Catatan UAS

IF1220 Matematika Diskrit

Naufarrel Zhafif Abhista
13523149
Z. Nayaka Athadiansyah
13523094

Contents

Chapter 1

Teori Bilangan

1.1 Definisi

- Teori bilangan adalah cabang matematika murni yang ditujukan untuk mempelajari bilangan bulat (*integer*) atau fungsi bernilai bilangan bulat.
- Bilangan bulat (*integer*) adalah bilangan yang tidak mempunyai pecahan desimal.

1.2 Sifat Pembagian Bilangan Bulat

Definisi 1.1: Keterbagian

Misalkan a dan b bilangan bulat, $a \neq 0$. a habis membagi b (a *divides* b) jika terdapat bilangan bulat c sedemikian sehingga $b = ac$. Dapat disimbolkan dengan,

$$a|b$$

yang berlaku ketika $b = ac$, $c \in \mathbb{Z}$ dan $a \neq 0$ (Dibaca a membagi b).

1.3 Teorema Euklidean dan PBB

1.3.1 Teorema 1 Euklidean

Teorema 1.1

Misalkan m dan n bilangan bulat, $n > 0$. Jika m dibagi dengan n maka hasil pembagiannya adalah q (*quotient*) dan sisanya r (*remainder*), sedemikian sehingga,

$$m = nq + r$$

dengan $0 \leq r < n$.

1.3.2 PBB(Pembagi Bersama Terbesar)

Definisi 1.2: Pembagi Bersama Terbesar

Misalkan a dan b bilangan bulat tidak nol. PBB/FPB/GCD dari a dan b adalah bilangan terbesar d sedemikian sehingga $d|a$ dan $d|b$. Sehingga, dapat dinyatakan,

$$PBB(a, b) = d$$

Teorema 1.2

Misalkan m dan n bilangan bulat, dengan $n > 0$, sedemikian sehingga,

$$m = nq + r, \ \ 0 \leq r < n$$

maka,

$$PBB(m, n) = PBB(n, r)$$

Contoh 1.1

Misalkan $m = 60, n = 18$. Maka, dapat dituliskan,

$$60 = 18 \cdot 3 + 6$$

Dari Teorema 1.2,

$$PBB(60, 18) = PBB(18, 6) = 6$$

1.3.3 Algoritma Euklidean

Algoritma Euklidean adalah algoritma yang bertujuan mencari PBB dari dua bilangan bulat. Ditemukan oleh Euclides.

Misalkan m dan n adalah bilangan bulat tak negatif dengan $m \leq n$. Misalkan $r_0 = m$ dan $r_1 = n$. Lakukan pembagian secara berturut-turut, diperoleh,

$$\begin{aligned}
r_0 &= r_1 q_1 + r_2 && 0 \leq r_2 < r_1 \\
r_1 &= r_2 q_2 + r_3 && 0 \leq r_3 < r_2 \\
&\vdots \\
r_{n-2} &= r_{n-1} q_{n-1} + r_n && 0 \leq r_n < r_{n-1} \\
r_{n-1} &= r_n q_n + 0
\end{aligned}$$

Menurut Teorema 1.2,

$$PBB(r_0, r_1) = PBB(r_1, r_2) = PBB(r_2, r_3) = \cdots = PBB(r_{n-1}, r_n) = PBB(r_n, 0) = r_n$$

Jadi, PBB dari m dan n adalah **sisa pembagian terakhir sebelum** 0 dari urutan pembagian tersebut. Bila kita konversi ke algoritma, dengan $m \geq n \geq 0$, berikut algoritma yang didapat.

```
while  n ≠ 0
    r ← m  mod n
    m ← n
    n ← r
→ m
```

Contoh 1.2: Penggunaan Algoritma Euklidean

Tentukan PBB dari 312 dan 70!
Jawab:

$$\begin{aligned}
312 &= 70 \cdot 4 + 32 \\
70 &= 32 \cdot 2 + 6 \\
32 &= 6 \cdot 5 + 2 \\
6 &= 2 \cdot 3 + 0
\end{aligned}$$

Sisa pembagian terakhir sebelum 0 adalah 2, maka $PBB(312, 70) = 2$

1.3.4 Kombinasi Linear

$PBB(a, b)$ dapat dinyatakan dalam suatu kombinasi linear a dan b dengan koefisien terkait.

Teorema 1.3

Misalkan a dan b bilangan bulat positif, maka terdapat bilangan bulat m dan n sedemikian sehingga $PBB(a, b) = ma + nb$.

Contoh 1.3: Kombinasi linear

Telah diketahui $PBB(312, 70) = 2$, nyatakan dalam bentuk kombinasi linear!
Jawab:

$$312 = 70 \cdot 4 + 32 \tag{1.1}$$
$$70 = 32 \cdot 2 + 6 \tag{1.2}$$
$$32 = 6 \cdot 5 + 2 \tag{1.3}$$
$$6 = 2 \cdot 3 + 0 \tag{1.4}$$

Susun (1.2) dan (1.3) menjadi,

$$2 = 32 - 6 \cdot 5 \tag{1.5}$$
$$6 = 70 - 32 \cdot 2 \tag{1.6}$$

Substitusi (1.6) ke (1.5),

$$2 = 32 - (70 - 32 \cdot 2) \cdot 5 = 11 \cdot 32 - 5 \cdot 70 \tag{1.7}$$

Susun (1.1) menjadi,

$$32 = 312 - 70 \cdot 4 \tag{1.8}$$

Substitusi (1.8) ke (1.7),

$$2 = 11 \cdot (312 - 70 \cdot 4) - 5 \cdot 70 = 11 \cdot 312 - 49 \cdot 70 \tag{1.9}$$

Jadi,

$$PBB(312, 70) = 11 \cdot 312 - 49 \cdot 70$$

1.3.5 Relatif Prima

- Dua buah bilangan bulat a dan b dikatakan relatif prima jika $PBB(a, b) = 1$.
- Sebagai contoh, 20 dan 3 relatif prima karena $PBB(20, 3) = 1$.
- Dalam bentuk kombinasi linear, jika dua bilangan a dan b relatif prima, maka terdapat m dan n, sedemikian sehingga,

$$ma + nb = 1$$

1.4 Aritmatika Modulo

1.4.1 Modulo

Definisi 1.3: Modulo

Misalkan a dan m bilangan bulat ($m > 0$). Operasi $a \mod m$ (dibaca "a modulo m") memberikan sisa jika a dibagi dengan m. Dinotasikan:

$$a \mod m = r$$

sedemikian sehingga $a = mq + r$, dengan $0 \leq r < m$. m disebut modulus dan hasil aritmatika modulo m terletak pada himpunan $\{0, 1, \ldots, m - 1\}$.

Contoh 1.4: Beberapa Modulo

- $23 \mod 5 = 3$
- $27 \mod 3 = 0$
- $0 \mod 5 = 0$
- $(-28) \mod 14 = 0$
- $(-6) \mod 4 \neq -2$ (salah!)
- $(-6) \mod 4 = 2$ (benar)

Penjelasan:
Cara menghasilkan modulo, seperti definisi, substitusikan angka pada $a \mod m = r$ ke $a = mq + r$. Ambil contoh $(-6) \mod 4$, kita peroleh $-6 = 4(-2) + r$ (Ingat bahwa $0 \leq r < m$), sehingga $r = 2$. Jadi, $(-6) \mod 8 = 2$. Cara yang sama berlaku untuk operasi modulo yang lain.

1.4.2 Kekongruenan Modulo

Misalkan $38 \mod 5 = 3$ dan $13 \mod 5 = 3$, maka dikatakan,

$$38 \equiv 13 \pmod 5$$

Definisi 1.4: Kekongruenan Modulo

Misalkan a dan b bilangan bulat dan $m > 0$, maka $a \equiv b \pmod m$ jika dan hanya jika $m|(a - b)$.
Jika a tidak kongruen dengan b dalam modulus m, maka kita tulis $a \not\equiv b \pmod m$.

1.4.3 Balikan Modulo

- Di dalam aritmetika bilangan riil, balikan sebuah bilangan yang tidak nol adalah bentuk pecahannya sedemikian sehingga hasil perkalian keduanya sama dengan 1.
- Dengan kata lain, jika a adalah sebuah bilangan tidak-nol, maka balikannya adalah $\frac{1}{a}$ sedemikian sehingga $a \cdot \frac{1}{a} = 1$. Balikan a dilambangkan dengan a^{-1}.
- Contoh: Balikan 4 adalah 1/4, sebab $4 \cdot \frac{1}{4} = 1$.
- Di dalam aritmetika modulo, balikan modulo sebuah bilangan bulat lebih sukar dihitung.

Definisi 1.5: Invers Modulo

Diberikan sebuah bilangan bulat $a \pmod m$. Balikan a dapat dihitung jika a dan m relatif prima dan $m > 1$. Balikan dari $a \pmod m$ adalah x sedemikian sehingga,

$$xa \equiv 1 \pmod m$$

Jadi, $a^{-1} \pmod m = x$.

Bukti: a dan m relatif prima, sehingga $PBB(a, m) = 1$. Dari sini, kita dapat ubah ke dalam bentuk kombinasi linear, berbentuk

$$xa + ym = 1$$

$$xa + ym \equiv 1 \pmod m$$

Karena $ym \equiv 0 \pmod 0$,

$$xa \equiv 1 \pmod m$$

Bukti ini juga menunjukkan bahwa invers dari modulo dapat dicari dengan kombinasi linear. Koefisien dari a adalah invers dari $a \pmod m$.

Contoh 1.5

Tentukan invers dari $4 \pmod 9$!
Jawab: $PBB(4, 9) = 1$, maka inversnya ada. Dari Algoritma Euclid,

$$9 = 2 \cdot 4 + 1 \tag{1.10}$$
$$4 = 4 \cdot 1 \tag{1.11}$$
$$1 = -2 \cdot 4 + 9 \cdot 1 \tag{1.12}$$

Yang mana bentuk ini sama dengan

$$-2 \cdot 4 \equiv 1 \pmod 9$$

Sehingga invers dari $4 \pmod 9$ adalah,

$$\therefore 4^{-1} \pmod 9 = -2 \pmod 9$$

Atau dapat dituliskan,

$$\therefore x = -2 + 9k,\ k \in \mathbb{Z}$$

1.5 Kongruen Linear

1.5.1 Kekongruenan Linear

- Kekongruenan linier (*linear congruence*) berbentuk:

  $$ax \equiv b \pmod m$$

  Untuk $m > 0, a$ dan b sembarang bilangan bulat, dan x variabel bilangan bulat.

