukan yang dapat diproses selama satu tahun komputasi menjadi lebih banyak lagi, yaitu 31.500 lebih. Hal ini diperlihatkan pada Gambar 10.1 di bawah ini. Jelaslah bahwa algoritma kedua lebih mangkus – yang berarti lebih bagus dibandingkan dengan algoritma pertama.



Gambar 10.1 Kebutuhan waktu algoritma terhadap ukuran masukan

10.3 Kebutuhan Waktu dan Ruang

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

Biasanya orang mengukur kebutuhan waktu sebuah algoritma dengan mengeksekusi langsung algoritma tersebut pada sebuah komputer, lalu dihitung berapa lama durasi waktu yang dibutuhkan untuk menyelesaikan sebuah persoalan dengan n yang berbeda-beda. Keakuratan waktu eksekusi algoritma dapat diperoleh dengan tidak menghitung kebutuhan waktu untuk menampilkan antarmuka program, operasi msukan/keluaran (baca, tulis), dan sebagainya. Jadi, benar-benar yang dihitung adalah kebutuhan waktu untuk bagian algoritma yang inti saja.

Sebagai ilustrasi, tinjau masalah menghitung rata-rata dari n buah data bilangan bulat. Kita mengasumsikan data masukan sudah dibaca dan disimpan di dalam elemen-elemen larik (tabel) a_1, a_2, \dots, a_n . Jadi, kita hanya memperhatikan bagian perhitungan rata-ratanya saja. Bagian perhitungan rata-rata dinyatakan di dalam prosedur HitungRerata pada Algoritma 10.1 di bawah ini. Jalankan program yang mengandung prosedur ini pada sebuah komputer. Hitung selisih waktu antara sebelum pemanggilan prosedur dan sesudah pemanggilan prosedur. Selisih kedua waktu ini adalah kebutuhan waktu aktual untuk menghitung rata-rata n buah data.

```
procedure HitungRerata(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
  i : integer
  jumlah : real

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

Algoritma 10.1 Algoritma menghitung nilai rata-rata.

Sayangnya, model perhitungan kebutuhan waktu algoritma seperti di atas ini kurang dapat diterima, karena dua alasan. Alasan pertama, arsitektur komputer yang berbeda menghasilkan waktu yang berbeda pula untuk melaksanakan operasi-operasi dasar (operasi penambahan, perkalian, pembagian, perbandingan, dan sebagainya), sehingga kita tidak mempunyai ukuran kebutuhan waktu yang unik untuk sebuah algoritma. Misalnya, bila anda jalankan sebuah program pada komputer *IBM*, kebutuhan waktunya akan berbeda bila program yang sama dieksekusi pada komputer *Macintosh*.

Alasan pertama ini dapat dijelaskan sebagai berikut. Komputer dengan arsitektur yang berbeda akan berbeda pula perintah (*instruction*) -dalam bahasa mesin tentunya- yang dimilikinya, dan akan berbeda pula kecepatan (*speed*) operasi piranti kerasnya. Dengan demikian, perbedaan di atas akan menghasilkan ukuran waktu (dan kebutuhan ruang memori) yang berbeda-beda pada setiap jenis komputer untuk program yang sama (program adalah realisasi algoritma dalam bahasa tingkat tinggi). Sebagai contoh, komputer tercepat saat ini dapat mengerjakan operasi dasar (seperti penjumlahan, perkalian, pembandingan, atau mempertukarkan dua buah bit) dalam waktu 10^{-9} detik, tetapi komputer PC melakukannya dalam 10^{-6} detik, 1000 kali lebih lambat untuk operasi yang sama [ROS99].

Alasan kedua, kebutuhan waktu sebuah algoritma bergantung pada *compiler* bahasa pemrograman yang digunakan. *Compiler* yang berbeda akan menerjemahkan program (dalam bahasa tingkat tinggi) ke dalam kode mesin (*object code* - dalam bahasa tingkat rendah) yang berbeda pula. Sebagai akibatnya, kode mesin yang berbeda akan menggunakan ruang memori dan memerlukan waktu pelaksanaan program yang berbeda pula.

Karena kebutuhan ruang dikaitkan dengan struktur data yang digunakan untuk mengimplementasikan algoritma, sementara topik struktur data di luar bahasan buku ini, maka kebutuhan ruang tidak dibahas di sini.

10.4 Kompleksitas Waktu dan Ruang

Secara teoritis, model abstrak pengukuran waktu/ruang harus independen dari pertimbangan mesin dan *compiler* apapun. Model abstrak seperti itu dapat dipakai 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 diekspresikan sebagai jumlah tahapan komputasi yang dibutuhkan untuk menjalankan algoritma sebagai fungsi dari ukuran masukan n . Kompleksitas ruang diekspresikan sebagai jumlah memori yang digunakan oleh struktur data yang terdapat di dalam algoritma

sebagai fungsi dari ukuran masukan n . Dengan menggunakan besaran kompleksitas waktu/ruang algoritma, kita dapat menentukan *laju* peningkatan waktu (ruang) yang diperlukan algoritma dengan meningkatnya ukuran masukan n .

Terminologi Kompleksitas Waktu/Ruang

Terminologi yang diperlukan dalam membahas kompleksitas waktu dan kompleksitas ruang suatu algoritma adalah:

1. Ukuran besar masukan data untuk suatu algoritma, n .
Sebagai contoh, dalam algoritma pengurutan elemen-elemen larik, n adalah jumlah elemen larik, sedangkan dalam algoritma perkalian matriks n adalah ukuran matriks $n \times n$. Pada beberapa kasus, ukuran masukan lebih tepat menggunakan dua buah besaran, misalnya jika masukan algoritma adalah graf, maka ukuran masukan adalah jumlah simpul dan jumlah sisi.
2. Kompleksitas waktu, $T(n)$, adalah jumlah operasi yang dilakukan untuk melaksanakan algoritma sebagai fungsi dari ukuran masukan n .
3. Kompleksitas ruang, $S(n)$, adalah ruang memori yang dibutuhkan algoritma sebagai fungsi dari ukuran masukan n .

Sebagaimana sudah diungkapkan pada bagian sebelumnya, kita tidak membahas kompleksitas ruang karena kebutuhan ruang dikaitkan dengan struktur data yang digunakan untuk mengimplementasikan algoritma, sementara topik struktur data di luar bahasan buku ini, maka kompleksitas ruang tidak akan kita tinjau. Pertimbangan lain mengapa hanya meninjau kompleksitas waktu adalah tingkat kekritisan memori. Ukuran memori sekarang ini tidak lagi menjadi persoalan kritis, karena komputer sekarang mempunyai ukuran memori yang besar dibandingkan dengan komputer *mainframe* 25 tahun yang lalu. Bahkan, bila memori masih kurang, memori sekunder pun dapat dijadikan sebagai memori tambahan (memori semu, *virtual memory*). Tetapi, ini tidak berarti kita melupakan kompleksitas ruang, hanya saja kompleksitas waktu akan selalu menjadi isu utama dalam merancang suatu algoritma.

Kompleksitas Waktu

Setelah menetapkan ukuran masukan, maka langkah selanjutnya dalam mengukur kompleksitas waktu adalah menghitung banyaknya operasi yang dilakukan oleh algoritma. Di dalam sebuah algoritma mungkin terdapat banyak sekali jenis-jenis operasi, misalnya operasi penjumlahan (termasuk pengurangan), operasi perbandingan, operasi pembagian, operasi pembacaan, pemanggilan prosedur, dan sebagainya.

Contoh 10.1

Sebagai contoh pertama, tinjau kembali Algoritma 10.1. Jenis-jenis operasi yang terdapat di dalam algoritma HitungRerata adalah

- operasi pengisian nilai (dengan operator “ \leftarrow ”)
- operasi penjumlahan (dengan operator “ $+$ ”)
- operasi pembagian (dengan operator “ $/$ ”)

Mari kita hitung kompleksitas waktu algoritam tersebut dengan cara menghitung masing-masing jumlah operasi. Jika operasi tersebut berada di dalam sebuah kalang (*loop*), maka jumlah operasinya bergantung berapa kali kalang tersebut diulang.

(i) Operasi pengisian nilai

Jumlah $\leftarrow 0,$	1 kali
$k \leftarrow 1,$	1 kali
jumlah \leftarrow jumlah + $a_k,$	n kali
$k \leftarrow k+1,$	n kali
$x \leftarrow \text{jumlah}/n),$	1 kali

Jumlah seluruh operasi pengisian nilai adalah

$$t_1 = 1 + 1 + n + n + 1 = 3 + 2n$$

(ii) Operasi penjumlahan

jumlah + $a_k,$	n kali
$k \leftarrow 1,$	n kali

Jumlah seluruh operasi penjumlahan adalah

$$t_2 = n + n = 2n$$

(iii) Operasi pembagian

Jumlah seluruh operasi pembagian adalah

$$\text{jumlah}/n) \quad 1 \text{ kali}$$

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

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

Idealnya, kita memang harus menghitung semua operasi yang ada di dalam suatu algoritma, seperti pada Contoh 10.1 di atas. Namun, untuk alasan praktis, kita cukup menghitung jumlah operasi abstrak yang *mendasari* suatu algoritma, dan memisahkan analisisnya dari implementasi [SED92]. Operasi abstrak ini disebut **operasi dasar** (*basic operation*). Sebagai contoh, pada algoritma pencarian, operasi abstrak yang mendasarinya adalah operasi perbandingan x dengan elemen-elemen larik. Dengan menghitung berapa kali operasi

perbandingan elemen untuk tiap-tiap nilai n pada dua buah algoritma pencarian, kita memperoleh kemangkusian relatif dari kedua buah algoritma tersebut.

Pada algoritma pengurutan, operasi dasar adalah operasi perbandingan elemen-elemen larik dan operasi pertukaran elemen-elemen. Jadi, kita cukup menghitung berapa kali operasi perbandingan dan berapa kali operasi pertukaran elemen di dalam algoritma pengurutan. Kedua operasi dasar ini dihitung secara terpisah, karena di dalam algoritma pengurutan jumlah operasi perbandingan tidak sama dengan jumlah operasi pertukaran.

Pada algoritma perkalian dua buah matriks $A \times B$ yang masing-masing berukuran $n \times n$, operasi dasar yang bagus untuk dipilih adalah operasi penjumlahan dan perkalian. Jadi, kita cukup menghitung berapa kali operasi penjumlahan dan berapa kali operasi perkalian pada algoritma perkalian dua buah matriks. Kedua operasi dasar ini dihitung secara terpisah.

Pada algoritma evaluasi polinom, $p(x) = a_0 + a_1x + a_2x^2 + \dots + a_nx^n$, operasi operasi dasar di dalam algoritmanya juga operasi penjumlahan dan perkalian. Jadi, kita cukup menghitung berapa kali operasi penjumlahan dan berapa kali operasi perkalian pada algoritma evaluasi polinom.

Contoh 10.2

Tinjau kembali algoritma HitungRerata pada Contoh 10.1. Operasi yang mendasar pada algoritma menghitung rata-rata adalah operasi penjumlahan elemen-elemen larik (yaitu operasi di dalam instruksi jumlah \leftarrow jumlah + a_k), yang mana dilaksanakan sebanyak n kali, yaitu sejumlah pengulangan yang dilakukan. Operasi-operasi lainnya, seperti pembagian) boleh kita abaikan (tidak dihitung). Jika kompleksitas waktu algoritma dihitung berdasarkan jumlah operasi penjumlahan ini, maka kompleksitas waktu HitungRerata adalah $T(n) = n$. ■

Contoh 10.3

Tinjau algoritma untuk mencari elemen terbesar di dalam sebuah larik (*array*) yang berukuran n elemen. Tentukan kompleksitas waktu algoritma, yang dalam hal ini kompleksitas waktu algoritma dihitung berdasarkan jumlah operasi perbandingan elemen-elemen, karena perbandingan adalah operasi dasar yang digunakan di dalam algoritma mencari nilai terbesar (operasi perbandingan $i \leq n$ bukan operasi perbandingan elemen larik, jadi tidak kita hitung).

Penyelesaian:

Operasi perbandingan elemen larik yang dimaksudkan di dalam algoritma adalah $A[i] > \text{maks}$. Operasi ini terdapat di dalam kalang *for*. Jumlah operasi perbandingan elemen ditentukan oleh berapa kali kalang *for* dieksekusi, yaitu $n - 1$ kali. Dengan demikian, kompleksitas waktu algoritma CariMaks adalah $T(n) = n - 1$. ■

```

procedure CariMaks(input a1, a2, ..., an : integer, output maks : integer)
{ Mencari elemen terbesar dari sekumpulan elemen larik integer a1, a2,
..., an. Elemen terbesar akan disimpan di dalam maks.
Masukan: a1, a2, ..., an
Keluaran: maks (nilai terbesar)
}

Deklarasi
i : integer

Algoritma
maks ← a1
i ← 2
while i ≤ n do
    if ai > maks then
        maks ← ai
    endif
    i ← i + 1
endwhile
( i > n )

```

Algoritma 10.2 Algoritma mencari elemen terbesar

Notasi kompleksitas waktu telah membebaskan kita dari pertimbangan spesifikasi komputer untuk menghitung kebutuhan waktu algoritma. Kita tidak peduli komputer apa yang digunakan untuk menjalankan algoritma, namun yang pasti apapun komputer atau bahasa pemrograman yang digunakan, jumlah komputasi di dalam algoritma tersebut tetap, yaitu $T(n)$.

Andaikan kita mengetahui informasi mengenai waktu yang dibutuhkan untuk melakukan operasi tertentu pada komputer tertentu, kita dapat menghitung kebutuhan waktu aktual yang sesungguhnya untuk sebuah algoritma. Misalnya pada algoritma CariMaks diandaikan satu kali operasi perbandingan di dalam komputer PC membutuhkan waktu 10^{-6} detik, maka untuk masukan sebanyak 1000 elemen, kebutuhan waktu algoritma CariMaks dihitung berdasarkan operasi perbandingan elemen saja, adalah $T(1000) = (1000 - 1) \times 10^{-6} = 0.000999$ detik. Bila algoritma algoritma CariMaks dijalankan pada komputer tercepat saat ini, maka kebutuhan waktu algoritma, dihitung berdasarkan operasi perbandingan elemen saja, adalah $T(1000) = (1000 - 1) \times 10^{-9} = 0.000000999$ detik. Apabila operasi-operasi selain perbandingan juga dihitung, kita akan memperoleh kebutuhan waktu yang lebih presisi. Namun, kita tidak melakuka perhitungan kebutuhan waktu aktual di sini. Kompleksitas waktu $T(n) = n - 1$ sudah cukup memberikan informasi mengenai unjuk kerja algoritma. Dengan kata lain, kita menghitung kebutuhan waktu algoritma secara teoritis, bukan secara praktis.

Hal lain yang harus diperhatikan dalam menghitung 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 buah elemen, a_1, a_2, \dots, a_n . Asumsikan elemen-elemen larik sudah terurut. Jika $a_1 = x$, maka waktu pencarinya 128 kali lebih cepat daripada jika $a_{128} = x$ atau jika x tidak terdapat di dalam larik. Demikian pula, jika $a_{64} = x$, maka waktu pencarinya $1/2$ kali lebih cepat daripada jika $a_{128} = x$. Karena itu, kompleksitas waktu dibedakan atas tiga macam :

1. $T_{\max}(n)$: kompleksitas waktu untuk kasus terburuk (*worst case*), yaitu kebutuhan waktu maksimum yang diperlukan sebuah algoritma sebagai fungsi dari n
2. $T_{\min}(n)$: kompleksitas waktu untuk kasus terbaik (*best case*), yaitu kebutuhan waktu minimum yang diperlukan sebuah algoritma sebagai fungsi dari n
3. $T_{\text{avg}}(n)$: 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 persoalan pencarian diandaikan bahwa data yang dicari mempunyai peluang yang sama untuk terletak di dalam larik.

Contoh 10.4

Diberikan larik bilangan bulat a_1, a_2, \dots, a_n yang telah terurut menaik (tidak ada elemen ganda). Hitunglah kompleksitas waktu terbaik, terburuk, dan rata-rata dari algoritma pencarian beruntun (*sequential search*) di bawah ini. Algoritma pencarian beruntun di bawah ini menghasilkan indeks elemen yang bernilai sama dengan x . Jika x tidak ditemukan, indeks 0 akan dihasilkan (catatlah bahwa algoritma pencarian beruntun ini merupakan versi lain dari algoritma pencarian beruntun yang pernah dituliskan pada Algoritma 5.4 di dalam Bab 4).

```
procedure PencarianBeruntun(input a1, a2, ..., an : integer, x : integer,
                           output idx : integer)
{ Mencari x di dalam elemen a1, a2, ..., an. Lokasi (indeks elemen) tempat
  x ditemukan diisi ke dalam idx. Jika x tidak ditemukan, maka idx diisi
  dengan 0.
  Masukan: a1, a2, ..., an
  Keluaran: idx
}
Deklarasi
  i : integer
  ketemu : boolean { bernilai true jika x ditemukan atau false jika x
                     tidak ditemukan }
Algoritma:
  i ← 1
```

```

ketemu ← false
while (i ≤ n) and (not ketemu) do
    if  $a_i = x$  then
        ketemu←true
    else
        i ← i + 1
    endif
endwhile
( i > n or ketemu )

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

```

Algoritma 10.3 Algoritma pencarian beruntun.

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 ($a_i = x$). Operasi perbandingan yang lain, seperti $i \leq n$ tidak akan dihitung. Operasi perbandingan elemen-elemen larik adalah operasi abstrak yang mendasari algoritma pencarian.

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

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

$$T_{\min}(n) = 1$$

2. *Kasus terburuk:* bila $a_n = x$ 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

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

Cara lain yang dapat digunakan dalam menghitung T_{avg} di atas adalah sbb: asumsikan bahwa peluang x terdapat di sembarang lokasi larik adalah sama, artinya, peluang elemen ke- j = x adalah $1/n$, atau kita tulis $P(a_j = x) = 1/n$. Jika $a_j = x$ maka T_j yang dibutuhkan adalah $T_j = j$. Jumlah perbandingan elemen larik secara rata-rata adalah:

$$T_{\text{avg}}(n) = \sum_{j=1}^n T_j P(A[j] = X) = \sum_{j=1}^n T_j \frac{1}{n} = \frac{1}{n} \sum_{j=1}^n T_j = \frac{1}{n} \sum_{j=1}^n j = \frac{1}{n} \left(\frac{n(n+1)}{2} \right) = \frac{n+1}{2}$$

Contoh 10.5

Diberikan larik bilangan bulat a_1, a_2, \dots, a_n yang telah terurut menaik (tidak ada elemen ganda). Hitunglah kompleksitas waktu terbaik, terburuk, dan rata-rata dari algoritma pencarian biner (*binary search*) di bawah ini. Algoritma pencarian beruntun di bawah ini menghasilkan indeks elemen yang bernilai sama dengan x . Jika x tidak ditemukan, indeks 0 akan dihasilkan

```

procedure PencarianBiner(input a1, a2, ..., an : integer, x : integer,
output idx : integer)
{ Mencari x di dalam elemen a1, a2, ..., an. Lokasi (indeks elemen) tempat
x ditemukan diisi ke dalam idx. Jika x tidak ditemukan, maka idx diisi
dengan 0.
Masukan: a1, a2, ..., an
Keluaran: idx
}

Deklarasi
    i, j, mid : integer
    ketemu : boolean

Algoritma
    i ← 1
    j ← n
    ketemu←false
    while (not ketemu) and ( i ≤ j ) do
        mid ← (i + j) div 2
        if amid = x then
            ketemu ← true
        else
            if amid < x then      { cari di belahan kanan }
                i ← mid + 1
            else                      { cari di belahan kiri }
                j ← mid - 1
            endif
        endif
    endwhile
    {ketemu or i > j }

    if ketemu then
        idx ← mid
    else
        idx ← 0
    endif
}

```

Algoritma 10.4 Algoritma pencarian biner

Penyelesaian:

Algoritma PencarianBiner membagi larik di pertengahan menjadi dua bagian yang berukuran sama ($n/2$ bagian), bagian kiri dan bagian kanan. Jika elemen pertengahan

tidak sama dengan x , keputusan dibuat untuk melakukan pencarian pada bagian kiri atau bagian kanan. Proses bagi-dua dilakukan lagi pada bagian yang dipilih. Perhatikanlah bahwa setiap kali memasuki kalang *while-do* maka ukuran larik yang ditelusuri berkurang menjadi setengah kali ukuran semula: $n, n/2, n/4, \dots$

Kita akan menghitung jumlah operasi perbandingan elemen dengan x yang terjadi selama pencarian ($a_{\text{mid}} = x$). Operasi perbandingan yang lain, seperti $i \leq j$ dan $a_{\text{mid}} < x$ tidak akan dihitung. Untuk penyederhanaan, asumsikan ukuran larik adalah perangkatan dari 2 (yaitu, $n = 2^k$).

1. Kasus terbaik

Kasus terbaik adalah bila x ditemukan pada elemen pertengahan (a_{mid}), dan operasi perbandingan elemen ($a_{\text{mid}} = x$) yang dilakukan hanya satu kali. Pada kasus ini

$$T_{\min}(n) = 1$$

2. Kasus terburuk:

Pada kasus terburuk, elemen x ditemukan ketika ukuran larik = 1. Pada kasus terburuk ini, ukuran larik setiap kali memasuki kalang *while-do* adalah:

$$n, n/2, n/4, n/8, \dots, 1 \quad (\text{sebanyak } ^2\log n \text{ kali})$$

artinya, kalang *while-do* dikerjakan sebanyak $^2\log n$ kali.

Contoh: $n = 128 \Rightarrow 128 \rightarrow 64 \rightarrow 32 \rightarrow 16 \rightarrow 8 \rightarrow 4 \rightarrow 2 \rightarrow 1$ (sebanyak $^2\log 128 = 7$ kali pembagian)

Jumlah operasi perbandingan elemen ($a_{\text{mid}} = x$) adalah:

$$T_{\max}(n) = ^2\log n$$

Kompleksitas waktu rata-rata algoritma pencarian bagidua lebih sulit ditentukan. ■

Kadang-kadang penentuan kasus berguna untuk menghitung jumlah operasi yang bergantung pada kondisi tertentu. Misalnya pada potongan algoritma berikut kita ingin menghitung nilai rata-rata elemen larik yang ganjil:

```
k ← 1
jumlah ← 0
for k ← 1 to n do
    if ak mod 2 = 1 then ( ak ganjil )
        jumlah ← jumlah + ak
    endif
endfor
```

Misalkan kompleksitas waktu algoritma dihitung berdasarkan jumlah operasi penjumlahan elemen (yaitu $\text{jumlah} + a_k$). Instruksi $\text{jumlah} \leftarrow \text{jumlah} + a_k$ hanya dikerjakan jika kondisi $a_k \bmod 2 = 1$ benar (yaitu jika a_k ganjil).

Meskipun demikian, kita tetap dapat menghitung kompleksitas waktu algoritma dengan mengasumsikan bahwa pada kasus terburuk semua elemen bernilai ganjil sehingga kondisi $a_k \bmod 2 = 1$ terpenuhi dan instruksi penjumlahan jumlah $\leftarrow \text{jumlah} + a_k$ dikerjakan sebanyak n kali. Jadi, pada kasus terburuk,

$$T_{\max}(n) = n$$

dan pada kasus terbaik, semua elemen genap sehingga instruksi penjumlahan jumlah $\leftarrow \text{jumlah} + a_k$ tidak pernah dikerjakan, jadi

$$T_{\min}(n) = 0$$

Contoh 10.6

Hitung jumlah operasi perbandingan elemen larik dan jumlah operasi pertukaran pada algoritma pengurutan seleksi (*selection sort*).

```
procedure UrutSeleksi(input/output a1, a2, ..., an : integer)
{ Mengurutkan elemen-elemen a1, a2, ..., an dengan metode selection sort.
  Masukan: a1, a2, ..., an.
  Keluaran: a1, a2, ..., an (sudah terurut menaik).
}

Deklarasi
  i, j, imaks, temp : integer

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

Algoritma 10.5 Algoritma pengurutan (*selection sort*)

Penyelesaian:

Algoritma UrutSeleksi terdiri dari $n - 1$ kali *pass*. Pada setiap *pass*, kita mencari elemen terbesar dari elemen-elemen a_1, a_2, \dots, a_n , lalu mempertukarkan elemen terbesar dengan a_n . *Pass* berikutnya akan mencari elemen terbesar dari dari sekumpulan a_1, a_2, \dots, a_{n-1} , begitu seterusnya sampai larik *pass* terakhir sehingga tinggal satu elemen yang pasti sudah terurut. Operasi abstrak yang mendasari algoritma pengurutan adalah operasi

perbandingan elemen larik ($a_j > a_{i\text{max}}$) dan operasi pertukaran (diwakili oleh tiga buah instruksi: $\text{temp} \leftarrow a_n$, $a_n \leftarrow a_{i\text{max}}$, $a_{i\text{max}} \leftarrow \text{temp}$). Kedua operasi ini kita pisahkan perhitungannya sebagai berikut:

- (i) Jumlah operasi perbandingan elemen

Untuk setiap $pass$ ke- i , $i = n, n - 1, \dots, 2$, operasi perbandingan elemen yang dilakukan adalah sebagai berikut:

$$\begin{aligned} i = n &\rightarrow \text{jumlah operasi perbandingan elemen} = n - 1 \\ i = n - 1 &\rightarrow \text{jumlah operasi perbandingan elemen} = n - 2 \\ i = n - 2 &\rightarrow \text{jumlah operasi perbandingan elemen} = n - 3 \\ &\vdots \\ i = 2 &\rightarrow \text{jumlah operasi perbandingan elemen} = 1 \end{aligned}$$

Jumlah seluruh operasi perbandingan elemen-elemen larik adalah

$$T(n) = (n - 1) + (n - 2) + \dots + 1 = \sum_{k=1}^{n-1} n - k = \frac{n(n-1)}{2}$$

Ini adalah kompleksitas waktu untuk kasus terbaik dan terburuk, karena algoritma Urut Seleksi tidak bergantung pada data masukan apakah sudah terurut atau acak.

- (ii) Jumlah operasi pertukaran

Untuk setiap i dari n sampai 2, terjadi satu kali pertukaran elemen, sehingga jumlah operasi pertukaran seluruhnya adalah

$$T(n) = n - 1.$$

Jadi, algoritma pengurutan seleksi membutuhkan $n(n - 1)/2$ buah operasi perbandingan elemen dan $n - 1$ buah operasi pertukaran. ■

Contoh 10.7

Hitung jumlah operasi perkalian pada algoritma yang menghitung $\sum_{j=n,\lfloor n/2 \rfloor, \lfloor n/4 \rfloor, \dots}^1 \sum_{i=1}^j x_i$.

Asumsikan n adalah perpangkatan dari 2.

```
procedure Kali(input x:integer, n:integer, output jumlah : integer)
{Mengalikan x dengan i = 1, 2, ..., j, yang dalam hal ini j = n, n/2, n/4, ..., 1
 Masukan: x dan n (n adalah perpangkatan dua).
 Keluaran: hasil perkalian (disimpan di dalam peubah jumlah).
}
Deklarasi
  i, j, k : integer

Algoritma
  j ← n
  while j ≥ 1 do
    for i ← 1 to j do
      x ← x * i
```

```

endfor
j ← d div 2
endwhile
{ j > 1 }
jumlah←x

```

Algoritma 10.6 Algoritma perkalian

Penyelesaian:

Untuk

- $j = n$, jumlah operasi perkalian = n ;
- $j = n/2$, jumlah operasi perkalian = $n/2$;
- $j = n/4$, jumlah operasi perkalian = $n/4$;
- ...
- $j = 1$, jumlah operasi perkalian = 1

Jumlah operasi perkalian seluruhnya adalah

$$n + n/2 + n/4 + \dots + 2 + 1$$

yang merupakan deret geometri dengan jumlah = $\frac{n(1 - 2^{2\log n^{-1}})}{1 - \frac{1}{2}} = 2(n - 1)$ ■

10.5 Kompleksitas Waktu Asimptotik

Seringkali kita kurang tertarik dengan kompleksitas waktu yang presisi untuk suatu algoritma, tetapi kita lebih tertarik pada bagaimana waktu terbaik dan waktu terburuk tumbuh bersamaan dengan meningkatnya ukuran masukan. Sebagai contoh, pada Algoritma 10.5, jumlah operasi perbandingan elemen adalah

$$T(n) = n(n - 1)/2$$

Kita mungkin tidak terlalu membutuhkan informasi seberapa tepat jumlah operasi perbandingan elemen pada algoritma pengurutan tersebut. Yang kita butuhkan adalah perkiraan kasar kebutuhan waktu algoritma dan seberapa cepat fungsi kebutuhan waktu itu tumbuh. Hal ini perlu untuk mengetahui kinerja algoritma. Kinerja algoritma baru akan tampak untuk n yang sangat besar, bukan pada n yang kecil. Bila anda menggunakan komputer untuk menjalankan algoritma PencarianBeruntun dan PencarianBiner untuk larik yang berukuran kecil (misalnya $n = 10$), maka perbedaan kecepatan keduanya tidak akan terlihat. Tetapi, bila kedua algoritma tersebut diterapkan untuk larik yang berukuran besar (misalnya $n = 5000$), perbedaan kecepatan keduanya akan terlihat sangat berarti.