- Dipecahkan dengan,

  $$ax = b + km \implies x = \frac{b + km}{a}$$

  Cobakan untuk $k \in Z$ dan ambil yang menghasilkan x bilangan bulat.

Contoh 1.6: Coba Sendiri!

Tentukan solusi dari $4x \equiv 3 \pmod 9$!

- Cara lain: Kalikan persamaan modulo dengan invers dari modulo terkait.

Contoh 1.7: Jawaban

Tentukan solusi dari $4x \equiv 3 \pmod 9$!
Jawab: Carilah invers dari $4 \pmod 9$. Dengan sedikit manipulasi, diperoleh,

$$1 = -2 \cdot 4 + 1 \cdot 9 \implies 4^{-1} \pmod 9 = -2$$

Kalikan $4x \equiv 3 \pmod 9$ dengan -2, diperoleh,

$$-8x \equiv -6 \pmod 9$$

Karena $-8 \equiv 1 \pmod 9$, maka $x \equiv -6 \pmod 9$, atau dapat dituliskan,

$$\therefore x = -6 + 9k,\ k \in \mathbb{Z}$$

Solusi lain yang kongruen dengan $-6 \pmod 9$ juga benar (contoh: $x = 3 + 9k$).

1.5.2 Sistem Kekongruenan Linear

Sistem kekongruenan linier adalah sistem yang terdiri dari lebih dari satu kekongruenan linear.

$$\begin{aligned} x &= a_1 \pmod{m_1} \\ x &= a_2 \pmod{m_2} \\ &\vdots \\ x &= a_n \pmod{m_n} \end{aligned}$$

Contoh 1.8

Sebuah bilangan bulat jika dibagi dengan 3 bersisa 2 dan jika ia dibagi dengan 5 bersisa 3. Berapakah bilangan bulat tersebut?
Jawab:

$$\begin{aligned} x \equiv 2 \pmod{3} &\iff x = 2 + 3k_1 \\ x \equiv 3 \pmod{5} &\iff x = 3 + 5k_2 \end{aligned}$$

Substitusikan $x = 2 + 3k_1$ ke $x = 3 + 5k_2$, kita dapatkan,

$$\begin{aligned} 2 + 3k_1 &\equiv 3 \pmod{5} \\ 3k_1 &\equiv 1 \pmod{5} \\ \therefore k_1 &\equiv 2 \pmod{5} \iff k_1 = 2 + 5k_2 \end{aligned}$$

Substitusikan $k_1 = 2 + 5k_2$ ke $x = 2 + 3k_1$, diperoleh,

$$\begin{aligned} x &= 2 + 3(2 + 5k_2) \\ x &= 8 + 15k_2 \iff x \equiv 8 \pmod{15} \end{aligned}$$

Semua nilai x yang kongruen dengan $8 \pmod{15}$ juga merupakan solusi valid.

1.5.3 *Chinese Remainder Theorem*

Pada abad pertama Masehi, 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.

Teorema 1.4: Chinese Remainder Theorem (CRT)

Misalkan $m_1, m_2, \cdots, m_n$ adalah bilangan bulat positif sedemikian sehingga $PBB(m_i, m_j) = 1$ untuk $i \neq j$. Maka, sistem kekongruenan linier,

$$\begin{aligned} x &= a_1 \pmod{m_1} \\ x &= a_2 \pmod{m_2} \\ &\vdots \\ x &= a_n \pmod{m_n} \end{aligned}$$

mempunyai sebuah solusi unik dalam modulus $m = m_1 \cdot m_2 \cdots m_n$. Yaitu, terdapat solusi x dengan $0 \leq x < m$ dan semua solusi lain yang kongruen dalam modulus m dengan solusi ini.

Contoh 1.9: Penyelesaian Masalah Sun Tse (Cara Biasa)

Selesaikan permasalahan Sun Tse!
Jawab: Susun SKL terlebih dahulu.

$$x = 3 \pmod 5 \tag{1.13}$$
$$x = 5 \pmod 7 \tag{1.14}$$
$$x = 7 \pmod{11} \tag{1.15}$$

SKL ini sama dengan,

$$x = 3 + 5k_1 \tag{1.16}$$
$$x = 5 + 7k_2 \tag{1.17}$$
$$x = 7 + 11k_3 \tag{1.18}$$

Substitusikan (1.16) ke (1.14), diperoleh

$$3 + 5k_1 = 5 \pmod 7 \implies 5k_1 = 2 \pmod 7 \implies k_1 \equiv 6 \pmod 7 \implies k_1 = 6 + 7k_2$$

Substitusikan hasil sebelumnya ke (1.16),

$$x = 3 + 5(6 + 7k_2) = 33 + 35k_2$$

Substitusikan hasil ini ke (1.15), diperoleh

$$33 + 35k_2 = 7 \pmod{11} \implies 35k_2 = -26 \pmod{11}$$

$$\implies k_2 \equiv 9 \pmod{11} \implies k_2 = 9 + 11k_3$$

Substitusikan hasil sebelumnya ke (1.17),

$$x = 33 + 35(9 + 11k_3) = 348 + 385k_3 \implies x \equiv 348 \pmod{385}$$

Perhatikan bahwa 348 adalah solusinya. Perhatikan juga bahwa $385 = 5 \cdot 7 \cdot 11$, artinya sesuai dengan CRT.

Teorema 1.5

Memanfaatkan CRT dan juga SKL, kita peroleh modulus, kita dapat membentuk solusi unik dari SKL, yakni,

$$x = a_1 M_1 y_1 + a_2 M_2 y_2 + \cdots + a_n M_n y_n$$

dengan M_k adalah perkalian semua modulus kecuali m_k dan y_k adalah balikan M_k dalam modulus m_k. Kita juga memperoleh modulus umum $m = m_1 \cdot m_2 \cdots m_n$
Bila dikombinasikan, diperoleh solusi unik $x \pmod m$.

Contoh 1.10: Penyelesaian Masalah

Selesaikan permasalahan Sun Tse!
Jawab: Susun SKL terlebih dahulu.

$$x = 3 \pmod 5$$
$$x = 5 \pmod 7$$
$$x = 7 \pmod{11}$$

Modulus umum: $m = m_1 \cdot m_2 \cdot m_3 = 5 \cdot 7 \cdot 11 = 385$
Kemudian, solusi unik dari sistem kekongruenan linear:

$$x = a_1 M_1 y_1 + a_2 M_2 y_2 + a_3 M_3 y_3$$

$M_1 = 7 \cdot 11 = 77;\ M_2 = 5 \cdot 11 = 55;\ M_3 = 5 \cdot 7 = 35$
$y_1 = 3;\ y_2 = 6;\ y_3 = 6$

Substitusikan, maka solusi uniknya:

$$\begin{aligned} x &= 3 \cdot 77 \cdot 3 + 5 \cdot 55 \cdot 6 + 7 \cdot 35 \cdot 6 \\ &= 3813 \\ \therefore x &\equiv 348 \pmod{385} \end{aligned}$$

1.6 Bilangan Prima

- Bilangan bulat positif $p > 1$ disebut bilangan prima jika pembaginya hanya 1 dan p.
- Seluruh bilangan prima adalah ganjil, kecuali 2.
- Bilangan tidak prima (selain 1) adalah bilangan komposit.

Teorema 1.6: ***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 1.7: ***Fermat's Little Theorem***

Jika p adalah bilangan prima dan a adalah bilangan bulat yang tidak habis dibagi dengan p, yaitu $PBB(a, p) = 1$, maka:

$$a^{p-1} \equiv 1 \pmod{p}$$

- Menurut teorema Fermat di atas, jika p adalah bilangan prima, maka $a^{p-1} \equiv 1 \pmod{p}$
- Tetapi, jika p bukan bilangan prima, maka $a^{p-1} \not\equiv 1 \pmod{p}$

Contoh 1.11: Tes *Fermat's Little Theorem*

Tes apakah 17 dan 21 bilangan prima atau bukan dengan Teorema Fermat!
Jawab: Pilih $a = 2$, karena $PBB(2, 17) = 1$ dan $PBB(2, 21) = 1$

1. $2^{17-1} = 65536 \equiv 1 \pmod{17} \implies$ Prima!
2. $2^{21-1} = 1048576 \not\equiv 1 \pmod{21} \implies$ Bukan prima.

- Kelemahan: Terdapat bilangan komposit n yang memenuhi $a^{n-1} \equiv 1 \pmod{n}$. Bilangan komposit n ini dinamakan prima semu (*pseudoprime*). Contohnya, 341.
- Kemunculan prima semu cukup jarang. Dari 10^{10} bilangan bulat, hanya 14.884 buah bilangan prima semu yang ada.

1.7 Aplikasi Teori Bilangan

1.7.1 ISBN *(International Standard Book Code)* 10 Angka

- Kode ISBN terdiri dari 10 angka, biasanya dikelompokkan dengan spasi atau garis, misalnya 0-3015-4561-9.
- Terdiri dari empat bagian kode:
  - Kode Identifikasi Negara
  - Kode Penerbit
  - Kode Unik Untuk Suatu Buku
  - Karakter Uji (disimbolkan dengan satu angka atau huruf X, artinya 10)
- Misalkan 10 angka ISBN dinyatakan dengan $x_1, x_2, \cdots, x_{10}$.

Figure 1.1: Contoh ISBN 8-1752-5766-0

- Kode ISBN ini memenuhi kekongruenan:

$$\sum_{i=1}^{10} ix_i \equiv 0 \pmod{11}$$

- Karakter uji r dihitung dengan:

$$\sum_{i=1}^{9} ix_i \mod 11 = r$$

Contoh 1.12

Mari kita coba olah ISBN di *figure 1.1*.

$$\begin{aligned}\sum_{i=1}^{10} ix_i &= 1\cdot 8 + 2\cdot 1 + 3\cdot 7 + 4\cdot 5 + 5\cdot 2 + 6\cdot 5 + 7\cdot 7 + 8\cdot 6 + 9\cdot 6 + 10\cdot 0 \\ &= 242 \equiv 0 \pmod{11}\end{aligned}$$

Kemudian, untuk karakter ujinya,

$$\begin{aligned}\sum_{i=1}^{10} ix_i &= 1\cdot 8 + 2\cdot 1 + 3\cdot 7 + 4\cdot 5 + 5\cdot 2 + 6\cdot 5 + 7\cdot 7 + 8\cdot 6 + 9\cdot 6 \\ &= 242\end{aligned}$$

Jadi, karakter ujinya adalah $242 \mod 11 = 0$

1.7.2 ISBN *(International Standard Book Code)* 13 Angka

- Sejak 1 Januari 2007, kode ISBN menjadi 13 angka (13 digit) mengikuti system EAN (European Number System).
- Bentuknya mirip dengan ISBN 10 angka, hanya saja ditambah 978 di depan kode ISBN.
- Kekongruenan ISBN 13 angka dihitung dengan cara yang berbeda dengan ISBN 10 angka.
- Misalkan 13 angka ISBN dinyatakan dengan $x_1, x_2, \cdots, x_{13}$. Kekongruenannya digambarkan dalam bentuk:

$$(x_1 + 3x_2 + x_3 + 3x_4 + x_5 + 3x_6 + x_7 + 3x_8 + x_9 + 3x_{10} + x_{11} + 3x_{12} + x_{13}) \equiv 0 \pmod{10}$$

  Atau dalam notasi sumasi:

$$\left(\sum_{i=1}^{7} x_{2i-1} + 3\sum_{i=1}^{6} x_{2i}\right) \equiv 0 \pmod{10}$$

- Karakter uji r diperoleh dengan cara:

$$r = 10 - ((x_1 + 3x_2 + x_3 + 3x_4 + x_5 + 3x_6 + x_7 + 3x_8 + x_9 + 3x_{10} + x_{11} + 3x_{12}) \mod 10)$$

  Jika $r = 10$, karakter yang dipakai bukan X, melainkan 0.