Langkah pertama dalam pengukuran kinerja algoritma adalah membuat makna “sebanding”. Gagasannya adalah dengan menghilangkan faktor koefisien di dalam ekspresi $T(n)$. Sebagai contoh, andaikan bahwa kompleksitas waktu terburuk dari sebuah algoritma adalah

$$T(n) = 2n^2 + 6n + 1$$

Untuk n yang besar, pertumbuhan $T(n)$ sebanding dengan n^2 (lihat Tabel 10.1). Pada kasus ini, $T(n)$ tumbuh seperti n^2 tumbuh.

Tabel 10.1 Perbandingan pertumbuhan $T(n)$ dengan n^2

n	$T(n) = 2n^2 + 6n + 1$	n^2
10	261	100
100	2061	1000
1000	2.006.001	1.000.000
10.000	2.000.060.001	1.000.000.000

Jika diperhatikan pada Tabel 10.1 tersebut, suku $6n + 1$ menjadi tidak berarti dibandingkan $2n^2$. Kita dapat mengabaikan suku-suku yang tidak mendominasi perhitungan pada rumus $T(n)$, sehingga kompleksitas waktu $T(n)$ adalah

$$2(n^2) + \text{suku-suku lainnya}$$

Dengan mengabaikan koefisien 2 pada $2n^2$, kita melihat $T(n)$ tumbuh seperti n^2 tumbuh saat n bertambah. Kita katakan bahwa $T(n)$ berorde n^2 dan kita tuliskan

$$T(n) = O(n^2)$$

yang dibaca “ $T(n)$ adalah O dari n^2 ”. Jadi, kita telah mengganti ekspresi seperti $T(n) = 2n^2 + 6n + 1$ dengan ekspresi yang lebih sederhana seperti n^2 yang tumbuh pada kecepatan yang sama dengan $T(n)$. Notasi “ O ” disebut notasi “ O -Besar” (*Big-O*) yang merupakan salah satu dari tiga notasi **kompleksitas waktu asimptotik**. Definisi formal notasi O -Besar dituliskan di bawah ini.

Notasi O -Besar

DEFINISI 10.1. $T(n) = O(f(n))$ (dibaca “ $T(n)$ adalah $O(f(n))$ ” yang artinya $T(n)$ berorde paling besar $f(n)$) bila terdapat konstanta C dan n_0 sedemikian sehingga

$$T(n) \leq C \cdot f(n)$$

untuk $n \geq n_0$.

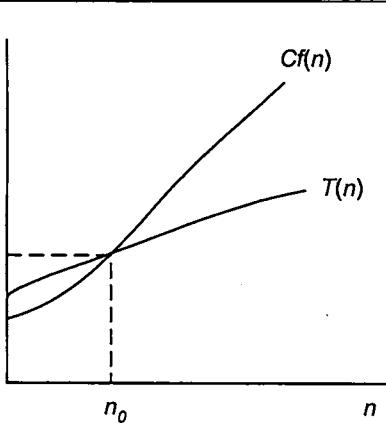
Makna notasi O -besar adalah jika sebuah algoritma mempunyai waktu asimptotik $O(f(n))$, maka jika n dibuat semakin besar, waktu yang dibutuhkannya tidak akan pernah melebihi suatu konstanta C dikali dengan $f(n)$. Jadi, $f(n)$ adalah batas lebih atas (*upper bound*) dari $T(n)$ untuk n yang besar. Kita katakan $T(n)$ berorde paling besar $f(n)$.

Gambar 10.2 memperlihatkan tafsiran geometri “ O besar”. Meskipun $T(n)$ pada mulanya berada di atas $Cf(n)$, tapi setelah $n = n_0$ ia selalu berada di bawah $Cf(n)$. Contoh nyatanya adalah $T(n) = n^2 + 5n$, meskipun pada awalnya kurva berada di atas $2n^2$, tetapi untuk $n \geq 5$

$$n^2 + 5n \leq 2n^2$$

Karena itu kita mengambil $C = 2$ dan $n_0 = 5$ untuk memperlihatkan bahwa

$$n^2 + 5n = O(n^2)$$



Gambar 10.2 Ilustrasi “ O besar”

Dari definisi O -Besar jelas menuliskan $T(n) = O(f(n))$ tidak sama dengan $O(f(n)) = T(n)$. Lagi pula, tidaklah bermakna apa-apa menyatakan $O(f(n)) = T(n)$. Penggunaan simbol “=” tidak menguntungkan karena simbol ini sudah umum menyatakan “kesamaan”. Kebingungan yang timbul dari penggunaan simbol ini dapat dihindari dengan membaca simbol “=” sebagai “adalah” dan bukan “sama dengan” [HOR90].

Untuk menunjukkan bahwa $T(n) = O(f(n))$ kita hanya perlu menemukan pasangan C dan n_0 sedemikian sehingga $T(n) \leq C(f(n))$. Tetapi, perlu diingat bahwa pasangan C dan n_0 yang memenuhi definisi di atas tidak unik. Ada banyak C dan

n_0 yang memenuhi definisi ini. Contoh-contoh berikut memperlihatkan cara memperoleh notasi kompleksitas asimptotik untuk bermacam-macam $T(n)$.

Contoh 10.8

Tunjukkan bahwa $T(n) = 2n^2 + 6n + 1 = O(n^2)$.

Penyelesaian:

Kita mengamati bahwa jika $n \geq 1$ maka $n \leq n^2$ dan $1 \leq n^2$ sehingga

$$2n^2 + 6n + 1 \leq 2n^2 + 6n^2 + n^2 = 9n^2 \text{ untuk semua } n \geq 1.$$

Jadi, kita dapat mengambil $C = 9$ dan $n_0 = 1$ untuk memperlihatkan bahwa

$$T(n) = 2n^2 + 6n + 1 = O(n^2).$$

Perhatikan bahwa $C = 9$ dan $n_0 = 1$ bukan satu-satunya pasangan nilai yang dapat kita gunakan untuk memenuhi Definisi 10.1. Kita juga dapat menyatakan bahwa

$$2n^2 + 6n + 1 \leq 6n^2 \text{ untuk semua } n \geq 2$$

Jadi, kita mengambil $C = 6$ dan $n_0 = 2$ untuk memperlihatkan bahwa

$$T(n) = 2n^2 + 6n + 1 = O(n^2).$$

Alternatif lain adalah dengan mengamati bahwa $6n \leq n^2$ untuk $n \geq 6$. Jadi, kita menyatakan bahwa

$$2n^2 + 6n + 1 = O(n^2)$$

karena

$$2n^2 + 6n + 1 \leq n^2 + n^2 + n^2 = 3n^2 \text{ untuk semua } n \geq 6$$

(di sini $C = 3$ dan $n_0 = 6$). ■

Contoh 10.9

Tunjukkan bahwa $T(n) = 5 = O(1)$.

Penyelesaian:

$5 = O(1)$ karena $5 \leq 6 \cdot 1$ untuk $n \geq 1$. Kita juga dapat memperlihatkan bahwa $5 = O(1)$ karena $5 \leq 10 \cdot 1$ untuk $n \geq 1$ ■

Contoh 10.10

Tunjukkan bahwa $T(n) = 3n + 2 = O(n)$.

Penyelesaian:

$3n + 2 = O(n)$ karena $3n + 2 \leq 3n + 2n = 5n$ untuk semua $n \geq 1$ ($C = 5$ dan $n_0 = 1$). ■

Contoh 10.12

Tunjukkan bahwa kompleksitas waktu algoritma pengurutan seleksi (*selection sort*) pada Contoh 10.6 adalah $T(n) = \frac{n(n-1)}{2} = O(n^2)$.

Penyelesaian:

$\frac{n(n-1)}{2} = O(n^2)$ karena $\frac{n(n-1)}{2} \leq \frac{n^2}{2} + \frac{n^2}{2} = n^2$ untuk semua $n \geq 1$ ($C = 1$ dan $n_0 = 1$). ■

Contoh 10.13

Tunjukkan bahwa $T(n) = 5n^2 = O(n^3)$, tetapi $T(n) = n^3 \neq O(n^2)$.

Penyelesaian:

$5n^2 = O(n^3)$ karena $5n^2 \leq n^3$ untuk semua $n \geq 5$. Tetapi, $T(n) = n^3 \neq O(n^2)$ karena tidak ada konstanta C dan n_0 sedemikian sehingga

$$n^3 \leq Cn^2 \Leftrightarrow n \leq C$$

untuk semua n_0 karena n dapat berupa sembarang bilangan yang besar. ■

Di bawah ini diberikan contoh-contoh yang lain untuk menentukan kompleksitas waktu asimptotik:

1. $T(n) = n + 1024 = O(n)$ $\{ n + 1024 \leq n + 1024n = 1025n \text{ untuk semua } n \geq 1 \}$
2. $T(n) = 1 + 2 + \dots + n = O(n^2)$ $\{ 1 + 2 + \dots + n \leq n + n + \dots + n = n . n = n^2 \text{ untuk } n \geq 1 \}$
3. $T(n) = 10n^2 + 4n + 2 = O(n^2)$ $\{ 10n^2 + 4n + 2 \leq 16n^2 \text{ untuk semua } n \geq 5 \}$
4. $T(n) = n^2 / 10 + 2^n = O(2^n)$ $\{ n^2 / 10 + 2^n \leq 10(2^n) \text{ untuk semua } n \geq 1 \}$
5. $T(n) = 6 \cdot 2^n + n^2 = O(2^n)$ $\{ 6 \cdot 2^n + n^2 \leq 7 \cdot 2^n \text{ untuk semua } n \geq 4 \}$
6. $T(n) = 1^k + 2^k + \dots + n^k = O(n^{k+1})$ $\{ 1^k + 2^k + \dots + n^k \leq n^k + n^k + \dots + n^k = n . n^k = n^{k+1} \text{ untuk } n \geq 1 \}$
7. $T(n) = \log n^3 = 3 \log n = O(\log n)$ $\{ 3 \log n \leq 4 \log n \text{ untuk semua } n \geq 1 \}$
8. $T(n) = 10 \log 3^n = 10n \log 3 = O(n)$ $\{ 10n \log 3 \leq 11n \log 3 \text{ untuk semua } n \geq 1 \}$

9. $T(n) = 5n \log n = O(n \log n)$ { $5n \log n \leq 6n \log n$ untuk semua $n \geq 1$ }
 10. $T(n) = 2n + 3^2 \log n = O(n)$ { $2n + 3^2 \log n \leq 2n + 3n = 5n$ untuk $n \geq 1$, karena $3^2 \log n < n$ untuk $n \geq 1$ }
 11. $T(n) = n = O(n^2)$ { $n \leq 1 \cdot n^2$ untuk semua $n \geq 1$ }
 12. $T(n) = n! = O(n^n)$ { $n! = 1 \cdot 2 \cdot 3 \dots \cdot n \leq n \cdot n \cdot n \dots \cdot n = n^n$ untuk $n \geq 1$ }
 13. $T(n) = \log n! = O(n \log n)$ { $\log n! \leq \log n^n = n \log n$
 atau dengan cara lain:

$$\begin{aligned} \log n! &= \log(n \times (n-1) \times \dots \times 2 \times 1) \\ &= \log n + \log(n-1) + \dots + \log 2 + 1 \leq \log n + \log n + \dots \\ &\quad + \log n + \log n = n \log n \end{aligned}$$
 }
 14. $T(n) = 21 + 1/n = O(n)$ { $21 + 1/n \leq 22n$ untuk semua $n \geq 1$ }
 15. $T(n) = 3n + 2 \neq O(1)$ karena tidak ada C dan n_0 sedemikian sehingga $3n + 2 \leq C \cdot 1$ untuk $n \geq n_0$.
 16. $T(n) = 10n^2 + 4n + 2 \neq O(n)$ karena tidak ada C dan n_0 sedemikian sehingga $10n^2 + 4n + 2 \leq C \cdot n$ untuk $n \geq n_0$.

Polinomial n derajat m dapat digunakan untuk memperkirakan kompleksitas waktu asimptotik. Suku berorde rendah dalam ekspresi $T(n)$ dapat diabaikan dalam menentukan orde keseluruhan. Jadi, $T(n) = 3n^3 + 6n^2 + n + 8 = O(n^3)$, $T(n) = 2n^2 + 6n + 1 = O(n^2)$, $T(n) = \frac{1}{2}n^2 = O(n^2)$. Hal ini dinyatakan dengan teorema berikut [HOR90]:

TEOREMA 10.1. Bila $T(n) = a_m n^m + a_{m-1} n^{m-1} + \dots + a_1 n + a_0$ adalah polinom derajat m maka $T(n) = O(n^m)$.

Bukti:

$$\begin{aligned} T(n) &\leq \sum_{i=0}^m a_i n^i \\ &\leq n^m \sum_{i=0}^m a_i n^{i-m} \\ &\leq n^m \sum_{i=0}^m a_i = Cn^m, \text{ untuk } n \geq 1, \text{ yang dalam hal ini } C = \sum_{i=0}^m a_i. \end{aligned}$$

Oleh karena itu

$$T(n) = O(n^m).$$

Teorema 10.1 menyatakan bahwa suku yang berderajat lebih tinggi mendominasi suku yang lebih rendah. "Mendominasi" diartikan bahwa "laju pertumbuhannya lebih cepat daripada"; $f(n)$ mendominasi $T(n)$ jika $T(n)$ adalah $O(f(n))$. Hal ini selalu terjadi, artinya, selalu ada suku yang dominan untuk n yang besar. Dalam menentukan kompleksitas waktu asimptotik, perhatian kita selalu tertuju pada suku yang dominan itu.

Besaran dominan lainnya adalah:

- eksponensial mendominasi sembarang perpangkatan (yaitu, $y^n > n^p$, $y > 1$),
- perpangkatan mendominasi $\ln n$ (yaitu $n^p > \ln n$),
- semua logaritma tumbuh pada laju yang sama (yaitu " $\log(n) = b\log(n)$ "),
- $n \log n$ tumbuh lebih cepat daripada n tetapi lebih lambat daripada n^2 .