Figure 1.2: Contoh ISBN 13 Angka.

1.7.3 Fungsi *Hash*

- Singkatnya, meng-alamat-kan data dalam memori.
- Diambil suatu kunci K yang kemudian dijadikan alamat data tersebut. Cara peng-alamat-an:

$$h(K) = K \mod m$$

dengan definisi K adalah kunci, m adalah tempat yang tersedia, dan $h(K)$ adalah alamat dari K.

Contoh 1.13

$m = 11$ mempunyai sel-sel memori yang diberi indeks 0 sampai 10. Akan disimpan data record yang masing-masing mempunyai kunci $15, 558, 32, 132, 102$, dan 5.

$$\begin{aligned} h(15) &= 15 \mod 11 = 4 \\ h(558) &= 558 \mod 11 = 8 \\ h(32) &= 32 \mod 11 = 10 \\ h(132) &= 132 \mod 11 = 0 \\ h(102) &= 102 \mod 11 = 3 \\ h(5) &= 5 \mod 11 = 5 \end{aligned}$$

K	132			102	15	5			558		32
Alamat	0	1	2	3	4	5	6	7	8	9	10

- Pada contoh di atas, bagaimana bila dimasukkan $K = 74$?
- Diperoleh $h(74) = 74 \mod 11 = 8$
- Terjadi **Kolisi**, solusinya adalah dimasukkan ke tempat selanjutnya yang kosong. Dalam kasus ini, maka masukkan 74 ke indeks ke-9.
- Fungsi *hash* juga dapat dipakai untuk me-*locate* elemen yang dicari.

1.7.4 Kriptografi

- Dari bahasa Yunani, artinya tulisan rahasia.
- Maknanya adalah menjaga keamanan pesan dengan mengubahnya dari pesan asli (*plaintext*) menjadi bentuk lain yang tak bermakna (*ciphertext*).
- Contohnya: culik anak itu jam 11 siang $\implies$ t$\hat{}$\$gfUi9rewoFpfdWqL:[uTcxZy
- Proses dari *plaintext* ke *ciphertext* dinamakan enkripsi, sebaliknya, dekripsi.

1.7.5 Kriptografi - *Caesar Cipher*

- Tiap huruf alfabet digeser 3 huruf ke kanan secara *wrapping*.
- Enkripsi dengan rumus:

$$c = E(p) = (p + 3) \mod 26$$

- Dekripsi dengan rumus:

$$p = D(c) = (c - 3) \mod 26$$

- Jika diperumum dengan pergeseran sejauh k, maka,

$$c = E(p) = (p + k) \mod 26$$

$$p = D(c) = (c - k) \mod 26$$

- Jika menggunakan basis ASCII, terdapat 256 karakter, maka,

$$c = E(p) = (p + k) \mod 256$$

$$p = D(c) = (c - k) \mod 256$$

1.7.6 Kriptografi - RSA

- Di algoritma RSA, enkripsi-dekripsi digunakan dengan dua kunci.
  1. Kunci publik e sebagai enkripsi pesan.
  2. Kunci privat e sebagai dekripsi pesan.
- Prosedur pembangkitan pasangan suci di RSA:
  1. Pilih dua bilangan prima, p dan q (rahasia).
  2. Hitung $n = pq$. Besaran n tidak perlu dirahasiakan
  3. Hitung $m = (p - 1)(q - 1)$ (rahasia).
  4. Pilih sebuah bilangan bulat untuk kunci publik, e, yang relatif prima terhadap m, yaitu $PBB(e, m) = 1$.
  5. Hitung kunci dekripsi, d, melalui kekongruenan $ed \equiv 1 \pmod{m}$.
- Prosedur enkripsi dan dekripsi:
  - Enkripsi: $p^e \equiv c \pmod{n}$ atau sama dengan $c = p^e \mod n$
  - Dekripsi: $c^d \equiv p \pmod{n}$ atau sama dengan $p = c^d \mod n$

1.7.7 Pembangkit Bilangan Acak-Semu

Pembangkit bilangan acak yang berbasis kekongruenan lanjar adalah *linear congruential generator* atau LCG.

$$X_n = (aX_{n-1} + b) \mod m$$

Xn = bilangan acak ke-n dari deretnya
$Xn - 1$ = bilangan acak sebelumnya
a = faktor pengali
b = increment
m = modulus
Kunci pembangkit adalah X_0 yang disebut umpan (*seed*).

Chapter 2

Kombinatorial

2.1 Definisi

Kombinatorial adalah cabang matematika untuk menghitung jumlah penyusunan objek-objek tanpa enumerasi kemungkinannya. Contohnya:

1. Nomor PIN kartu ATM bank adalah 6 angka. Berapa jumlah PIN yang dapat dibuat?
2. Kode buku sebuah perpustakaan terdiri dari dua huruf dan diikuti 4 angka. Berapa jumlah buku yang dapat dikodekan?
3. Berapa banyak cara membentuk sebuah komisi beranggotakan 10 orang yang dipilih dari 100 anggota DPR jika ketua DPR harus termasuk di dalamnya?

2.2 Kaidah Dasar Menghitung

Bila diberikan dua percobaan dengan hasil p dan q,

- Kaidah Perkalian: Percobaan 1 **dan** Percobaan 2 $= p \times q$ hasil.
- Kaidah Penjumlahan: Percobaan 1 **atau** Percobaan 2 $= p + q$ hasil.

Contoh 2.1: Ketua Angkatan

Berapa banyak cara memilih ketua angkatan IF 2023 (tidak terpaut gender) bila ada 65 pria dan 15 wanita?
Jawab: Ketua angkatan bisa dipilih dari pria **atau** wanita. Maka, terdapat $65 + 15 = 80$ cara.

Berapa banyak cara memilih dua wakil ketua angkatan IF 2023 dengan satu pria dan satu wanita, bila ada 65 pria dan 15 wanita?
Jawab: Harus ada satu pria **dan** satu wanita. Maka, terdapat $65 \times 15 = 975$ cara.

Kaidah ini dapat diperluas. Misalkan ada n percobaan dengan masing-masing percobaan menghasilkan p_i, maka:

- Kaidah Perkalian: $p_1 \times p_2 \times \cdots \times p_n$ hasil.
- Kaidah Penjumlahan: $p_1 + p_2 + \cdots + p_n$ hasil.

Contoh 2.2: Kata Sandi

Berapa banyak kata sandi yang dapat dibentuk dengan panjang 6-8 karakter dari a-z dan 0-9, *case-insensitive*?
Jawab: Terdapat $26 + 10 = 36$ karakter yang dapat mengisi satu tempat. Hitunglah kemungkinan enam, tujuh, dan delapan karakter.

- Untuk 6 karakter:
$$36 \times 36 \times \cdots \times 36 = 36^6$$
- Untuk 7 karakter:
$$36 \times 36 \times \cdots \times 36 = 36^7$$

- Untuk 8 karakter:

$$36 \times 36 \times \cdots \times 36 = 36^8$$

Sehingga totalnya adalah,

$$P = 36^6 + 36^7 + 36^8$$

Contoh 2.3: Susunan Huruf

Tersedia 6 huruf: a, b, c, d, e, f. Berapa jumlah susunan 3-huruf jika:

1. Tidak ada huruf yang diulang.
2. Boleh ada huruf yang berulang.
3. Tidak boleh ada huruf yang berulang, tetapi huruf e harus ada.
4. Boleh ada huruf yang berulang, tetapi huruf e harus ada.

Jawab:

1. $P = 6 \times 5 \times 4 = 120$
2. $P = 6 \times 6 \times 6 = 216$
3. $P = (1 \times 5 \times 4) \times 3 = 60$. Dikali tiga karena huruf e dapat berada di awal, tengah, atau akhir.
4. Rumusan skenario:
   - Semua kemungkinan : $Ps = 6 \times 6 \times 6 = 216$
   - Kemungkinan tidak ada huruf e: $P' = 5 \times 5 \times 5 = 125$
   - Jadi, kemungkinan ada minimal satu huruf e: $P = Ps - P' = 216 - 125 = 91$.

2.3 Prinsip Eksklusi-Inklusi

Contoh 2.4: Prinsip Eksklusi-Inklusi

Setiap byte disusun oleh 8-bit. Berapa banyak jumlah byte yang dimulai dengan '11' atau berakhir dengan '11'?
Jawab: Misalkan:

- A = himpunan byte yang dimulai dengan 11
- B = himpunan byte yang diakhiri dengan 11

Maka,

- $A \cup B$ = himpunan byte yang dimulai atau diakhiri dengan 11
- $A \cap B$ = himpunan byte yang dimulai dan diakhiri dengan 11

Dapat dituliskan:

$$A \cup B = A + B - (A \cap B) = 2^6 + 2^6 - 2^4 = 112$$

2.4 Permutasi

Definisi 2.1: Definisi Permutasi

Permutasi adalah jumlah urutan berbeda dari pengaturan objek-objek. Permutasi sejatinya adalah bentuk khusus dari kaidah perkalian.

Misalkan jumlah objek adalah n, maka,

- urutan pertama dipilih dari n objek.
- urutan kedua dipilih dari $n-1$ objek.
  $\vdots$
- urutan terakhir dipilih dari 1 objek yang tersisa.

Menurut kaidah perkalian, permutasi dari n objek adalah,

$$n(n-1)(n-2)\cdots(2)(1) = n!$$

Contoh 2.5: Contoh Permutasi

Berapa banyak kata 5 huruf yang terbentuk dari huruf-huruf kata "HAPUS"?
Jawab: Terdapat 5 posisi. Maka, penyelesaiannya $5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 = 120$ kata.