Perhatikan bahwa kita juga dapat menyatakan bahwa $2n^2 + 6n + 1 = O(n^3)$ karena $2n^2 + 6n + 1 \leq 9n^3$ untuk semua $n \geq 1$, atau $2n^2 + 6n + 1 \leq 2n^3$ untuk semua $n \geq 4$. Dengan cara yang sama kita juga dapat menyatakan $2n^2 + 6n + 1 = O(n^4)$, dan seterusnya. Jadi, bila $T(n) = O(f(n))$, dan $g(n)$ adalah fungsi yang nilainya lebih besar dari $f(n)$, maka $T(n) = O(g(n))$. Ini berarti, fungsi $f(n)$ pada notasi $T(n) = O(f(n))$ dapat diganti dengan fungsi lain yang lebih besar [ROS99]. Namun, agar notasi O -besar memiliki makna, maka untuk alasan lebih praktis, fungsi f di dalam notasi $T(n) = O(f(n))$ dipilih fungsi yang sekecil mungkin, biasanya dari fungsi acuan seperti $1, \log n, n, n \log n, n^2, n^3, \dots, 2^n, n!$. Jadi, jika suatu algoritma memiliki kebutuhan waktu $T(n) = 2n^2 + 6n + 1$, kompleksitas waktunya ditulis $O(n^2)$, bukan $O(n^3)$, $O(n^4)$, dan seterusnya.

Teorema O -Besar

TEOREMA 10.2. Misalkan $T_1(n) = O(f(n))$ dan $T_2(n) = O(g(n))$, maka

- (a) (i) $T_1(n) + T_2(n) = O(\max(f(n), g(n)))$
(ii) $T_1(n) + T_2(n) = O(f(n) + g(n))$
- (b) $T_1(n)T_2(n) = O(f(n))O(g(n)) = O(f(n)g(n))$
- (c) $O(cf(n)) = O(f(n))$, c adalah konstanta
- (d) $f(n) = O(f(n))$

Perhatikan di dalam Teorema 10.2 bahwa bahwa $T_1(n) + T_2(n)$ diberikan dalam dua rumus yang berbeda, hal ini karena notasi O Besar adalah mekanisme untuk menemukan batas lebih atas untuk laju pertumbuhan kompleksitas waktu algoritma dan bukan batas lebih atas yang paling kecil [AZM88]. Keduanya digunakan dalam konteks yang berbeda.

Contoh 10.14

Misalkan $T_1(n) = O(n)$, $T_2(n) = O(n^2)$, dan $T_3(n) = O(mn)$, dengan m adalah peubah maka

- (a) $T_1(n) + T_2(n) = O(\max(n, n^2)) = O(n^2)$ (Teorema 10.2(a)(i))
(b) $T_2(n) + T_3(n) = O(n^2 + mn)$ (Teorema 10.2(a)(ii))
(c) $T_1(n)T_2(n) = O(n \cdot n^2) = O(n^3)$ (Teorema 10.2(b)) ■

Contoh 10.15

$$\begin{aligned}O(5n^2) &= O(n^2) && (\text{Teorema 10.2(c)}) \\n^2 &= O(n^2) && (\text{Teorema 10.2d})\end{aligned}$$

Aturan Menentukan Kompleksitas Waktu Asimptotik

Kompleksitas waktu asimptotik suatu algoritma dapat ditentukan dengan salah satu dari 2 cara di bawah ini:

1. Cara I

Jika kompleksitas waktu $T(n)$ dari algoritma sudah dihitung, maka kompleksitas waktu asimptotiknya dapat langsung ditentukan dengan mengambil suku yang mendominasi fungsi T dan menghilangkan koefisiennya. Cara ini bersuaian dengan Teorema 10.2.

Contoh:

- (i) pada algoritma CariElemenTerbesar, $T(n) = n - 1 = O(n)$
(ii) pada algoritma PencarianBeruntun,

$$T_{\min}(n) = 1 = O(1)$$

$$T_{\max}(n) = n = O(n)$$

$T_{\text{avg}}(n) = (n + 1)/2 = O(n)$, atau, dengan cara lain sebagai berikut:

- (iii) pada algoritma PencarianBiner,

$$T_{\min}(n) = 1 = O(1)$$

$$T_{\max}(n) = \log n = O(\log n)$$

- (iv) pada algoritma PengurutanSeleksi,

$$T(n) = \frac{n(n-1)}{2} = O(n^2)$$

(semua notasi O -Besar pada (i), (ii), (iii), dan (iv) di atas didasarkan pada jumlah perbandingan elemen-elemen larik)

(v) pada algoritma Kali,

$$T(n) = 2(n - 2) = O(n) \quad \text{dihitung dari jumlah operasi perkalian}$$

(vi) $T(n) = (n + 2) \log(n^2 + 1) + 5n^2 = O(n^2)$

Penjelasannya adalah sebagai berikut:

$$\begin{aligned} T(n) &= (n + 2) \log(n^2 + 1) + 5n^2 \\ &= f(n)g(n) + h(n), \text{ dengan } f(n) = (n + 2), g(n) = \log(n^2 + 1), \\ &\quad \text{dan } h(n) = 5n^2 \end{aligned}$$

Kita rinci satu per satu:

$$\begin{aligned} \Rightarrow f(n) &= (n + 2) = O(n) \\ \Rightarrow g(n) &= \log(n^2 + 1) = O(\log n), \text{ karena} \\ &\quad \log(n^2 + 1) \leq \log(2n^2) \\ &\quad = \log 2 + \log n^2 \\ &\quad = \log 2 + 2 \log n \leq 3 \log n \text{ untuk } n > 2 \\ \Rightarrow h(n) &= 5n^2 = O(n^2) \end{aligned}$$

maka

$$\begin{aligned} T(n) &= (n + 2) \log(n^2 + 1) + 5n^2 = O(n)O(\log n) + O(n^2) \\ &= O(n \log n) + O(n^2) \quad (\text{Teorema 10.2(b)}) \\ &= O(\max(n \log n, n^2)) \quad (\text{Teorema 10.2(a)}) \\ &= O(n^2) \end{aligned}$$

2. Cara II

Umumnya menghitung kompleksitas waktu untuk kasus terbaik dan kasus rata-rata sangat sulit dilakukan. Hal ini disebabkan informasi tentang probabilitas setiap permutasi data masukan yang berukuran n tidak diketahui atau tidak dapat ditentukan. Karena itu cukup beralasan kalau kita hanya meninjau kompleksitas waktu terburuk saja. Di bawah ini diberikan beberapa aturan untuk menghitung kompleksitas waktu asimptotik untuk kasus terburuk [AZM88]. Di dalam aturan tersebut kita tidak menghitung jumlah pelaksanaan operasi dasar sebagaimana pada perhitungan $T(n)$, tetapi kita menghitung langsung dengan menggunakan notasi O -Besar:

- (a) Pengisian nilai (*assignment*), perbandingan, operasi aritmetika (+, -, *, /, div, mod), *read*, *write*, pengaksesan elemen larik, memilih *field* tertentu dari sebuah *record*, dan pemanggilan fungsi/prosedur membutuhkan waktu $O(1)$.

Contoh: Tinjau potongan algoritma berikut

<u>read</u> (x)	$O(1)$
$x \leftarrow x + 1$	$O(1) + O(1) + O(1) = O(1)$
<u>write</u> (x)	$O(1)$

Kompleksitas waktu asimptotik algoritma = $O(1) + O(1) + O(1) = O(1)$

Penjelasan:

$$\begin{aligned} O(1) + O(1) + O(1) &= O(\max(1,1)) + O(1) && (\text{Teorema 10.2(a)(i)}) \\ &= O(1) + O(1) \\ &= O(\max(1,1)) = O(1) && (\text{Teorema 10.2(a)(i)}) \end{aligned}$$

- (b) if C then S1 else S2 membutuhkan waktu $T_C + \max(T_{S1}, T_{S2})$ yang dalam hal ini T_C , T_{S1} , dan T_{S2} adalah kompleksitas waktu C, S1, dan S2.

Contoh: Tinjau potongan algoritma berikut

<u>read</u> (x)	$O(1)$
<u>if</u> x <u>mod</u> 2 = 0 <u>then</u>	$O(1)$
x \leftarrow x + 1	$O(1)$
<u>write</u> (x)	$O(1)$
<u>else</u>	
<u>write</u> (x)	$O(1)$
<u>endif</u>	

Kompleksitas waktu asimptotik algoritma:

$$\begin{aligned} &= O(1) + O(1) + \max(O(1)+O(1), O(1)) \\ &= O(1) + \max(O(1), O(1)) \\ &= O(1) + O(1) \\ &= O(1) \end{aligned}$$

- (c) Kalang for. Kompleksitas waktu kalang for adalah jumlah pengulangan dikali dengan kompleksitas waktu badan (*body*) kalang.

Contoh: Tinjau potongan algoritma berikut

<u>for</u> i \leftarrow 1 <u>to</u> n <u>do</u>	
jumlah \leftarrow jumlah + a _i	$O(1)$
<u>endfor</u>	

Kompleksitas waktu asimptotik = $n \cdot O(1) = O(n \cdot 1) = O(n)$

atau dengan pendekatan lain:

$$\begin{aligned}\text{Kompleksitas waktu asimptotik} &= n \cdot O(1) \\ &= O(n) O(1) \\ &= O(n)\end{aligned}$$

Teorema 10.2(d)
Teorema 10.2(c)

Contoh: Tinjau potongan algoritma berikut

```
for i ← 1 to n do
    for j ← 1 to n do
        aij ← 0
    endfor
endfor
```

Kalang terdalam mempunyai kompleksitas waktu $O(n)$. Kalang terluar dikerjakan sebanyak n kali, sehingga kompleksitas waktu asimptotik seluruhnya adalah

$$n \cdot O(n) = O(n \cdot n) = O(n^2)$$

Contoh: Tinjau potongan algoritma berikut yang memiliki kalang bersarang dengan dua buah instruksi di dalamnya

```
for i ← 1 to n do
    for j ← 1 to i do
        a ← a + 1
        b ← b - 2
    endfor
endfor
```

$$\text{waktu untuk } a \leftarrow a+1 = O(1)$$

$$\text{waktu untuk } b \leftarrow b - 2 = O(1)$$

$$\text{total waktu untuk badan kalang} = O(1) + O(1) = O(1)$$

$$\text{waktu untuk kalang terdalam} = i \cdot O(1) = O(i) \cdot O(1) = O(i \cdot 1) = O(i)$$

$$\begin{aligned}\text{waktu untuk kalang terluar} &= \sum_{i=1}^n O(i) \\ &= O\left(\sum_{i=1}^n i\right) \quad (\text{Teorema 10.2(a)(ii)}) \\ &= O\left(\frac{n(n+1)}{2}\right) \\ &= O\left(\frac{n^2}{2} + \frac{n}{2}\right)\end{aligned}$$

$$= O\left(\frac{n^2}{2}\right) \quad (\text{Teorema 10.2(a)(i)})$$

$$= O(n^2) \quad (\text{Teorema 10.2(c)})$$

Jadi, kompleksitas waktu algoritma adalah $O(n^2)$.

- (d) while C do S; dan repeat S until C; Untuk kedua buah kalang, kompleksitas waktunya adalah jumlah pengulangan dikali dengan kompleksitas waktu badan C dan S. Masalah yang muncul adalah bila jumlah pengulangan tidak dapat ditentukan karena pengulangan dilakukan bergantung pada kondisi yang harus dipenuhi .

Contoh: Tinjau potongan algoritma berikut

```
i ← 2                                O(1)
while i ≤ n do                    O(1)
    jumlah ← jumlah + ai           O(1)
    i ← i + 1                         O(1)
endwhile
```

Kalang while dieksekusi sebanyak $n - 1$ kali, sehingga kompleksitas waktu asimptotik algoritma adalah

$$\begin{aligned} &= O(1) + (n - 1) \{ O(1) + O(1) + O(1) \} \\ &= O(1) + (n - 1) O(1) \\ &= O(1) + O(n - 1) \\ &= O(1) + O(n) \\ &= O(n) \end{aligned}$$

Jadi, kompleksitas waktu algoritma adalah $O(n)$.

Contoh: Tinjau potongan algoritma berikut yang mempunyai kalang yang tidak dapat ditentukan panjangnya:

```
ketemu ← false
while (p ≠ Nil) and (not ketemu) do
    if p↑.kunci = x then
        ketemu ← true
    else
        p ← p↑.Next
    endif
endwhile
{ p = Nil or ketemu }
```

Di sini, pengulangan akan berhenti bila x yang dicari ditemukan di dalam senarai atau seluruh elemen senarai sudah dibandingkan. Jika jumlah elemen

senarai adalah n , maka kompleksitas waktu terburuknya adalah $O(n)$ -yaitu kasus x tidak ditemukan. Pada kebanyakan kasus, tiap elemen senarai mempunyai peluang yang sama mengandung nilai x . Tetapi, ingatlah kembali bahwa kompleksitas waktu asimptotik menyatakan waktu terpanjang yang dibutuhkan untuk eksekusi algoritma sebagai fungsi dari n . Jadi, kita andaikan x terdapat pada elemen terakhir atau x tidak ditemukan. Dengan kata lain, kompleksitas waktu kalang tersebut adalah $O(n)$.

(e) case A

```
A = a1 : S1
A = a2 : S2
...
A = an : Sn
endcase
```

membutuhkan waktu $\max(T_{S1}, T_{S2}, \dots, T_{Sn})$ yang dalam hal ini $T_{S1}, T_{S2}, \dots, T_{Sn}$ dan T_S adalah kompleksitas waktu $S1, S2, \dots, Sn$

Contoh: Tinjau potongan algoritma berikut

```
read(n)                                O(1)
case n
    n mod 2 = 0 : for i ← 1 to n do
        write(i*i)
    endfor
    x mod 2 = 1 : write('ganjil')
endcase
```

Kompleksitas waktu asimptotik algoritma:

$$\begin{aligned} &= O(1) + \max(O(n), O(1)) \\ &= O(1) + O(n) \\ &= O(n) \end{aligned}$$

- (f) Untuk fungsi/prosedur rekursif, digunakan teknik perhitungan kompleksitas dengan *relasi rekurens*.

Contoh: Tinjau potongan algoritma rekursif berikut

```
funcion faktorial(input n : integer) → integer
    { mengembalikan nilai n!, n tidak negatif }

Algoritma
    if n = 0 then
        return 1
    else
        return n * faktorial(n-1)
    endif
```

Kompleksitas waktu untuk algoritma faktorial rekursif di atas dihitung berdasarkan jumlah operasi perkalian dalam relasi rekurens (*recurrence relation*) berikut:

- Untuk kasus basis, tidak ada operasi perkalian (0),
- Untuk kasus rekurens, kompleksitas waktu diukur dari jumlah perkalian
(1) ditambah kompleksitas waktu untuk faktorial ($n - 1$).