2.4.1 Permutasi r dari n elemen

Definisi 2.2: Permutasi r dari n elemen

Permutasi r dari n elemen adalah jumlah kemungkinan urutan r buah elemen yang dipilih dari n buah elemen, dengan $r \leq n$, yang dalam hal ini, pada setiap kemungkinan urutan tidak ada elemen yang sama.

$$P(n,r) = n(n-1)(n-2)\cdots(n-(r-1)) = \frac{n!}{(n-r)!}$$

Contoh 2.6: Pengaturan Supir

Sebuah mobil mempunyai 4 tempat duduk. Berapa banyak cara 3 orang didudukkan jika diandaikan satu orang harus duduk di kursi sopir?
Jawab: Banyak cara memilih satu supir dari tiga orang adalah 3 cara. Tersisa dua orang dan tiga kursi, maka cara perolehannya $P(3,2) = 6$ cara. Total cara yang mungkin adalah,

$$3 \cdot P(3,2) = 18$$

2.5 Kombinasi

Definisi 2.3: Kombinasi

- Bentuk khusus dari permutasi adalah kombinasi. Jika pada permutasi urutan kemunculan diperhitungkan, maka pada kombinasi urutan kemunculan diabaikan.
- Misalkan terdapat n benda dan ingin diambil r buah benda saja. Jika urutan diabaikan, maka akan terdapat $r!$ kemunculan yang sama.
- Sehingga, untuk mencari kombinasi,

$$C(n,r) = \binom{n}{r} = \frac{P(n,r)}{r!} = \frac{n!}{r!(n-r)!}$$

Contoh 2.7: Mengambil Bola

Misalkan ada 2 buah bola yang warnanya sama dan 3 buah kotak. Setiap kotak hanya boleh berisi paling banyak satu buah bola. Banyak cara memasukkan bola ke kotaknya adalah,

$$C(3,2) = \frac{3!}{2!(3-2)!} = 3$$

Interpretasi Kombinasi:

- Banyaknya himpunan bagian yang terdiri dari r elemen yang dapat dibentuk dari himpunan dengan n elemen.
- Banyak cara memilih r buah elemen dari n buah elemen yang ada, tetapi urutan elemen di dalam susunan hasil pemilihan tidak penting.

2.6 Permutasi dan Kombinasi Bentuk Umum

Misalkan ada n buah benda yang tak seluruhnya berbeda warna (ada yang warnanya sama). Berapa cara pengaturan n buah bola ke dalam kotak-kotak tersebut (tiap kotak berisi satu buah bola)?

Jawab: Jika n buah bola itu kita anggap berbeda semuanya, maka jumlah cara pengaturan n buah bola ke dalam n buah kotak adalah,

$$P(n,n) = n!$$

Yang artinya, dari pengaturan tersebut,

- ada $n_1!$ cara memasukkan bola berwarna 1
- ada $n_2!$ cara memasukkan bola berwarna 2
  $\vdots$
- ada $n_k!$ cara memasukkan bola berwarna k

Permutasi n buah bola di mana n_1 diantaranya berwarna 1, n_2 bola berwarna 2, $\cdots$, n_k bola berwarna k adalah,

$$P(n; n_1, n_2, \cdots, n_k) = \frac{P(n,n)}{n_1! n_2! \cdots n_k!} = \frac{n!}{n_1! n_2! \cdots n_k!}$$

Jumlah cara pengaturan seluruh bola kedalam kotak, yang ditemukan melalui kombinasi, adalah,

$$\begin{aligned} C(n; n_1, n_2, \cdots, n_k) &= C(n, n_1) C(n - n1, n_2) \cdots C(n - n_1 - n_2 - \cdots - n_{k-1}, n_k) \\ &= \frac{n!}{n_1!(n-n_1)!} \frac{(n-n_1)!}{n_2!(n-n_1-n_2)!} \cdots \frac{(n-n_1-n_2-\cdots-n_{k-1})!}{n_k!(n-n_1-n_2-\cdots-n_{k-1}-n_k)!} \\ &= \frac{n!}{n_1! n_2! \cdots n_k!} \end{aligned}$$

Teorema 2.1: Permutasi dan Kombinasi Bentuk Umum

Kesimpulannya, untuk permutasi dan kombinasi unsur umum,

$$P(n; n_1, n_2, \cdots, n_k) = C(n; n_1, n_2, \cdots, n_k) = \frac{n!}{n_1! n_2! \cdots n_k!}$$

Contoh 2.8: Mississippi

Berapa banyak "kata" yang dapat dibentuk dengan menggunakan huruf-huruf dari kata MISSISSIPPI?
Jawab: $S = \{M, I, S, S, I, S, S, I, P, P, I\}$

- Huruf M = 1 buah (n_1)
- Huruf I = 4 buah (n_2)
- Huruf S = 4 buah (n_3)
- Huruf P = 2 buah (n_4)
- $n = 11$

Dari Teorema 2.1,

$$\begin{aligned} P(n; n_1, n_2, \cdots, n_k) &= \frac{n!}{n_1! n_2! \cdots n_k!} \\ P(11; 1, 4, 4, 2) &= \frac{11!}{1! \cdot 4! \cdot 4! \cdot 2!} \\ &= \frac{11 \cdot 10 \cdot 9 \cdot 8 \cdot 7 \cdot 6 \cdot 5 \cdot \cancel{4!}}{1! \cdot \cancel{4!} \cdot 4! \cdot 2!} \\ &= 34650 \end{aligned}$$

2.7 Kombinasi Dengan Pengulangan

Misalkan terdapat r buah bola yang semua berwarna sama dan tersedia n buah kotak. Tinjau dua kasus berikut:

1. Jika masing-masing kotak hanya boleh diisi paling banyak satu buah bola, maka jumlah cara memasukkan bola: $C(n, r)$.
2. Jika masing-masing kotak boleh lebih dari satu buah bola (tidak ada pembatasan jumlah bola), maka jumlah cara memasukkan bola adalah,
$$C(n + r - 1, r)$$

Contoh 2.9: Penjumlahan Angka

Pada persamaan $x_1 + x_2 + x_3 + x_4 = 12$, x_i adalah bilangan bulat ≥ 0. Berapa jumlah kemungkinan solusinya?
Jawab: Analogikan seperti bola dan kotak. 12 bola akan dimasukkan ke 4 kotak. Maka, kemungkinan solusinya adalah,

$$C(12 + 4 - 1, 12) = C(15, 12) = 455$$

buah solusi.

Contoh 2.10: Penjumlahan Angka 2

Pada persamaan $x_1 + x_2 + x_3 + x_4 = 12$, x_i adalah bilangan bulat ≥ 0, $x_2 \geq 2$. Berapa jumlah kemungkinan solusinya?
Jawab: Analogikan seperti bola dan kotak. 12 bola akan dimasukkan ke 4 kotak. Namun, karena $x_2 \geq 2$, masukkan dua buah bola ke kotak x_2 terlebih dahulu. Sekarang, $n = 4$ dan $r = 10$. Pecahkan, solusinya adalah,

$$C(10 + 4 - 1, 4) = C(13, 10) = 286$$

buah solusi.

2.8 Koesifien Binomial

Perhatikan bahwa bilangan pada segitiga pascal adalah koefisien dari ekspansi binomial pangkat n.

Teorema 2.2: Koefisien Binomial dan Kombinasi

Koefisien ekspansi binomial dapat dituliskan dalam bentuk kombinasi, menjadi,

$$\begin{aligned} (x + y)^n &= C(n, 0)x^n + C(n, 1)x^{n-1}y + C(n, 2)x^{n-2}y^2 + \cdots + C(n, k)x^{n-k}y^k + \cdots + C(n, n)y^n \\ &= \sum_{k=0}^{n} C(n, k)x^{n-k}y^k \end{aligned}$$

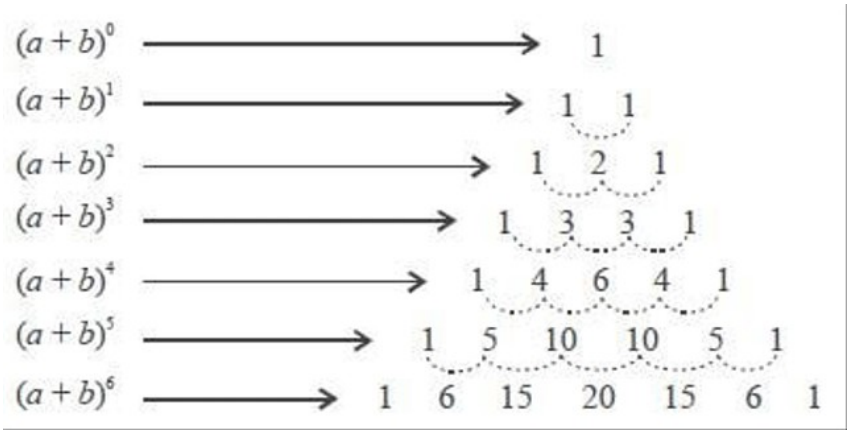


Figure 2.1: Hubungan Segitiga Pascal dan Ekspansi Binomial

2.9 *Pigeonhole Principle*

Definisi 2.4: 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 dua atau lebih objek. Maka, total jumlah objek paling banyak adalah n. Ini kontradiksi, karena jumlah objek paling sedikit $n+1$.

- Prinsip sarang merpati, jika diterapkan dengan baik, akan memberikan hanya objek-objek yang ada, dan bukan memberitahukan bagaimana mencari objek tersebut dan berapa banyak.
- Pada masalah sarang burung merpati, prinsip ini tidak memberitahukan di sarang merpati mana yang berisi lebih dari satu ekor merpati.

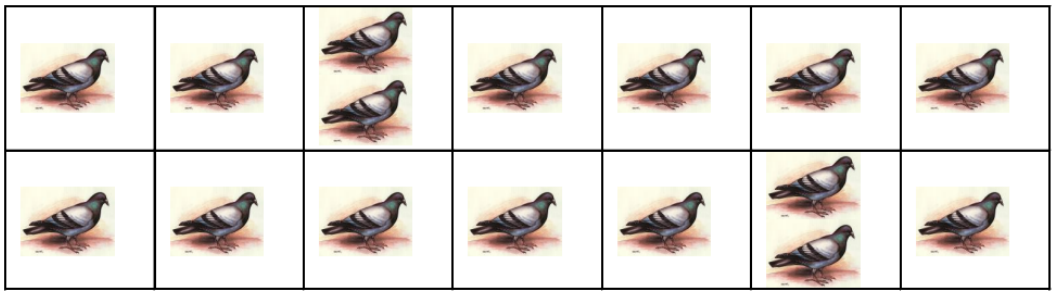


Figure 2.2: 14 Sarang dan 16 Merpati

Definisi 2.5: Prinsip Sarang Merpati Yang Dirampatkan

Jika M objek ditempatkan di dalam n buah kotak, maka paling sedikit terdapat satu kotak yang berisi minimal $\lceil \frac{M}{n} \rceil$objek.

Contoh 2.11: Ulang Tahun

Di antara 50 orang mahasiswa, berapa orang yang mungkin pada berulang tahun pada bulan yang sama?
Jawab: Minimal terdapat $\lceil \frac{50}{12} \rceil = 5$ orang.

Chapter 3

Graf

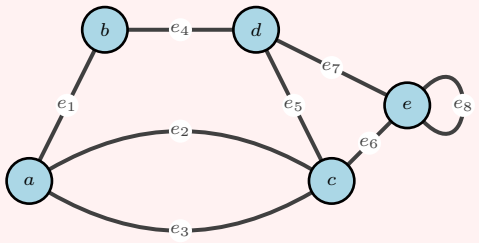
3.1 Definisi

Definisi 3.1: Graf

Graf adalah suatu pasangan terurut (V, E), dengan V adalah himpunan simpul (*vertices*) yang berisi elemen-elemen yang merepresentasikan titik atau objek dalam graf dan E adalah himpunan sisi (*edges*) yang berisi pasangan terurut (u, v), di mana $u, v \in V$ dan $u \neq v$, yang merepresentasikan hubungan atau koneksi antara simpul u dan simpul v.

Contoh 3.1

Pada graf G berikut,



$$V = \{a, b, c, d, e\}$$

dan

$$E = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8\} = \{(a, b), (a, c), (a, c), (b, d), (c, d), (c, e), (d, e), (e, e)\}.$$

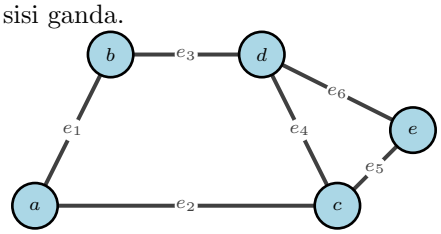
Sisi e_2 dan e_3 disebut **sisi ganda** (*multiple/parallel edges*) karena menghubungkan dua simpul yang sama, yakni a dan c. Sisi e_8 dinamakan **gelang** (*loop*) karena menghubungkan suatu simpul dengan dirinya sendiri.

3.2 Jenis-jenis Graf

3.2.1 Berdasarkan keberadaan gelang atau sisi ganda

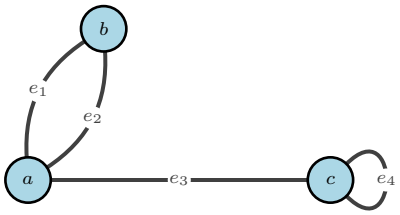
Graf sederhana

Graf yang tidak memuat gelang maupun sisi ganda.

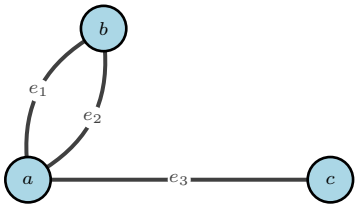


Graf tak-sederhana

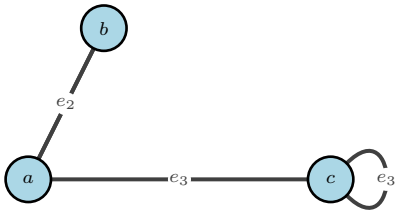
Graf yang memuat gelang atau sisi ganda.



1. **Graf ganda** (*multigraph*)
   Graf yang memuat sisi ganda.



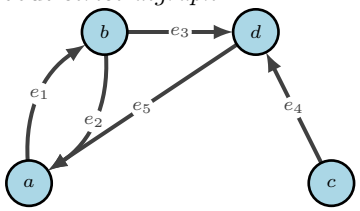
2. **Graf semu** (*pseudograph*)
   Graf yang memuat gelang.



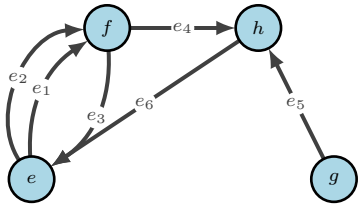
3.2.2 Berdasarkan keberadaan orientasi arah pada sisi

Graf berarah (*digraph*)

Graf yang tiap sisinya mempunyai orientasi arah. Meskipun menghubungkan dua simpul yang sama, dua buah sisi dianggap berbeda apabila arahnya berbeda (lihat sisi e_1 dan e_2 pada G_1). Sebaliknya, dua buah sisi dianggap sisi ganda apabila menghubungkan dua simpul yang sama dengan arah yang sama pula (lihat sisi e_1 dan e_2 pada G_2). Graf yang mengandung sisi semacam ini disebut **graf ganda berarah** (*multi-digraph*, lihat G_2), sementara graf berarah yang tidak mempunyai sisi ganda disebut *strict digraph*.



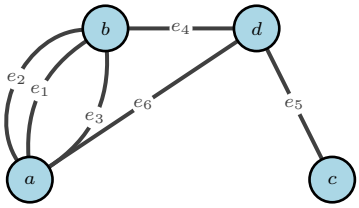
G_1



G_2

Graf tak-berarah

Graf yang semua sisinya tidak mempunyai orientasi arah.



Tabel berikut menggambarkan karakteristik macam-macam graf.

Jenis Graf	dapat mempunyai gelang?	dapat mempunyai sisi ganda?
graf tak sederhana	YA	YA
graf berarah ganda	YA	YA
graf ganda	TIDAK	YA
graf semu	YA	YA
graf sederhana	TIDAK	TIDAK
graf berarah	TIDAK	TIDAK

Table 3.1: Karakteristik tiap jenis graf

3.3 Terminologi

3.3.1 Ketetanggaan (*Adjacency*)

Dua buah simpul dikatakan bertetangga bila keduanya terhubung langsung oleh sebuah sisi.

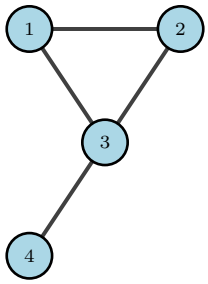


Figure 3.1: Contoh Ketetanggaan: Simpul 1 bertetangga dengan simpul 2 dan 3.

3.3.2 Bersisian (*Incidency*)

Sebuah sisi dikatakan bersisian dengan simpul jika sisi tersebut menghubungkan simpul tersebut.

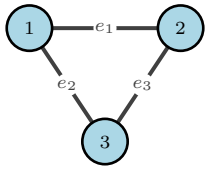


Figure 3.2: Contoh Bersisian: Sisi e_1 bersisian dengan simpul 1 dan 2.

3.3.3 Simpul Terpencil *(Isolated Vertex)*

Sebuah simpul terpencil adalah simpul yang tidak memiliki tetangga.

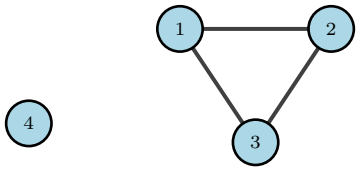


Figure 3.3: Contoh Simpul Kosong: Simpul 4 tidak memiliki tetangga.

3.3.4 Graf Kosong (*Null/Empty Graph*)

Sebuah graf kosong adalah graf yang tidak memiliki simpul.

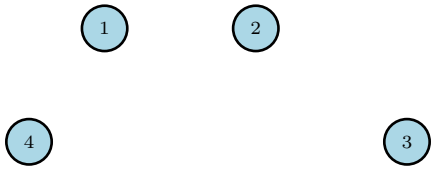


Figure 3.4: Contoh Graf Kosong.

3.3.5 Derajat Simpul

Derajat sebuah simpul adalah jumlah sisi yang bersisian dengan simpul tersebut.

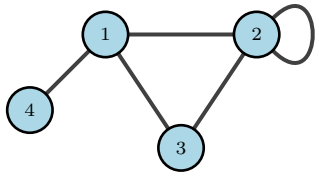


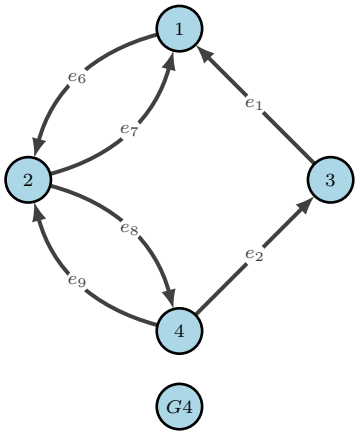
Figure 3.5: Contoh Derajat: Simpul 1 memiliki derajat 2, simpul 2 memiliki derajat 4 (*loop* dihitung dua kali; keluar dan masuk, simpul 3 memiliki derajat 2, dan simpul 4 memiliki derajat 1.

Pada graf berarah, derajat simpul dibedakan lagi menjadi:

- **Derajat Masuk (In-degree)**: Jumlah sisi yang masuk ke simpul.
- **Derajat Keluar (Out-degree)**: Jumlah sisi yang keluar dari simpul.

Contoh untuk graf G_4:

$$d_{\text{in}}(1) = 2, \quad d_{\text{out}}(1) = 1$$
$$d_{\text{in}}(2) = 2, \quad d_{\text{out}}(2) = 3$$
$$d_{\text{in}}(3) = 2, \quad d_{\text{out}}(3) = 1$$
$$d_{\text{in}}(4) = 1, \quad d_{\text{out}}(4) = 2$$



Teorema 3.1: Lema Jabat Tangan Dan Teorema Terkait

Jumlah derajat semua simpul pada suatu graf adalah genap, yaitu dua kali jumlah sisi pada graf tersebut. Secara matematis:

$$\sum_{v \in V} d(v) = 2|E|$$

Contoh untuk graf G_2:

$$d(1) + d(2) + d(3) + d(4) = 2 + 3 + 3 + 2 = 10$$
$$|E| = 5 \quad \Rightarrow \quad \sum_{v \in V} d(v) = 2|E|$$

Hal ini melahirkan kesimpulan:
Teorema: Untuk sembarang graf G, banyaknya simpul berderajat ganjil selalu genap.