Jadi,

$$T(n) = \begin{cases} 0, & n = 0 \\ 1 + T(n-1), & n > 0 \end{cases}$$

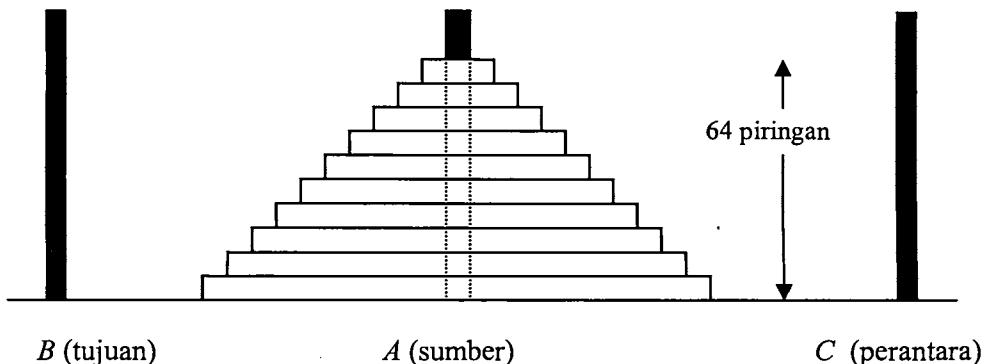
dengan a dan b adalah konstanta. $T(n)$ dapat dipecahkan sebagai berikut:

$$\begin{aligned} T(n) &= 1 + T(n-1) \\ &= 1 + 1 + T(n-2) = 2 + T(n-2) \\ &= 2 + 1 + T(n-3) = 3 + T(n-3) \\ &\vdots \\ &= n + T(0) \\ &= n + 0 \\ &= O(n) \end{aligned}$$

Jadi, kompleksitas algoritma faktorial adalah $O(n)$.

Contoh 10.16

(Menara Hanoi). Contoh ini merupakan legenda klasik pendeta Budha. Di kota Hanoi, Vietnam, terdapat tiga buah tiang tegak setinggi 5 meter dan 64 buah piringan (*disk*) dari berbagai ukuran (Gambar 10.3). Tiap piringan mempunyai lubang di tengahnya yang memungkinkannya untuk dimasukkan ke dalam tiang.



Gambar 10.3 Menara Hanoi

Pada mulanya piringan tersebut tersusun pada sebuah tiang sedemikian rupa sehingga piringan yang di bawah mempunyai ukuran lebih besar daripada ukuran piringan di atasnya. Pendeta Budha memberi pertanyaan kepada muridnya: bagaimana memindahkan seluruh piringan tersebut ke sebuah tiang yang lain; setiap kali hanya satu piringan yang boleh dipindahkan, tetapi tidak boleh ada piringan besar di atas piringan kecil. Tiang yang satu lagi dapat dipakai sebagai tempat peralihan dengan tetap memegang aturan yang telah disebutkan. Menurut legenda pendeta Budha, bila pemindahan seluruh piringan itu berhasil dilakukan, maka dunia akan kiamat!

Berdasarkan aturan yang ditetapkan oleh pendeta Budha, maka kita harus memindahkan piringan paling bawah terlebih dahulu ke tiang B sebagai alas bagi piringan yang lain. Untuk mencapai maksud demikian, berpikirlah secara rekursif: andaikan kita mengangkat 63 piringan teratas dari A ke C , lalu pindahkan piringan paling bawah dari A ke B , lalu angkat 63 piringan dari C ke B .

*angkat 63 piringan dari A ke C
pindahkan 1 piringan terbawah dari A ke B
angkat 63 piringan dari C ke B*

Selanjutnya -dengan tetap berpikir rekursif- pekerjaan mengangkat 63 piringan dari sebuah tiang ke tiang lain dapat dibayangkan sebagai mengangkat 62 piringan antara kedua tiang tersebut, lalu memindahkan piringan terbawah dari sebuah tiang ke tiang lain, begitu seterusnya.

Prosedur rekursif untuk memindahkan n buah piringan dari A ke B adalah:

Hanoi(n, A, B, C)
(a) basis
jika $n = 1$, pindahkan piringan dari A ke B
(b) rekurens
jika $n > 1$,
- Hanoi($n - 1, A, C, B$)
- pindahkan 1 piringan dari A ke B
- Hanoi($n - 1, C, B, A$)

```
procedure Hanoi(input n, A, B, C:integer)
(Memindahkan  $n$  buah piringan (disk) dari  $A$  ke  $B$ ,  $n$  adalah jumlah piringan.
Pada mulanya seluruh piringan berada di tiang  $A$ , piringan terbesar berada
paling bawah. Setelah proses pemindahan, seluruh piringan pindah ke tiang
 $B$ , piringan paling besar berada paling bawah. )
```

Algoritma

```
if  $n = 1$  then
    write('pindahkan piringan dari ', A, ' ke ', B)
else
    Hanoi( $n - 1, A, C, B$ )
    write('pindahkan piringan dari ', A, ' ke ', B)
    Hanoi( $n - 1, C, B, A$ )
endif
```

Algoritma 10.7 Algoritma Menara Hanoi

Kompleksitas waktu dari algoritma Menara Hanoi diukur dari jumlah perpindahan piringan dari satu tiang ke tiang lain. Untuk $n = 1$ piringan (basis), hanya ada satu piringan yang dipindahkan, sedangkan untuk $n > 1$ piringan (rekurens), satu piringan dipindahkan ditambah dengan jumlah piringan yang dipindahkan pada pemanggilan prosedur Hanoi untuk $n - 1$ piringan lain. Kompleksitas waktu algoritma adalah

$$T(n) = \begin{cases} 1 & , n = 1 \\ 1 + 2T(n-1) & , n > 1 \end{cases}$$

Bagian rekurens dipecahkan sebagai berikut:

$$\begin{aligned} T(n) &= 1 + 2T(n-1) \\ &= 1 + 2(1 + 2T(n-2)) = 1 + 2 + 2^2T(n-2) \\ &= 1 + 2 + 2^2(1 + 2T(n-3)) = 1 + 2 + 2^2 + 2^3 T(n-3)) \\ &\vdots \\ &= (1 + 2 + 2^2 + \dots + 2^{n-2}) + 2^{n-1} T(1) \\ &= 1 + 2 + 2^2 + \dots + 2^{n-2} + 2^{n-1} \cdot 1 = 1 + 2 + 2^2 + \dots + 2^{n-2} + 2^{n-1} \\ &= 2^n - 1 = O(2^n) \end{aligned}$$

Jadi, persoalan menara Hanoi mempunyai kompleksitas eksponensial, $O(2^n)$. Perhatikanlah bahwa $T(n) = 2^n - 1$ adalah jumlah seluruh perpindahan piringan dari satu tiang ke tiang lainnya. Jika untuk memindahkan satu piringan dibutuhkan waktu satud etik, maka waktu yang dibutuhkan untuk memindahkan perpindahan seluruh piringan adalah $2^{64} - 1$ detik = 18.446.744.073.709.551.615 atau setara dengan 600.000.000.000 tahun! Karena itu, legenda yang menyatakan bahwa dunia akan kiamat bila orang berhasil memindahkan 64 piringan di menara Hanoi dari tiang *A* ke tiang *B* ada benarnya, karena 600 miliar tahun adalah waktu yang sangat lama, dunia saat ini mungkin sudah tidak mungkin bertahan lagi dalam jangka waktu itu, dan akhirnya hancur (kiamat)! ■

Contoh 10.17

Misalkan kita memiliki dua buah larik *a* dan *b*, masing-masing dengan elemen a_1, a_2, \dots, a_n , dan b_1, b_2, \dots, b_n , yang keduanya berukuran masing-masing n elemen dan setiap larik sudah terurut menaik. Kita ingin membentuk sebuah larik baru, *c*, yang merupakan gabungan dari dua buah larik tersebut. Penggabungan dilakukan dengan cara membandingkan satu elemen pada larik *a* dengan satu elemen pada larik *b*. Jika elemen pada *a* lebih kecil dari elemen pada *b*, maka salin elemen dari *a* ke *c*. Elemen berikutnya pada *a* maju satu elemen, sedangkan elemen *b* tetap. Hal yang sama juga berlaku bila elemen dari *b* lebih kecil dari elemen *a*, maka salin elemen dari *b* ke *c*. Larik *b* maju satu elemen, larik pertama tetap. Dengan cara seperti ini, akan ada larik yang elemennya sudah duluan habis disalin, sedangkan larik yang lain masih tersisa. Elemen larik yang tersisa disalin ke larik *c*.

Contoh ilustrasi penggabungan:

a	b		c
1 13 24	2 15 27	$1 < 2 \rightarrow 1$	1
1 13 24	2 15 27	$2 < 13 \rightarrow 2$	1 2
1 13 24	2 15 27	$13 < 15 \rightarrow 13$	1 2 13
1 13 24	2 15 27	$15 < 24 \rightarrow 15$	1 2 13 15
1 13 24	2 15 27	$24 < 27 \rightarrow 24$	1 2 13 15 24
1 13 24	2 15 27	$27 \rightarrow$	1 2 13 15 24 27

Algoritmanya adalah sebagai berikut:

```
procedure GabungLarikTerurut(input a1, a2, ..., an,
                                b1, b2, ..., bn : integer,
                                output c1, c2, ..., c2n : integer)
{Penggabungan dua buah larik terurut, a dan b, menghasilkan larik baru,
c, yang terurut menaik.
Masukan: a1, a2, ..., an dan b1, b2, ..., bn
Keluaran: c1, c2, ..., c2n
}

Deklarasi
k1, k2, k3 : integer

Algoritma
k1 ← 1
k2 ← 1
k3 ← 1
while (k1 ≤ n) and (k2 ≤ n) do
    if ak1 ≤ bk2 then
        ck3 ← ak1
        k1 ← k1 + 1
    else
        ck3 ← bk2
        k2 ← k2 + 1
    endif
    k3 ← k3 + 1
endwhile
{ k1 > n or k2 > n }

{ salin sisa larik a, jika ada }
while (k1 ≤ n) do
    ck3 ← ak1
    k1 ← k1 + 1
```

```

    k3 ← k3 + 1
endwhile
{ k1 > n }

{ salin sisa larik b, jika ada }
while (k2 ≤ n) do
    ck3 ← bk2
    k2 ← k2 + 1
    k3 ← k3 + 1;
endwhile
{ k2 > n }

```

Algoritma 10.7 Penggabungan dua buah larik terurut

Berapa kompleksitas asimptotik algoritma GabungLarikTerurut di atas?

Penyelesaian:

Algoritma GabungLarikTerurut mempunyai tiga buah kalang *while-do*. Empat buah instruksi pengisian nilai sebelum kalang *while-do* yang pertama mempunyai kompleksitas $O(1)$.

Kita akan menghitung kompleksitas kalang *while-do* yang pertama. Pada kasus terburuk, yaitu pada kasus seluruh elemen larik *a* lebih kecil dari seluruh elemen larik *b* (atau sebaliknya), kalang *while-do* ini akan dijalankan sebanyak n kali. Instruksi *if-then-else* di dalam badan kalang *while-do* mempunyai kompleksitas $O(1)$. Dengan demikian, kompleksitas kalang *while-do* pertama adalah

$$n \cdot O(1) = O(n)$$

Salah satu dari kalang *while-do* kedua atau kalang *while-do* ketiga akan dieksekusi, yaitu untuk menyalin elemen larik yang tersisa. Kita akan menghitung kompleksitas waktu kalang kedua/ketiga. Pada kasus terburuk tersebut di atas, semua elemen larik *a* atau semua elemen larik *b* disalin ke larik *c*. Ini berarti kalang *while-do* kedua/ketiga akan dijalankan sebanyak n kali. Instruksi penyalinan di dalam badan kalang itu membutuhkan waktu $O(1)$. Dengan demikian, kompleksitas kalang *while-do* kedua/ketiga adalah

$$n \cdot O(1) = O(n)$$

Kompleksitas waktu asimptotik seluruhnya adalah

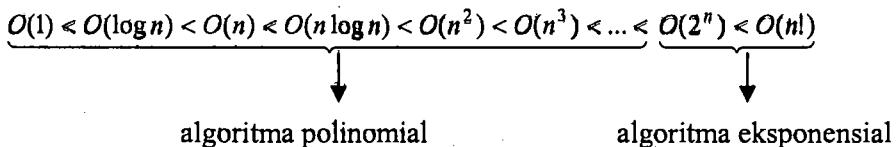
$$O(1) + O(n) + O(n) = O(1) + O(\max(n, n)) = O(\max(1, n)) = O(n)$$

Jadi, kompleksitas asimptotik algoritma GabungLarikTerurut adalah $O(n)$. ■

Pengelompokan Algoritma Berdasarkan Notasi O -Besar

Tiap-tiap algoritma mempunyai kompleksitas waktu asimptotik masing-masing. Kompleksitas waktu asimptotik ini dapat digunakan untuk mengelompokkan algoritma (Tabel 10.2).

Urutan spektrum kompleksitas waktu algoritma adalah :



Tabel 10.2 Kelompok algoritma berdasarkan kompleksitas waktu asimptotiknya

Kelompok Algoritma	Nama
$O(1)$	konstan
$O(\log n)$	logaritmik
$O(n)$	lanjar
$O(n \log n)$	$n \log n$
$O(n^2)$	kuadratik
$O(n^3)$	kubik
$O(2^n)$	eksponensial
$O(n!)$	faktorial

Urutan spektrum kompleksitas waktu di atas ditafsirkan sebagai berikut: jika $O(f(n))$ terletak sebelum $O(g(n))$, maka itu berarti $f(n) \leq g(n)$ untuk semua bilangan bulat n . Jadi, jika algoritma A dan B mempunyai waktu pelaksanaan masing-masing $O(f(n))$ dan $O(g(n))$ dan $O(f(n))$ terletak sebelum $O(g(n))$, maka algoritma A dikatakan lebih *mangkus* daripada algoritma B untuk ukuran masukan yang besar.

Penjelasan masing-masing kelompok algoritma adalah sebagai berikut [SED92]:

- $O(1)$ Kompleksitas $O(1)$ berarti waktu pelaksanaan algoritma adalah tetap, tidak bergantung pada ukuran masukan. Dengan kata lain, jumlah operasi abstrak adalah tetap dan total kebutuhan waktunya dibatasi oleh sebuah konstanta. Bila n dijadikan dua kali semula, misalnya, waktu pelaksanaan algoritmanya hanya bertambah sejumlah konstan. Algoritma yang termasuk kelompok ini adalah algoritma yang kebanyakan instrusinya dilaksanakan satu kali atau paling banyak beberapa kali. Contohnya prosedur tukar di bawah ini:

```

procedure Tukar(input/output a, b :integer)
{ Mempertukarkan nilai a dan b. Setelah pertukaran, a berisi nilai b
  dan b berisi nilai a semula.
  Masukan: a dan b
  Keluaran: a dan b
}

Deklarasi
  temp : integer

Algoritma
  temp ← a
  a ← b
  b ← temp

```

Algoritma 10.8 Pertukaran dua buah nilai

Di sini jumlah operasi penugasan (*assignment*) ada tiga buah dan tiap operasi dilakukan satu kali. Jadi, $T(n) = 3 = O(1)$.