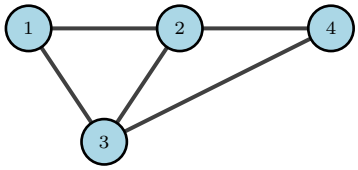


Figure 3.6: Graf G_2

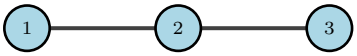


Figure 3.7: Contoh Lintasan Dari 1 Ke 3: $1 \rightarrow 2 \rightarrow 3$. Panjang Lintasan $= 2$

3.3.6 Lintasan (Path)

- 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 berbentuk v_0, e_1, v_1, e_2, v_2, $\cdots$, v_n–1, e_n, v_n sedemikian sehingga $e_1 = (v_0, v_1), e_2 = (v_1, v_2), \cdots, e_n = (v_n - 1, v_n)$ adalah sisi-sisi dari graf G.
- Jika graf mengandung sisi ganda, maka sisi e_i perlu dituliskan di dalam lintasan. Jika graf sederhana (tidak mengandung sisi ganda), sisi e_i tidak perlu ditulis.
- Panjang lintasan adalah banyak sisi yang dilewati.

3.3.7 Sirkuit/Siklus (Circuit/Cycle)

Sirkuit adalah lintasan tertutup yang dimulai dan berakhir di simpul yang sama tanpa mengulangi sisi.

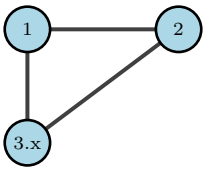


Figure 3.8: Contoh Sirkuit: $1 \rightarrow 2 \rightarrow 3 \rightarrow 1$.

3.3.8 Keterhubungan (*Connection*)

Graf dikatakan terhubung jika terdapat lintasan antara setiap pasangan simpul.

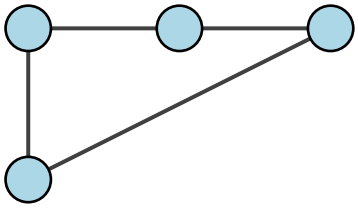


Figure 3.9: Contoh Graf Terhubung (*Connected*).

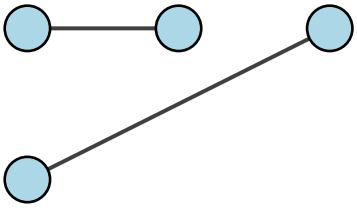


Figure 3.10: Contoh Graf Tidak Terhubung (*Disconnected*).

Graf berarah G disebut **graf terhubung kuat** (*strongly connected graph*) apabila untuk setiap pasangan simpul sembarang u dan v di G, terdapat lintasan dari u ke v dan dari v ke u. Jika tidak, G disebut **graf terhubung lemah**.

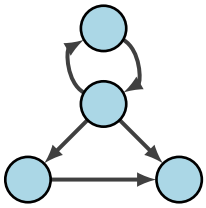


Figure 3.11: Graf terhubung lemah.

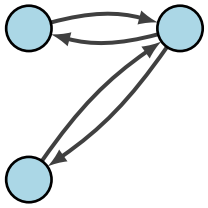


Figure 3.12: Graf terhubung kuat.

3.3.9 Upagraf dan Komplemen *(Subgraph and Complement)*

Upagraf adalah bagian dari graf yang terdiri dari sebagian simpul dan sisi graf induknya. Komplemen adalah graf dengan sisi yang tidak ada di upagraf dari suatu graf induk.

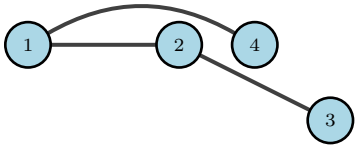


Figure 3.13: Graf Induk.

Figure 3.14: Upagraf dari Graf 1.13.

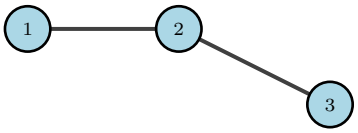


Figure 3.15: Komplemen dari Upagraf 1.14 dengan Graf 1.13.

3.3.10 Upagraf Merentang (*Spanning Subgraph*)

Upagraf merentang adalah upagraf yang mencakup semua simpul dari graf induk.

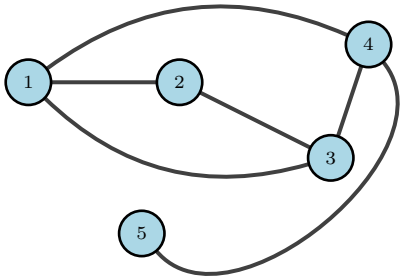


Figure 3.16: Graf Induk.

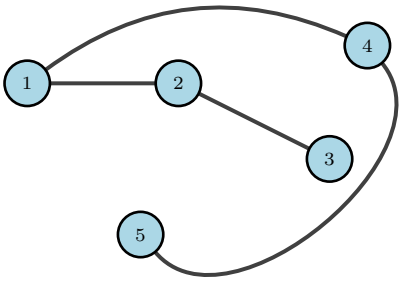


Figure 3.17: Graf Merentang Dari Graf 1.16.

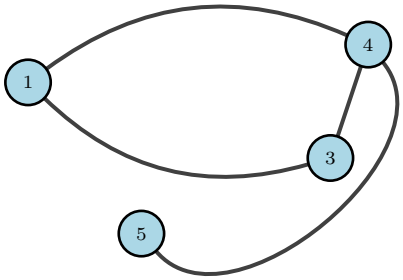


Figure 3.18: Bukan Graf Merentang Dari Graf 1.16 (Tidak ada simpul nomor 2).

3.3.11 Cut-Set

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.

Sebagai contoh, pada graf berikut, beberapa cut-set adalah:
- {(1,2), (1,5), (3,5), (3,4)} (Contohnya di bawah)
- {(1,2), (2,5)}
- {(1,3), (1,5), (1,2)}
- {(2,6)}

Namun, {(1,2), (2,5), (4,5)} bukan cut-set karena himpunan bagiannya, {(1,2), (2,5)}, adalah cut-set.

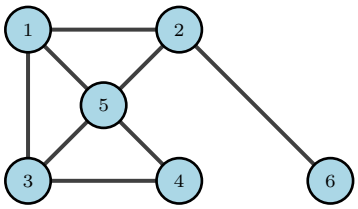


Figure 3.19: Graf Awal

Figure 3.20: Hasil 1

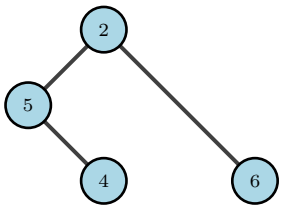


Figure 3.21: Hasil 2

3.3.12 Graf Berbobot

Graf berbobot adalah graf yang setiap sisinya diberi sebuah harga (bobot). Bobot ini dapat merepresentasikan jarak, biaya, waktu, atau nilai lainnya sesuai konteks penggunaan graf.

Sebagai contoh, pada graf berikut, bobot-bobot yang diberikan adalah:
- *(a,b)*: 10
- *(a,c)*: 12
- *(b,c)*: 8
- *(b,d)*: 15
- *(c,d)*: 9
- *(c,e)*: 11
- *(d,e)*: 14

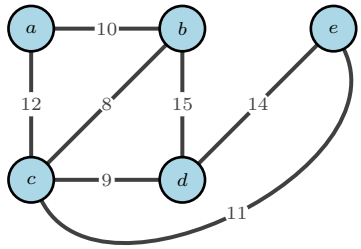


Figure 3.22: Contoh Graf Berbobot.

3.4 Beberapa Jenis Graf Khusus

3.4.1 Graf Lengkap

Graf lengkap adalah graf sederhana yang setiap simpulnya terhubung dengan semua simpul lainnya. Sebuah graf lengkap dengan n simpul memiliki $\frac{n(n-1)}{2}$ sisi.

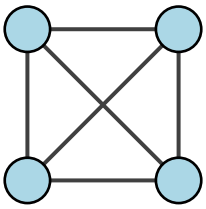


Figure 3.23: Graf Lengkap dengan 4 simpul

3.4.2 Graf Lingkaran

Graf lingkaran adalah graf sederhana di mana setiap simpul terhubung dengan tepat dua simpul lainnya, membentuk sebuah siklus.

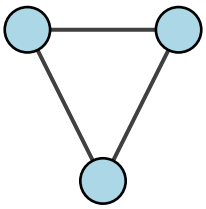


Figure 3.24: Graf Lingkaran dengan 3 simpul

3.4.3 Graf Teratur

Graf teratur adalah graf di mana setiap simpul memiliki derajat yang sama. Misalnya, graf teratur derajat r memiliki n simpul, dan setiap simpul terhubung dengan r simpul lainnya.

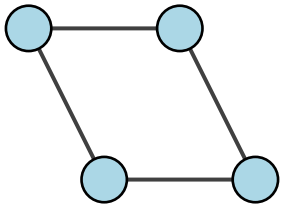


Figure 3.25: Graf Teratur dengan derajat 2

3.4.4 Graf Bipartite

Graf *bipartite* adalah graf yang himpunan simpulnya dapat dipisah menjadi dua himpunan bagian V_1 dan V_2, sedemikian sehingga setiap sisi pada graf G menghubungkan sebuah simpul di V_1 ke sebuah simpul di V_2. Graf *bipartite* dinyatakan sebagai $G(V_1, V_2)$.

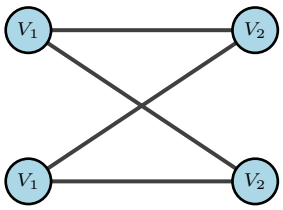


Figure 3.26: Graf Bipartite dengan V_1 dan V_2

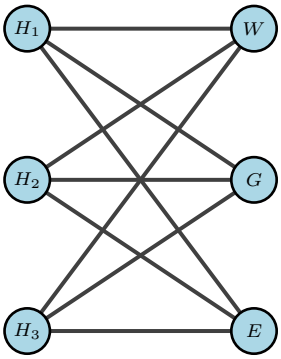


Figure 3.27: Graf Bipartite dengan $V_1 = \{H_1, H_2, H_3\}$ dan $V_2 = \{W, G, E\}$

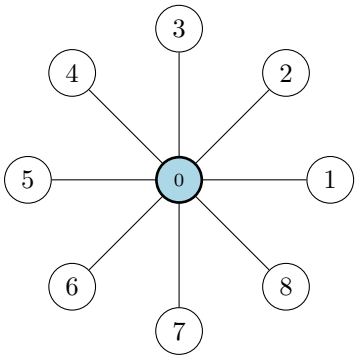


Figure 3.28: Graf Bipartite dengan V_1 = {simpul di tengah} dan V_2 = {simpul lainnya}

3.5 Representasi Graf

3.5.1 Matriks Ketetanggaan (*adjacency matrix*)

Matriks ketetanggaan adalah matriks $A = [a_{ij}]$, dengan elemen:

$$a_{ij} = \begin{cases} 1, & \text{jika simpul } i \text{ dan } j \text{ bertetangga} \\ 0, & \text{jika simpul } i \text{ dan } j \text{ tidak bertetangga} \end{cases}$$

Sebagai contoh, untuk graf berbobot:

	a	**b**	**c**	**d**	**e**
a	0	10	12	0	0
b	10	0	8	15	0
c	12	8	0	9	11
d	0	15	9	0	14
e	0	0	11	14	0

Pada graf berbobot, nilai setiap elemen a_{ij} adalah bobot sisi (i, j). Untuk graf berarah, elemen a_{ij} bernilai:

$$a_{ij} = \begin{cases} 1, & \text{jika terdapat sisi dari simpul } i \text{ menuju simpul } j \\ 0, & \text{jika tidak terdapat sisi dari simpul } i \text{ menuju simpul } j \end{cases}$$

Sebagai contoh, untuk graf berarah:

	1	**2**	**3**	**4**
1	0	1	0	1
2	0	0	1	0
3	0	0	0	1
4	0	1	0	0

3.5.2 Matriks Bersisian (*incidency matrix*)

Matriks bersisian adalah matriks $A = [a_{ij}]$, dengan elemen:

$$a_{ij} = \begin{cases} 1, & \text{jika simpul } i \text{ bersisian dengan sisi } j \\ 0, & \text{jika simpul } i \text{ tidak bersisian dengan sisi } j \end{cases}$$

Sebagai contoh, untuk graf berikut:

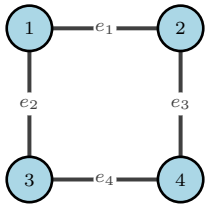


Figure 3.29: Graf untuk Matriks Bersisian

	e_1	e_2	e_3	e_4
1	1	1	0	0
2	1	0	1	0
3	0	1	0	1
4	0	0	1	1

3.5.3 Senarai Ketetanggaan (*adjacency list*)

Senarai ketetanggaan adalah daftar yang mencantumkan simpul-simpul tetangga dari setiap simpul pada graf. Sebagai contoh:

Simpul	**Tetangga**
1	2, 3
2	1, 4
3	1, 4
4	2, 3

Senarai ini berguna untuk representasi graf yang efisien dalam hal penggunaan memori.

3.6 Graf Isomorfik

Dua buah graf dikatakan isomorfik jika terdapat korespondensi satu-satu antara simpul-simpulnya dan antara sisi-sisinya sedemikian sehingga hubungan kebersisian tetap terjaga. Dengan kata lain, jika simpul u dan v pada graf G_1 bertetangga, maka simpul u' dan v' yang bersesuaian pada graf G_2 juga harus bertetangga.

Sebagai contoh, berikut adalah dua graf yang isomorfik:

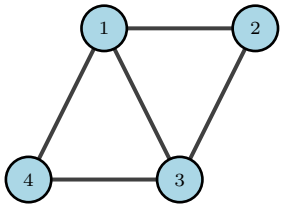


Figure 3.30: Graf G_1

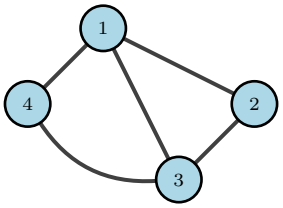


Figure 3.31: Graf G_2

Graf G_1 dan G_2 memiliki jumlah simpul, jumlah sisi, dan hubungan kebersisian yang sama, sehingga mereka adalah isomorfik.

3.6.1 Ciri-Ciri Graf Isomorfik

Beberapa ciri graf isomorfik adalah sebagai berikut:

- Memiliki jumlah simpul yang sama.
- Memiliki jumlah sisi yang sama.
- Derajat dari setiap simpul di graf pertama sama dengan derajat simpul yang bersesuaian di graf kedua.

Namun, ketiga syarat ini tidak cukup. Pemeriksaan lebih lanjut pada hubungan kebersisian diperlukan untuk memastikan isomorfisme.

3.6.2 Contoh Pemeriksaan Isomorfisme

Misalkan dua graf dengan matriks ketetanggaan sebagai berikut:

Graf G_1:

	1	2	3	4
1	0	1	1	1
2	1	0	1	0
3	1	1	0	1
4	1	0	1	0

Graf G_2:

	a	b	c	d
a	0	1	1	1
b	1	0	1	0
c	1	1	0	1
d	1	0	1	0

Kedua graf tersebut memiliki matriks ketetanggaan yang identik, sehingga keduanya adalah isomorfik.

3.6.3 Graf Tidak Isomorfik

Graf berikut tidak isomorfik karena hubungan kebersisian tidak sama meskipun jumlah simpul dan sisi sama:

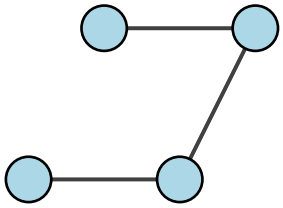


Figure 3.32: Graf G_3

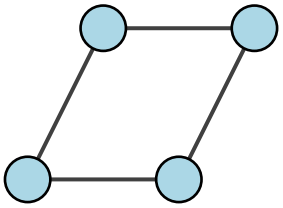


Figure 3.33: Graf G_4

Graf G_3 dan G_4 memiliki jumlah simpul dan sisi yang sama, tetapi hubungan kebersisian berbeda, sehingga keduanya tidak isomorfik.

3.7 Graf Planar dan Graf Bidang

3.7.1 Graf Planar

Graf planar adalah graf yang dapat digambarkan pada bidang datar sedemikian sehingga tidak ada sisi-sisi yang saling berpotongan kecuali di simpul. Sebagai contoh, graf K_4 berikut adalah graf planar:

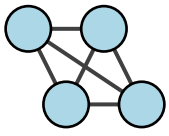


Figure 3.34: Graf K_4, contoh graf planar

Sebuah graf dikatakan planar jika memenuhi ketidaksamaan Euler:

$$n - e + f = 2,$$

dengan n adalah jumlah simpul, e adalah jumlah sisi, dan f adalah jumlah wilayah (termasuk wilayah luar).

3.7.2 Graf Bidang

Graf bidang adalah representasi graf planar yang digambar pada bidang datar dengan sisi-sisi yang tidak saling berpotongan. Berikut adalah contoh graf bidang untuk K_4:

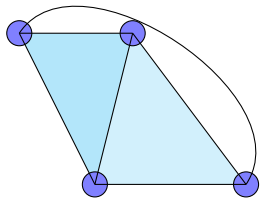


Figure 3.35: Graf bidang K_4

Graf bidang membagi bidang menjadi beberapa wilayah tertutup. Misalnya, pada graf K_4 di atas, terdapat 4 wilayah (termasuk wilayah yang di dalam garis melengkung dan wilayah luar graf).

3.7.3 Teorema Kuratowski

Teorema Kuratowski menyatakan bahwa sebuah graf bersifat planar jika dan hanya jika graf tersebut tidak mengandung subgraf yang isomorfik dengan K_5 atau $K_{3,3}$. K_5 adalah graf lengkap dengan lima simpul, sedangkan $K_{3,3}$ adalah graf bipartit lengkap dengan tiga simpul di setiap himpunan bagian.

Contoh graf K_5:

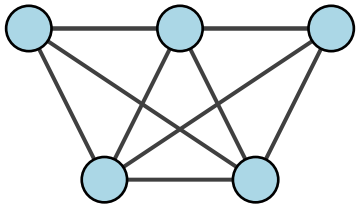


Figure 3.36: Graf K_5, graf tidak planar

Contoh graf $K_{3,3}$:

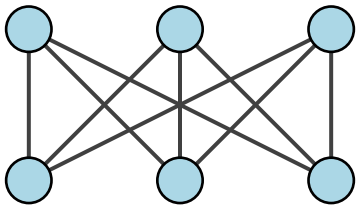


Figure 3.37: Graf $K_{3,3}$, graf tidak planar

Graf K_5 dan $K_{3,3}$ adalah contoh graf tidak planar karena tidak dapat digambarkan pada bidang datar tanpa sisi yang saling berpotongan.

3.7.4 Graf Dual

Graf dual adalah graf yang dibangun dari graf bidang dengan mendefinisikan simpul untuk setiap wilayah pada graf bidang asli dan menghubungkan dua simpul jika wilayah-wilayah yang bersesuaian berbatasan secara langsung. Graf dual sangat berguna dalam berbagai aplikasi, seperti perencanaan jaringan dan analisis peta.

Graf bidang memiliki wilayah yang dipetakan ke simpul-simpul pada graf dual. Dalam contoh ini, simpul pada graf dual merepresentasikan wilayah dari graf planar asli.

Langkah-Langkah Membentuk Graf Dual

Berikut adalah langkah-langkah untuk membentuk graf dual dari sebuah graf planar:

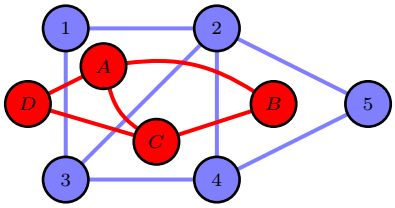


Figure 3.38: Contoh merah sebagai graf dual dari graf bewarna biru. Graf merah menyatakan wilayah dari grab biru.

1. Identifikasi semua wilayah (termasuk wilayah luar) pada graf planar.
2. Tempatkan simpul di setiap wilayah yang telah diidentifikasi.
3. Hubungkan dua simpul dengan sebuah sisi jika wilayah-wilayah yang bersesuaian berbatasan langsung melalui sebuah sisi graf planar.

Langkah-langkah ini menghasilkan graf dual seperti pada gambar sebelumnya.

3.8 Lintasan dan Sirkuit Euler

Lintasan Euler adalah lintasan yang melalui masing-masing sisi dalam graf tepat satu kali, sedangkan sirkuit Euler adalah lintasan yang kembali ke simpul awal dan melalui masing-masing sisi tepat satu kali. Graf yang memiliki sirkuit Euler disebut graf Euler, dan graf yang memiliki lintasan Euler disebut graf semi-Euler.

Teorema 3.2: Lintasan Dan Sirkuit Euler

Teorema 1: Graf tidak berarah G adalah graf Euler jika dan hanya jika G terhubung dan setiap simpul berderajat genap.
Teorema 2: Graf tidak berarah memiliki lintasan Euler jika dan hanya jika G terhubung dan memiliki tepat dua simpul berderajat ganjil.
Teorema 3: Graf berarah G memiliki sirkuit Euler jika dan hanya jika:
1. Setiap simpul memiliki derajat masuk sama dengan derajat keluar.
2. Semua simpul yang memiliki derajat lebih besar dari nol berada dalam satu komponen terhubung kuat.

Sebagai contoh:

- Graf dengan simpul berderajat genap adalah graf Euler.
- Graf dengan dua simpul berderajat ganjil adalah graf semi-Euler.