- $O(\log n)$ Kompleksitas waktu logaritmik berarti laju pertumbuhan waktunya berjalan lebih lambat daripada pertumbuhan n . Algoritma yang termasuk kelompok ini adalah algoritma yang memecahkan persoalan besar dengan mentransformasikannya menjadi beberapa persoalan yang lebih kecil yang berukuran sama (misalnya algoritma pencarian_biner). Di sini basis algoritma tidak terlalu penting sebab bila n dinaikkan dua kali semula, misalnya, $\log n$ meningkat sebesar sejumlah konstanta. Fungsi $\log n$ hanya meningkat menjadi dua kali semula jika n dinaikkan sebesar n^2 kali semula (basis dua).
- $O(n)$ Algoritma yang waktu pelaksanaannya lanjut umumnya terdapat pada kasus yang setiap elemen masukannya dikenai proses yang sama, misalnya algoritma pencarian_beruntun. Bila n dijadikan dua kali semula, maka waktu pelaksanaan algoritma juga dua kali semula.
- $O(n \log n)$ Waktu pelaksanaan yang $n \log n$ terdapat pada algoritma yang memecahkan persoalan menjadi beberapa persoalan yang lebih kecil, menyelesaikan tiap persoalan secara independen, dan menggabung solusi masing-masing persoalan. Algoritma yang diselesaikan dengan teknik bagi dan gabung mempunyai kompleksitas asimptotik jenis ini. Bila $n = 1000$, maka $n \log n$ mungkin 20.000. Bila n dijadikan dua kali semula, maka $n \log n$ menjadi dua kali semula (tetapi tidak terlalu banyak)

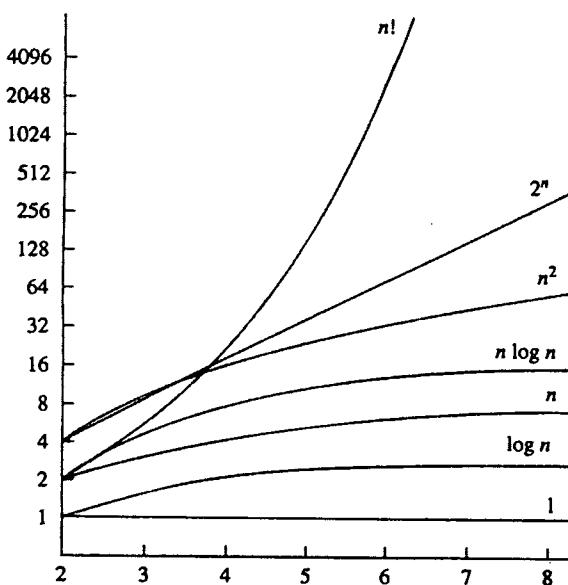
- $O(n^2)$ Algoritma yang waktu pelaksanaannya kuadratik hanya praktis digunakan untuk persoalana yang berukuran kecil. Umumnya algoritma yang termasuk kelompok ini memproses setiap masukan dalam dua buah kalang bersarang, misalnya pada algoritma urut_maks. Bila $n = 1000$, maka waktu pelaksanaan algoritma adalah 1.000.000. Bila n dinaikkan menjadi dua kali semula, maka waktu pelaksanaan algoritma meningkat menjadi empat kali semula.
- $O(n^3)$ Seperti halnya algoritma kuadratik, algoritma kubik memproses setiap masukan dalam tiga buah kalang bersarang, misalnya algoritma perkalian matriks. Bila $n = 100$, maka waktu pelaksanaan algoritma adalah 1.000.000. Bila n dinaikkan menjadi dua kali semula, waktu pelaksanaan algoritma meningkat menjadi delapan kali semula.
- $O(2^n)$ Algoritma yang tergolong kelompok ini mencari solusi persoalan secara "*brute force*", misalnya pada algoritma mencari sirkuit Hamilton (lihat Bab Graf). Bila $n = 20$, waktu pelaksanaan algoritma adalah 1.000.000. Bila n dijadikan dua kali semula, waktu pelaksanaan menjadi kuadrat kali semula!
- $O(n!)$ Seperti halnya pada algoritma eksponensial, algoritma jenis ini memproses setiap masukan dan menghubungkannya dengan $n - 1$ masukan lainnya, misalnya algoritma Persoalan Pedagang Keliling (*Travelling Salesperson Problem* - lihat bab 9). Bila $n = 5$, maka waktu pelaksanaan algoritma adalah 120. Bila n dijadikan dua kali semula, maka waktu pelaksanaan algoritma menjadi faktorial dari $2n$.

Enam buah notasi orde yang pertama, $O(1)$ sampai $O(n^3)$, adalah orde algoritma yang penting. Di sini kebutuhan waktunya dibatasi oleh polinomial, dan algoritmanya dinamakan **algoritma polinomial**. Algoritma yang kompleksitas waktu asimptotiknya $O(2^n)$ dinamakan **algoritma eksponensial**, karena bila n diperbesar, kebutuhan waktunya meningkat dengan tajam. Algoritma yang kebutuhan waktunya $O(2^n)$ tidak dapat digolongkan sebagai algoritma polinomial karena tidak ada bilangan bulat m sedemikian sehingga polinom n^m membatasi 2^n , atau $2^n \neq O(n^m)$ untuk sembarang bilangan bulat m . Algoritma yang kebutuhan waktunya $O(n!)$ juga digolongkan sebagai algoritma eksponensial karena pertumbuhan waktunya mempunyai kemiripan dengan fungsi 2^n .

Tabel 10.3 dan Gambar 10.4 di bawah ini memperlihatkan bagaimana kebutuhan waktu untuk ketujuh fungsi kompleksitas waktu tumbuh [HOR90] (diandaikan konstanta atau koefisien di depan persamaannya sama dengan satu).

Tabel 10.3 Nilai masing-masing fungsi untuk setiap bermacam-macam nilai n

$\log n$	n	$n \log n$	n^2	n^3	2^n	$n!$
0	1	0	1	1	2	1
1	2	2	4	8	4	2
2	4	8	16	64	16	24
3	9	24	64	512	256	362880
4	16	64	256	4096	65536	20922789888000
5	32	160	1024	32768	4294967296	(terlalu besar untuk ditulis)



Gambar 10.4 Laju pertumbuhan fungsi kompleksitas waktu [ROS99].

Tabel 10.4 memperlihatkan waktu eksekusi algoritma yang kompleksitas waktunya adalah fungsi-fungsi acuan, yaitu $f(n) = n$, $f(n) = \log n$, $f(n) = n \log n$, $f(n) = n^2$, $f(n) = n^3$, dan $f(n) = 2^n$ [NEA96]. Asumsi yang digunakan adalah satu kali operasi dasar pada setiap algoritma membutuhkan waktu 1 nanodetik (10^{-9} detik). Tabel tersebut memperlihatkan hasil yang mengejutkan. Orang mungkin berharap bahwa selama algoritmanya bukan algoritma eksponensial, ia merupakan algoritma yang memadai. Tetapi, algoritma kuadratik membutuhkan waktu 31,7 tahun untuk memproses masukan sebanyak 1 bilion, sedangkan algoritma $O(n \log n)$ membutuhkan waktu hanya 29,9 detik untuk memproses masukan sebanyak itu. Sebuah algoritma sebaiknya $O(n \log n)$ atau kita mengsumsikan bahwa algoritma dapat memproses masukan yang berukuran sangat besar dalam waktu yang dapat ditoleransi. Hal ini tidaklah berarti bahwa

algoritma yang kompleksitas waktunya lebih tinggi tidak berguna. Algoritma dengan kebutuhan waktu kuadratik, kubik, dan yang lebih tinggi sering berguna memproses masukan pada banyak aplikasi [NEA96].

Tabel 10.4 Waktui eksekusi algoritma dengan berbagai macam kompleksitas waktu [NEA96]

n	$f(n) = \log n$	$f(n) = n$	$f(n) = n \log n$	$f(n) = n^2$	$f(n) = n^3$	$f(n) = 2^n$
10	0,003 μ s	0,01 μ s	0,033 μ s	0,1 μ s	1 μ s	1 μ s
20	0,004 μ s	0,02 μ s	0,086 μ s	0,4 μ s	8 μ s	1 ms
30	0,005 μ s	0,03 μ s	0,147 μ s	0,9 μ s	27 μ s	1 s
40	0,005 μ s	0,04 μ s	0,213 μ s	1,6 μ s	64 μ s	18,3 menit
50	0,006 μ s	0,05 μ s	0,282 μ s	2,5 μ s	125 μ s	13 hari
10^2	0,007 μ s	0,10 μ s	0,664 μ s	10 μ s	1 ms	$4 \cdot 10^{13}$ thn
10^3	0,010 μ s	1,00 μ s	9,966 μ s	1 ms	1 s	
10^4	0,013 μ s	10 μ s	130 μ s	100 ms	16,7 menit	
10^5	0,017 μ s	0,10 ms	1,67 ms	10 s	11,6 hari	
10^6	0,020 μ s	1 ms	19,93 ms	16,7 menit	31,7 tahun	
10^7	0,023 μ s	0,01 s	0,23 s	1,16 hari	31,709 th	
10^8	0,027 μ s	0,10 s	2,66 s	115,7 hari	$1,17 \cdot 10^7$ th	
10^9	0,030 μ s	1 s	29,90 s	31,7 tahun		

Keterangan:

$$1 \mu\text{s} = 10^{-6} \text{ second}$$

$$1 \text{ ms} = 10^{-3} \text{ second}$$

Sekarang, tinjau secara khusus algoritma yang kebutuhan waktunya eksponensial. Algoritma eksponensial tumbuh sangat cepat dengan bertambahnya nilai n . Algoritma eksponensial hanya bagus untuk nilai n yang sangat kecil, bahkan bila kita menurunkan nilai koefisien di depannya, pengurangan kebutuhan waktunya tidak mengalami banyak kemajuan. Agar lebih jelas mengapa perubahan koefisien atau konstanta, daripada perubahan orde, menghasilkan sedikit perbaikan dalam kebutuhan waktu pelaksanaan program, kita dapat melihatnya pada contoh di berikut ini.

Misalkan dua buah algoritma adalah $O(n^2 \cdot 2^n)$ dan $O(n \cdot 2^n)$. Kedua algoritma adalah eksponensial, tetapi fungsi kompleksitas algoritma pertama memiliki faktor tambahan n dibandingkan dengan yang kedua. Koefisien di depan persamaannya diandaikan sama dengan satu. Untuk bermacam-macam n , nilai setiap fungsi ditunjukkan pada Tabel 10.5. Dengan menggunakan andaian bahwa satu buah operasi memakan waktu 0,001 detik, kita menemukan bahwa untuk $n = 30$ kebutuhan waktu untuk algoritma pertama 8,9 jam dan algoritma kedua 11 hari. Meskipun faktor tambahan n membuat perbedaan yang berarti, ciri eksponensial mendominasi perhitungan dan menyiratkan bahwa kedua algoritma membutuhkan waktu yang lama. Jika kita dapat mempercepat algoritma kedua dengan faktor $1/10$, sehingga menjadi $(1/10) n^2 \cdot 2^n$, maka untuk ukuran masukan $n > 10$

algoritma pertama masih lebih cepat. Selanjutnya, untuk $n = 30$ waktu yang dibutuhkan algoritma pertama masih lebih besar dari 24 jam. Kesimpulan yang kita buat dari contoh ini adalah: algoritma eksponensial membutuhkan waktu yang besar. Pengubahan koefisien maupun pemakaian komputer yang lebih cepat tidak memberi perbaikan waktu komputasi yang berarti. Jalan alternatifnya ialah menurunkan algoritma dengan orde yang lebih baik [HOR77].

Tabel 10.5 Nilai fungsi $n^2 \cdot 2^n$ dan $n \cdot 2^n$ untuk setiap bermacam-macam nilai n

n	$n^2 \cdot 2^n$	$n \cdot 2^n$
5	160	800
10	10240	102400
15	491520	7372800
20	20971520	419430400
30	3.2×10^{10}	9.6×10^{11}

Beberapa catatan lain yang ditambahkan di sini adalah:

1. Perlu diingat sekali lagi, bahwa kompleksitas waktu asimptotik $-O(f(n))$ - hanyalah ukuran *kasar* kebutuhan waktu pelaksanaan sebuah algoritma untuk n yang besar, jadi bukan ukuran waktu sebenarnya. Notasi O -Besar mengekspresikan berapa waktu yang dibutuhkan untuk menyelesaikan masalah dengan meningkatnya ukuran masukan. Jika kita ingin mengukur kebutuhan waktu yang lebih presisi, kita harus juga membandingkan koefisien dan konstanta dalam persamaan $T(n)$ -nya. Sebagai misal, meskipun $T(n) = 1000n \log n$ dan $T(n) = (n \log n)/10$ sama-sama berorde $O(n \log n)$, tetapi $T(n) = (n \log n)/10$ lebih cepat untuk n yang besar. Contoh lainnya, andaikan algoritma A membutuhkan waktu $T(n) = 300n$ dan algoritma B membutuhkan waktu $T(n) = 5n^2$. Untuk ukuran masukan berukuran $n = 5$, algoritma A membutuhkan 1500 satuan waktu dan algoritma B membutuhkan 125 satuan waktu. Jadi, untuk masukan yang berukuran kecil algoritma B lebih cepat daripada algoritma A . Tentu saja algoritma A lebih cepat daripada algoritma B untuk n yang besar.
2. Kompleksitas waktu asimptotik dapat digunakan untuk membandingkan dua buah algoritma. Misalkan ada sebuah persoalan yang sama diselesaikan dengan dua buah algoritma yang berbeda. Algoritma I mempunyai kompleksitas waktu $O(n)$, dan algoritma II mempunyai kompleksitas $O(n^2)$. Manakah yang lebih cepat, algoritma I atau algoritma II untuk n yang besar? Mudah dilihat bahwa untuk n yang cukup besar, waktu untuk algoritma II tumbuh lebih cepat daripada waktu algoritma I. Sedangkan untuk menghitung kebutuhan waktu yang sebenarnya, kita juga harus melihat konstanta yang mendahuluinya.

Contoh (a): (i) Algoritma I $\rightarrow T(n) = 2n = O(n)$
 Algoritma II $\rightarrow T(n) = n^2 = O(n^2)$

Terlihat, untuk $n > 2$ algoritma I yang berorde $O(n)$ paling cepat waktu pelaksanaannya dibandingkan dengan algoritma II yang berorde $O(n^2)$. Jadi, algoritma I cepat untuk $n > 2$.

Contoh (b): (ii) Aalgoritma I $\rightarrow T(n) = 10^4 n = O(n)$
 Aalgoritma II $\rightarrow T(n) = n^2 = O(n^2)$

Terlihat, untuk $n < 10^4$ algoritma II paling cepat. Tetapi untuk $n > 10^4$ algoritma I yang paling cepat.

3. Sebuah masalah yang mempunyai algoritma dengan kompleksitas polinomial pada kasus-terburuk dianggap mempunyai algoritma yang “bagus”; artinya masalah tersebut mempunyai algoritma yang mangkus, dengan catatan polinomial tersebut berderajat rendah. Jika polinomnya berderajat tinggi, waktu yang dibutuhkan untuk mengeksekusi algoritma tersebut panjang. Untunglah pada kebanyakan kasus, fungsi polinomnya mempunyai derajat yang rendah.
4. Suatu masalah dikatakan *tractable* (mudah dari segi komputasi) jika ia dapat diselesaikan dengan algoritma yang memiliki kompleksitas polinomial kasus terburuk (artinya dengan algoritma yang mangkus), karena algoritma akan menghasilkan solusi dalam waktu yang lebih pendek [ROS99]. Sebaliknya, sebuah masalah dikatakan *intractable* (sukar dari segi komputasi) jika tidak ada algoritma yang mangkus untuk menyelesaiakannya. Untuk menunjukkan bahwa suatu masalah *tractable* kita cukup menunjukkan ada algoritma yang mangkus untuk menyelesaikan masalah itu. Sebagai contoh, masalah menentukan nilai maksimum, masalah pencarian, masalah pengurutan, adalah masalah yang *tractable* karena algoritmanya mempunyai kompleksitas waktu polinom untuk kasus terburuk. Untuk menunjukkan bahwa suatu algoritma *intractable*, kita harus menunjukkan bahwa tidak ada algoritma yang mangkus untuk menyelesaiakannya. Masalah TSP (*Travelling Salesperson Problem*) – lihat Bab Graf- adalah contoh masalah yang belum dapat diselesaikan dengan algoritma polinomial, karena itu *intractable*. Namun orang belum mampu membuktikan bahwa algoritma yang mangkus untuk menyelesaikan masalah itu tidak mungkin ada. Algoritma yang ada untuk menyelesaikan masalah *intractable*, seperti TSP itu, mempunyai kompleksitas eksponensial.
5. Masalah yang sama sekali tidak memiliki algoritma untuk memecahkannya disebut **masalah tak-terselesaikan** (*unsolved problem*). Sebagai contoh, masalah penghentian (*halting problem*) adalah masalah tak-terselesaikan. Masalah penghentian berbunyi: jika diberikan program dan sejumlah masukan, apakah program tersebut berhenti pada akhirnya [JOH90]?

6. Kebanyakan masalah yang dapat dipecahkan dipercaya tidak memiliki algoritma penyelesaian dalam kompleksitas waktu polinomial untuk kasus terburuk, karena itu dianggap *intractable*. Tetapi, jika solusi masalah tersebut ditemukan, maka solusinya dapat diperiksa dalam waktu polinomial. Masalah yang solusinya dapat diperiksa dalam waktu polinomial dikatakan termasuk ke dalam **kelas NP** (*non-deterministic polynomial*). Masalah yang *tractable* termasuk ke dalam **kelas P** (*polynomial*). Jenis kelas masalah lain adalah kelas **NP-lengkap** (*NP-complete*). Kelas masalah *NP-lengkap* memiliki sifat bahwa jika ada sembarang masalah di dalam kelas ini dapat dipecahkan dalam waktu polinomial, berarti semua masalah di dalam kelas tersebut dapat dipecahkan dalam waktu polinomial. Atau, jika kita dapat membuktikan bahwa salah satu dari masalah di dalam kelas itu *intractable*, berarti kita telah membuktikan bahwa semua masalah di dalam kelas tersebut *intractable*. Meskipun banyak penelitian telah dilakukan, tidak ada algoritma dalam waktu polinomial yang dapat memecahkan masalah di dalam kelas *NP-lengkap*. Secara umum diterima, meskipun tidak terbukti, bahwa tidak ada masalah di dalam kelas *NP-lengkap* yang dapat dipecahkan dalam waktu polinomial [ROS99].
7. Meskipun algoritma eksponensial secara asimptotik lebih buruk daripada algoritma polinomial, tetapi untuk n yang kecil algoritma eksponensial menunjukkan kinerja yang lebih baik. Contohnya, $2^n \leq 100n^2$ untuk $n = 1, 2, \dots, 14$. Jadi, untuk n yang kecil lebih tepat menggunakan algoritma eksponensial.
9. Penelitian untuk menemukan algoritma dengan kompleksitas waktu yang kecil masih terus berlangsung sampai saat ini, dan ini merupakan tantangan bagi para ilmuwan komputer, terutama para analis algoritma. Penemuan yang cukup bersejarah adalah algoritma **Transformasi Fourier Cepat** (*Fast Fourier Transform*) -atau TFC- yang ditulis oleh J.W Cooley dan J.W Tukey pada tahun 1965. Sebelumnya, algoritma Transformasi Fourier Farik (*discrete*) -atau TFF- yang biasa berorde $O(n^2)$, tetapi TFC memperkecilnya menjadi $O(n \log n)$. Dengan algoritma TFF, waktu komputasi transformasi Fourier berkurang cukup tajam. Transformasi Fourier banyak digunakan pada bidang pengolahan sinyal.

Notasi Omega-Besar dan Theta-Besar

Notasi *O*-Besar menyediakan batas lebih atas (*upper bound*) untuk fungsi $T(n)$, tetapi tidak memberikan batas lebih bawah (*lower bound*). Untuk itu, kita mendefinisikan batas lebih bawah (*lower bound*) untuk kompleksitas waktu, yang dilambangkan dengan *Omega-Besar* (*Big- Ω*).

Definisi Ω -Besar adalah:

DEFINISI 10.2. $T(n) = \Omega(g(n))$ (dibaca “ $T(n)$ adalah omega ($f(n)$), yang artinya $T(n)$ berorde paling kecil $g(n)$ ”) bila terdapat konstanta C dan n_0 sedemikian sehingga

$$T(n) \geq C \cdot f(n)$$

untuk $n \geq n_0$.

Kita juga mendefinisikan Θ -Besar,

DEFINISI 10.3. $T(n) = \Theta(h(n))$ (dibaca “ $T(n)$ adalah theta ($h(n)$)” yang artinya $T(n)$ berorde sama dengan $h(n)$ jika $T(n) = O(h(n))$ dan $T(n) = \Omega(g(n))$).

Contoh 10.18

Tentukan notasi Ω dan Θ untuk $T(n) = 2n^2 + 6n + 1$.

Penyelesaian:

Karena $2n^2 + 6n + 1 \geq 2n^2$ untuk $n \geq 1$, maka dengan mengambil $C = 2$ kita memperoleh

$$2n^2 + 6n + 1 = \Omega(n^2)$$

Karena $2n^2 + 6n + 1 = O(n^2)$ dan $2n^2 + 6n + 1 = \Omega(n^2)$, maka $2n^2 + 6n + 1 = \Theta(n^2)$. ■

Contoh 10.19

Tentukan notasi notasi O , Ω dan Θ untuk $T(n) = 5n^3 + 6n^2 \log n$.

Penyelesaian:

Karena $0 \leq 6n^2 \log n \leq 6n^3$, maka $5n^3 + 6n^2 \log n \leq 11n^3$ untuk $n \geq 1$. Dengan mengambil $C = 11$, maka

$$5n^3 + 6n^2 \log n = O(n^3)$$

Karena $5n^3 + 6n^2 \log n \geq 5n^3$ untuk $n \geq 1$, maka dengan mengambil $C = 5$ kita memperoleh

$$5n^3 + 6n^2 \log n = \Omega(n^3)$$

Karena $5n^3 + 6n^2 \log n = O(n^3)$ dan $5n^3 + 6n^2 \log n = \Omega(n^3)$, maka $5n^3 + 6n^2 \log n = \Theta(n^3)$. ■

Contoh 10.19

Tentukan notasi notasi O , Ω dan Θ untuk $T(n) = 1 + 2 + \dots + n$.

Penyelesaian:

$1 + 2 + \dots + n = O(n^2)$ karena $1 + 2 + \dots + n \leq n + n + \dots + n = n^2$ untuk $n \geq 1$.

$1 + 2 + \dots + n = \Omega(n)$ karena $1 + 2 + \dots + n \leq 1 + 1 + \dots + 1 = n$ untuk $n \geq 1$.

Kita tidak dapat menurunkan notasi Θ -Besar untuk $1 + 2 + \dots + n$ karena batas bawah n tidak sama dengan batas atas n^2 . Satu cara untuk mendapatkan batas bawah yang sama dengan batas atas adalah dengan menghilangkan setengah pertama dari suku-suku deret. Dengan menjumlahkan hanya suku-suku yang lebih besar dari $\lceil n/2 \rceil$, kita mendapatkan

$$\begin{aligned} 1 + 2 + \dots + n &\geq \lceil n/2 \rceil + \dots + (n-1) + n \\ &\geq \lceil n/2 \rceil + \dots + \lceil n/2 \rceil + \lceil n/2 \rceil \\ &= \lceil (n+1)/2 \rceil \lceil n/2 \rceil \\ &\geq (n/2)(n/2) \\ &= n^2/4 \end{aligned}$$

Kita menyimpulkan bahwa

$$1 + 2 + \dots + n = \Omega(n^2)$$

Oleh karena itu,

$$1 + 2 + \dots + n = \Theta(n^2)$$

Sebuah fakta yang berguna untuk menentukan orde kompleksitas adalah dari suku tertinggi di dalam polinomial. Sebagai contoh, bila $T(n) = 3n^4 + 6n^3 + 18n + 2$, maka $T(n)$ adalah berorde n^4 (yaitu $O(n^4)$, $\Omega(n^4)$, dan $\Theta(n^4)$). Ini dikatakan dengan teorema berikut:

TEOREMA 10.3. Bila $T(n) = a_m n^m + a_{m-1} n^{m-1} + \dots + a_1 n + a_0$ adalah polinom derajat m maka $T(n)$ adalah berorde n^m .

Contoh 10.20

Tentukan notasi notasi O , Ω dan Θ untuk algoritma pengurutan seleksi (*selection sort*).

Penyelesaian:

Kompleksitas waktu algoritma pengurutan seleksi sudah dihitung pada Contoh 10.6, yaitu $T(n) = \frac{n(n-1)}{2} = n^2/2 - n/2$. Menurut Teorema 10.3, algoritma ini berorde n^2 . Jadi, kompleksitas asimptotik algoritma pengurutan seleksi adalah $O(n^2)$, $\Omega(n^2)$, dan $\Theta(n^2)$. ■

10.6 Ragam Soal dan Penyelesaian

Contoh 10.21

Di bawah ini adalah algoritma (dalam notasi Pascal-like) untuk menguji apakah dua buah matriks, A dan B , yang masing-masing berukuran $n \times n$, sama.

```
function samaMatriks(A, B : matriks; n : integer) → boolean
  ( true jika A dan B sama; sebaliknya false jika A ≠ B )

Deklarasi
  i, j : integer

Algoritma:
  for i ← 1 to n do
    for j ← 1 to n do
      if Ai,j ≠ Bi,j then
        return false
      endif
    endfor
  endfor
  return true
```

- (a) Apa kasus terbaik dan terburuk untuk algoritma di atas?
(b) Tentukan kompleksitas waktu terbaik dan terburuk dalam notasi O .

Penyelesaian:

- (a) Kasus terbaik adalah jika elemen-elemen pertama dari kedua buah matriks, yaitu A_{11} dan B_{11} tidak sama sehingga operasi perbandingan elemen-elemen matriks cukup dilakukan sekali saja (fungsi mengembalikan nilai *false*).

Kasus terburuk adalah jika kedua matriks sama, yaitu seluruh elemen matriks A dan B pada posisi yang bersesuaian sama (A_{ij} dan B_{ij} untuk semua i dan j).

- (b) Operasi dasar yang dihitung adalah operasi perbandingan elemen-elemen kedua buah matriks. Kompleksitas waktu terbaik adalah $O(1)$ karena hanya ada satu kali operasi perbandingan elemen. Pada kasus terburuk, seluruh elemen matriks dibandingkan sehingga jumlah operasi perbandingan elemen matriks adalah sebanyak $T(n) = n \cdot n = n^2$ kali. Jadi, kompleksitas waktu untuk kasus terburuk adalah $O(n^2)$. ■

Contoh 10.22

Berapa kali instruksi *assignment* pada potongan program dalam notasi Bahasa Pascal di bawah ini dieksekusi? Tentukan juga notasi O-besar.

```
for i := 1 to n do
  for j := 1 to n do
    for k := 1 to j do
      x := x + 1;
```

Penyelesaian:

Kalang terluar (kalang i) dieksekusi n kali. Untuk setiap nilai i , kalang terluar kedua (kalang j) dieksekusi n kali. Untuk setiap nilai j , kalang terdalam (kalang k) dieksekusi j kali, $j = 1, 2, 3, \dots, n$, sehingga jumlah pengulangan seluruhnya adalah

$$T(n) = n(1 + 2 + 3 + \dots + n) = n^2(n+1)/2$$

dalam notasi O -besar:

$$T(n) = n^2(n+1)/2 \cdot O(1) = O(n^3)$$

Contoh 10.23

Untuk soal (a) dan (b) berikut, tentukan C , $f(n)$, n_0 , dan notasi O -besar sedemikian sehingga $T(n) = O(f(n))$ jika $T(n) \leq C \cdot f(n)$ untuk semua $n \geq n_0$:

- (a) $T(n) = 2 + 4 + 6 + \dots + 2n$
 (b) $T(n) = (n + 1)(n + 3)/(n + 2)$

Penyelesaian:

$$\begin{aligned} \text{(a)} \quad 2 + 4 + 6 + \dots + 2n &= 2(1 + 2 + 3 + \dots + n) \\ &\leq 2(n + n + n + \dots + n) = 2(n^2) \text{ untuk } n \geq 1 \end{aligned}$$

sehingga $C = 2$, $f(n) = n^2$, $n_0 = 1$, didapat $T(n) = O(n^2)$

$$\begin{aligned} \text{(b)} \quad (n + 1)(n + 3)/(n + 2) &= (n^2 + n + 3)/(n + 2) \\ &\leq n + n = 2n \text{ untuk } n \geq 2 \end{aligned}$$

sehingga $C = 2$, $f(n) = n$, $n_0 = 2$, didapat $T(n) = O(n)$

atau dengan pendekatan lain:

perhatikan $(n + 1)/(n + 2)$ akan mendekati 1 untuk n yang besar
 jadi $(n + 1)(n + 3)/(n + 2)$ akan tumbuh sebagaimana $(n + 3)$ tumbuh
 kompleksitas $(n + 3) = O(n)$ karena $n + 3 \leq n + 3n = 4n$ untuk $n \geq 1$

sehingga $C = 4$, $f(n) = n$, $n_0 = 1$, dan didapat kompleksitas asimptotik yang sama, $T(n) = O(n)$.

Contoh 10.24

[AZM88] Tentukan kompleksitas waktu potongan algoritma di bawah ini dalam notasi O -Besar.

```

ukuran ← m
i ← 1
while i < n do
    i ← i + 1
    ProcA(i)      { prosedur ProcA membutuhkan waktu O(n) }
    if ukuran > 0 then
        s ← sebuah nilai di dalam rentang 1 .. ukuran
        ukuran ← ukuran - s
  
```

```

for j  $\leftarrow$  1 to s do
    ProcB(s)           { prosedur ProcB membutuhkan waktu  $O(n)$  }
    endfor
endif
endwhile

```

Penyelesaian:

Kalang *while* dieksekusi sebanyak n kali, karena i dimulai dari 1 dan nilai i selalu bertambah 1 di dalam kalang. Jadi, ProcA dieksekusi sebanyak n kali, sehingga total kebutuhan waktu untuk ProcA adalah $(n - 1) \times O(n)$. Kita tidak dapat menentukan jumlah pengulangan untuk kalang *for* (di dalam kalang *while*) sebagai fungsi dari i . Tetapi, jika s_i adalah nilai s di dalam kalang while untuk suatu nilai i , maka jumlah pengulangan untuk ProcB adalah sebanyak $\sum_i s_i$. Jumlah nilai s_i adalah $\sum_i s_i = m$, karena ukuran diinisialisasi dengan m dan pada setiap pengulangan kalang *while* ukuran dikurangi dengan s_i . Total kebutuhan waktu untuk ProcB adalah $m \times O(n)$. Instruksi yang lain seperti *assignment*, perbandingan ($i < n$ dan ukuran > 0), dan sebagainya membutuhkan waktu $O(1)$. Dengan demikian, kebutuhan waktu potongan algoritma di atas adalah:

$$(n - 1) \times O(n) + m \times O(n) + O(1) = O(n^2) + O(mn) + O(1) = O(n^2 + mn)$$

Soal Latihan

- Untuk $T(n) = 2 + 4 + 8 + 16 + \dots + 2^n$, tentukan C , $f(n)$, n_0 , dan notasi O -besar sedemikian sehingga $T(n) = O(f(n))$ jika $T(n) \leq C \cdot f(n)$ untuk semua $n \geq n_0$.
- Untuk tiap fungsi berikut, tentukan $f(n)$ sedemikian fungsi tersebut adalah $O(f(n))$:
 - $17n^3 + 1.7n^2 + n$
 - $2n^2 + n \log n$
 - $3n + n \log n$
 - $2^n + n^9$
 - $(n + \sqrt{n})(\log n + n + 3)$
- Tentukan kompleksitas waktu asimptotik kode program berikut:

```
for k ← 1 to n do
    for i ← 1 to n do
        for j ← 1 to n do
            wij ← wij or wik and wkj
        endfor
    endfor
endfor
```

- Bandingkan fungsi n^2 dan $2^n/4$ untuk bermacam-macam nilai n . Tentukan kapan fungsi kedua lebih besar daripada fungsi pertama?
- Perlihatkan bahwa $T(n) = 1^k + 2^k + \dots + n^k = O(n^{k+1})$
- Temukan $f(n)$, C , dan n_0 sedemikian sehingga $T(n)$ di bawah ini adalah $O(f(n))$. Untuk fungsi f di dalam $O(f(n))$, gunakan fungsi f yang sederhana dengan pangkat sekecil mungkin:
 - $T(n) = 5n^3 + (\log n)^4$
 - $T(n) = (n^4 + 3 \log n)/(n^4 + 2)$
 - $T(n) = n \log(n^2 + 1) + n^2 \log n$
- Selesaikan tiga soal di bawah ini:
 - Perlihatkan bahwa $n^2 + 5n + 13$ adalah $O(n^3)$ tetapi n^3 bukan $O(n^2 + 5n + 13)$
 - Perlihatkan bahwa 2^n adalah $O(3^n)$ tetapi 3^n bukan $O(2^n)$
- Perlihatkan bahwa $n! = O(n^n)$

10. Perlihatkan bahwa kesamaan berikut tidak benar:
- $10n^2 + 9 = O(n)$
 - $n^3 2^n + 6n^2 3^n = O(n^3 2^n)$
11. Tulislah notasi Tetha-Besar untuk setiap fungsi di bawah ini:
- $T(n) = 6n^3 + 12n^2 + 1$
 - $T(n) = 3n^2 + 2n \log n$
12. Berapa kali pernyataan $x \leftarrow x + 1$ pada algoritma di bawah ini dilaksanakan? Nyatakan kompleksitas waktu asimptotiknya dalam notasi O -Besar, Θ -Besar, dan Ω -Besar.
- ```
i ← 2
while i < n do
 i ← i * i
 x ← x + 1
end
```
  - ```
for i ← 1 to 2n do
    x ← x + 1
endfor
```
 - ```
i ← 1
while i < 2n do
 x ← x + 1
 i ← i + 2
end
```
  - ```
for i ← 1 to 2n do
    for j ← 1 to n do
        x ← x + 1
    endfor
endfor
```
 - ```
i ← n
while i ≥ 1 do
 x ← x + 1
 i ← i div 2
endwhile
```
  - ```
for i ← 1 to n do
    for j ← 1 to i div 2 do
        x ← x + 1
    endfor
endfor
```
13. Tulislah algoritma untuk menentukan nilai maksimum sekaligus nilai minimum larik *integer*. Ukuran larik adalah n elemen. Berapa kompleksitas waktunya? Berapa kompleksitas waktu asimptotiknya dalam notasi O -Besar, Θ -Besar, dan Ω -Besar.
14. Tulislah algoritma untuk menjumlahkan dua buah matriks yang masing-masing berukuran $n \times n$. Berapa kompleksitas waktunya? Berapa kompleksitas waktu asimptotiknya dalam notasi O -Besar, Θ -Besar, dan Ω -Besar?

15. Tulislah algoritma untuk menyalin (*copy*) sebuah larik ke larik yang lain. Ukuran larik adalah n elemen. Berapa kompleksitas waktunya? Berapa kompleksitas waktu asimptotiknya dalam notasi O -Besar, Θ -Besar, dan Ω -Besar.

16. Pecahkan relasi rekurens berikut:

$$T(n) = \begin{cases} a, & n = 0 \\ b + nT(n-1), & n > 0 \end{cases}$$

17. Menghitung perpangkatan a^n , $a \in R$ dan n adalah bilangan bulat (asumsi: n adalah perpangkatan dari 2, atau $n = 2^k$), dapat dilakukan dengan dua buah algoritma di bawah ini (dalam notasi Bahasa Pascal):

(i) Algoritma pertama (iteratif)

$$a^n = a \cdot a \cdot a \dots a$$

(sebanyak n kali)

(ii) Algoritma kedua (rekursif)

$$a^n = 1 \quad \text{jika } n = 0$$

$$= a^{n/2} \cdot a^{n/2} \quad \text{jika } n > 0 \text{ dan } n \text{ genap}$$

$$= a \cdot a^{n/2} \cdot a^{n/2} \quad \text{jika } n > 0 \text{ dan } n \text{ ganjil}$$

```
function p1(a:real;n:integer):real;
{ menghitung a^n =a*a*a* ...*a }
var
  k : integer;
  p : real;
begin
  p:=a;
  for k:=2 to n do
    p:=p*a;
  {endfor}
  p1:=p;
end;
```

```
function p2(a:real;n:integer):real;
{menghitung a^n=a^(n/2)*a^(n/2) }
begin
  if n=0 then
    p2:=1
  else
    if odd(n) then
      p2:=sqr(p2(a,n div 2))*a;
    else
      2:=sqr(p2(a,n div 2))
    {endif}
  end;
```

Keterangan: `sqr` adalah fungsi kuadrat

- (a) Hitung kompleksitas waktu $T(n)$ dan waktu asimptotik $O(f(n))$ masing-masing algoritma pertama dan kedua berdasarkan jumlah operasi perkalian.
 (b) Algoritma manakah yang lebih mangkus/cepat untuk n yang besar?

18. Diberikan algoritma pengurutan *bubble-sort* seperti berikut ini:

```

procedure BubbleSort(input/output a1, a2, ..., an; integer)
{ Mengurut tabel integer TabInt[1..n] dengan metode pengurutan bubble-
sort
  Masukan: a1, a2, ..., an
  Keluaran: a1, a2, ..., an (terurut menaik)
}

```

Deklarasi

```

k : integer { indeks untuk traversal tabel }
pass : integer { tahapan pengurutan }
temp : integer { peubah bantu untuk pertukaran elemen tabel }

```

Algoritma

```

for pass ← 1 to n - 1 do
  for k ← n downto pass + 1 do
    if ak < ak-1 then
      { pertukarkan ak dengan ak-1 }
      temp ← ak
      ak ← ak-1
      ak-1←temp
    endif
  endfor
endfor

```

- (a) Hitung berapa jumlah operasi perbandingan elemen-elemen tabel.
- (b) Berapa kali maksimum operasi pertukaran elemen-elemen tabel dilakukan?
- (c) Hitung kembali kompleksitas waktu asimptotik algoritma pengurutan *bubble-sort*.

19. Sepotong algoritma disajikan di bawah ini:

```

for i←1 to n do
  for j←1 to i do
    for k←1 to j do
      x←x+1
    endfor
  endfor
endfor

```

- (a) Jika $T(n)$ dihitung dari operasi penjumlahan pada pernyataan $x \leftarrow x + 1$, tentukan $T(n)$.
- (b) Nyatakan $T(n)$ dalam notasi O -besar, Ω -besar, dan Θ -besar.

20. Pertanyaan yang sama dengan soal nomor 16 untuk potongan algoritma berikut ini:

```

j ← n
while j ≥ 1 do
  for i ← 1 to j do
    x ← x + 1
  endfor
  j ← j div 2
endwhile

```

21. Algoritma di bawah ini menghitung nilai polinom untuk $x = t$:

$$p(x) = a_0 + a_1x + a_2x^2 + \dots + a_nx^n$$

```
function p(input x:real)→real
{ Mengembalikan nilai p(x)}
```

Deklarasi

j, k : integer
jumlah, suku : real

Algoritma

```
jumlah ← a0
for j ← 1 to n do
    { hitung ajxj }
    suku ← aj
    for k ← 1 to j do
        suku ← suku * x
    endfor
    jumlah ← jumlah + suku
endfor
return jumlah
```

Hitunglah berapa operasi perkalian dan berapa operasi penjumlahan yang dilakukan oleh algoritma di atas? Jumlahkan kedua hitungan tersebut, lalu tentukan juga kompleksitas waktu asimptotik algoritma tersebut dalam notasi O-Besar.

Algoritma mengevaluasipolinom yang lebih baik dapat dibuat dengan metode Horner berikut:

$$p(x) = a_0 + x(a_1 + x(a_2 + x(a_3 + \dots + x(a_{n-1} + a_nx))))\dots))$$

```
function p2(input x : real) → real
{ Mengembalikan nilai p(x) dengan metode Horner}
```

Deklarasi

k : integer
 b_1, b_2, \dots, b_n : real

Algoritma

```
 $b_n \leftarrow a_n$ 
for k ← n - 1 downto 0 do
     $b_k \leftarrow a_k + b_{k+1} * x$ 
endfor
return  $b_0$ 
```

Hitunglah berapa operasi perkalian dan berapa operasi penjumlahan yang dilakukan oleh algoritma di atas? Jumlahkan kedua hitungan tersebut, lalu tentukan juga kompleksitas waktu asimptotik algoritma tersebut dalam notasi O -Besar. Manakah yang terbaik, algoritma p atau $p2$?

Daftar Pustaka¹

- [AND79] Anderson, Robert B., *Proving Programs Correct*, John Wiley & Sons, 1979.
- [BRA88] Brassad, Gilles & Paul Bratley, *Algorithmics, Theory and Practice*, Prentice Hall, 1988
- [AZM88] Azmoodeh, Manoochehr, *Abstract Data Types and Algorithms*, Macmillan Education, 1988
- [DEO74] Deo, Narshing, *Graph Theory with Applications To Engineering and Computer Science*, Prentice-Hall International, 1974
- [DOE85] Doerr, Alan & Kenneth Levasseur, *Applied Discrete Structures for Computer Science*, SRA Associates, 1985
- [DUL94] Dulimarta, Hans Sudiana, *Catatan Kuliah Teori Graph*, Teknik Informatika ITB, 1994
- [GIB85] Gibbons, Alan, *Algorithmic Graph Theory*, Cambridge University Press, 1985

¹ Disusun menurut urutan abjad nama kedua dari pengarang buku

- [HOR76] Horowitz, Ellis & Sartaj Sahni, *Fundamental of Data Structures*, Pitman Publishing Limited, 1976.
- [HOR78] Horowitz, Ellis & Sartaj Sahni, *Fundamental of Computer Algorithms*, Pitman Publishing Limited, 1978.
- [JOH97] Johnsonbaugh, Richard, *Discrete Mathematics*, 4th, Pentice Hall, 1997
- [LIU85] Liu, C.L, *Element of Discrete Mathematics*, McGraw-Hill, Inc, 1985.
- [MAN82] Mano, M. Moris, *Computer System Architecture 2nd*, Prentice-Hall International, 1982
- [NEA96] Neapolitan, Richard E. & Kumarss Naimipour, *Foundations of Algorithms*, D. C. Heath and Company, 1996.
- [ROS99] Rosen, Kenneth H., *Discrete Mathematics and Its Applications*, 4th, McGraw-Hill International 1999.
- [RHE94] Rhee, Man Young, *Cryptography and Secure Communications*, McGraw-Hill, 1994.
- [SAN93] Santoso, Judhi S. *Catatan Kuliah Teori Graph dan Aplikasinya*, Teknik Informatika ITB, 1993.
- [SCH96] Schneier, Bruce, *Apiled Cryptography 2nd*, John Wiley & Sons, 1996
- [SED92] Sedgewick, Robert, *Algorithms in C++*, Addison-Wesley Publishing Company, 1992.
- [LIP92] Lipschutz, Seymour & Marc Lars Lipson, *2000 Solved Problems in Discrete Mathematics*, McGraw-Hill, 1992.