Berikut adalah beberapa contoh lintasan dan sirkuit Euler berdasarkan graf:

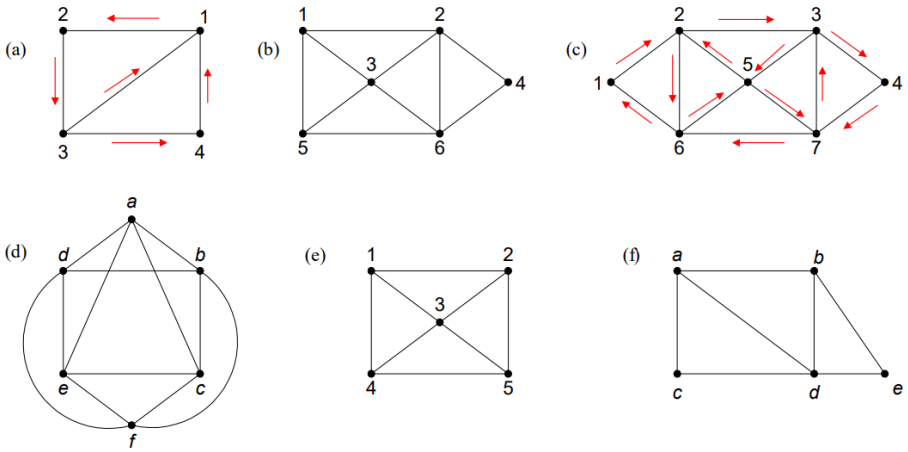


Figure 3.39: Contoh lintasan dan sirkuit Euler pada graf

- **Lintasan Euler pada graf (a)**: $3, 1, 2, 3, 4, 1$

- **Lintasan Euler pada graf (b)**: $1, 2, 4, 6, 2, 3, 6, 5, 1, 3$
- **Sirkuit Euler pada graf (c)**: $1, 2, 3, 4, 7, 3, 5, 7, 6, 5, 2, 6, 1$
- **Sirkuit Euler pada graf (d)**: $a, c, f, e, c, b, d, e, a, d, f, b, a$
- **Graf (e)** tidak memiliki lintasan maupun sirkuit Euler.
- **Graf (f)** memiliki lintasan Euler: a, c, d, a, b, e, d, b

Persoalan Jembatan Königsberg adalah contoh klasik dari graf yang tidak memiliki lintasan maupun sirkuit Euler karena setiap simpul memiliki derajat ganjil.

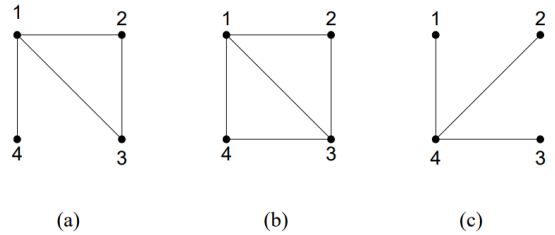
3.9 Lintasan dan Sirkuit Hamilton

Lintasan Hamilton adalah lintasan yang melalui setiap simpul dalam graf tepat satu kali, sedangkan sirkuit Hamilton adalah lintasan yang kembali ke simpul awal dan melalui setiap simpul tepat satu kali. Graf yang memiliki sirkuit Hamilton disebut graf Hamilton, sedangkan graf yang hanya memiliki lintasan Hamilton disebut graf semi-Hamilton.

Teorema 3.3: Lintasan Dan Sirkuit Hamilton

Teorema 4: Syarat cukup agar graf sederhana G dengan $n \geq 3$ simpul adalah graf Hamilton jika derajat setiap simpul $d(v) \geq n/2$.
Teorema 5: Graf lengkap K_n adalah graf Hamilton.

Sebagai contoh, graf berikut memiliki lintasan dan sirkuit Hamilton:



(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

Figure 3.40: Contoh Graf Hamilton

3.10 Aplikasi Graf

3.10.1 Lintasan Terpendek

Lintasan terpendek digunakan untuk mencari jalur dengan bobot total minimum antara dua simpul dalam graf berbobot. Algoritma populer untuk menyelesaikan persoalan ini adalah algoritma Dijkstra dan algoritma Bellman-Ford. Akan dibahas lebih lanjut pada IF2122 Strategi Algoritma.

3.10.2 *Traveling Salesman Problem* (TSP)

Persoalan pedagang keliling adalah menentukan tur terpendek yang melewati setiap simpul dalam graf berbobot tepat satu kali dan kembali ke simpul awal. Solusi TSP adalah sirkuit Hamilton dengan bobot minimum.
Misalkan sebuah persoalan TSP direpresentasikan dengan graf lengkap 4 simpul. Jumlah sirkuit Hamilton di dalam graf lengkap dengan n simpul adalah:

$$\frac{(n-1)!}{2}$$

Sebagai contoh, graf di bawah memiliki $(4-1)!/2 = 3$ sirkuit Hamilton, yaitu:
Graf di atas memiliki 3 sirkuit Hamilton:

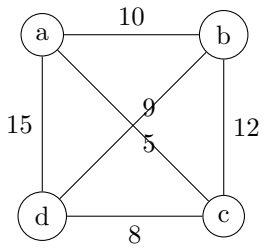


Figure 3.41: Graf lengkap 4 simpul dengan bobot sisi

1. $l_1 = (a, b, c, d, a)$ dengan bobot: $10 + 12 + 8 + 15 = 45$
2. $l_2 = (a, c, d, b, a)$ dengan bobot: $5 + 9 + 15 + 10 = 39$
3. $l_3 = (a, c, b, d, a)$ dengan bobot: $5 + 12 + 9 + 15 = 41$

Solusi TSP: Sirkuit Hamilton terpendek adalah $l_2 = (a, c, d, b, a)$ dengan bobot total 39.
Jika jumlah simpul $n = 20$, akan terdapat $(19)!/2$ sirkuit Hamilton atau sekitar 6×10^{16} penyelesaian.

3.10.3 *Chinese Postman Problem*

Persoalan ini adalah menentukan rute terpendek untuk melewati setiap sisi dalam graf setidaknya satu kali dan kembali ke simpul awal. Jika graf adalah graf Euler, sirkuit Euler dapat ditemukan dengan mudah. Jika bukan graf Euler, beberapa sisi harus dilalui lebih dari sekali untuk memenuhi persyaratan.

3.10.4 Pewarnaan Graf

Pewarnaan graf adalah proses pemberian warna pada simpul-simpul atau sisi-sisi graf sehingga memenuhi aturan tertentu. Pewarnaan graf digunakan dalam berbagai aplikasi seperti pewarnaan peta, penjadwalan, dan masalah alokasi.

Pewarnaan Simpul

Pada pewarnaan simpul, setiap simpul diberi warna sehingga tidak ada dua simpul bertetangga yang memiliki warna yang sama. Jumlah minimum warna yang diperlukan untuk mewarnai graf disebut **bilangan kromatik** (χ) graf.

Teorema 3.4: Teorema Empat Warna

Setiap graf planar dapat diwarnai dengan paling banyak empat warna.

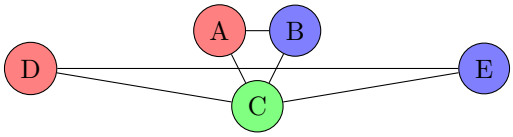


Figure 3.42: Graf planar dengan pewarnaan simpul menggunakan tiga warna, $\chi = 3$

Algoritma Welsh-Powell

Teorema 3.5: Algoritma Welsh-Powell

Algoritma Welsh-Powell adalah algoritma untuk pewarnaan simpul graf yang bekerja dengan mengurutkan simpul berdasarkan derajatnya secara menurun dan memberikan warna secara bertahap. Langkah-langkahnya adalah:

1. Urutkan simpul-simpul graf berdasarkan derajatnya secara menurun.
2. Pilih simpul pertama dan beri warna pertama.
3. Warnai simpul-simpul berikutnya dengan warna yang sama jika simpul tersebut tidak bertetangga dengan simpul yang sudah diwarnai.

4. Ulangi langkah ini hingga semua simpul terwarnai.

Sebagai contoh, misalkan graf berikut:

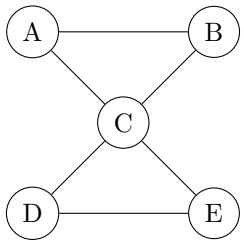


Figure 3.43: Graf untuk contoh Algoritma Welsh-Powell

Langkah-langkah pewarnaan menggunakan algoritma Welsh-Powell:

- Urutkan simpul berdasarkan derajat: $C(4), A(2), B(2), D(2), E(2)$.
- Warnai simpul C dengan warna pertama (merah).
- Warnai simpul A dan E dengan warna kedua (biru).
- Warnai simpul B dan D dengan warna ketiga (hijau).

Hasil pewarnaan:

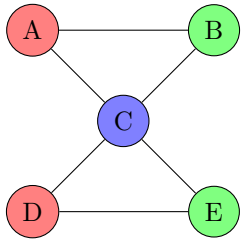


Figure 3.44: Hasil pewarnaan menggunakan Algoritma Welsh-Powell, $\chi = 3$

Pewarnaan Sisi (Tidak Dibahas)

Pada pewarnaan sisi, setiap sisi diberi warna sehingga tidak ada dua sisi yang *incident* pada simpul yang sama memiliki warna yang sama. Jumlah minimum warna yang diperlukan untuk mewarnai sisi-sisi graf disebut **bilangan kromatik sisi** (χ') graf.

Teorema 3.6: Teorema Vizing

Untuk graf sederhana G, bilangan kromatik sisi $\chi'(G)$ adalah $\Delta(G)$ atau $\Delta(G) + 1$, di mana $\Delta(G)$ adalah derajat maksimum graf.

Sebagai contoh:

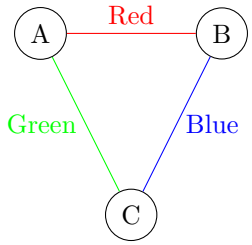


Figure 3.45: Pewarnaan sisi pada graf

Aplikasi Pewarnaan Graf

Pewarnaan graf memiliki berbagai aplikasi dalam kehidupan nyata, antara lain:

- **Pewarnaan Peta**: Untuk memastikan bahwa wilayah yang bertetangga memiliki warna yang berbeda.
- **Penjadwalan Ujian**: Untuk memastikan bahwa mahasiswa yang mengambil mata kuliah yang sama tidak dijadwalkan pada waktu yang sama.
- **Alokasi Frekuensi Radio**: Untuk menghindari interferensi antara jaringan yang berdekatan.

Chapter 4

Pohon

4.1 Definisi Pohon

Pohon adalah graf tak-berarah yang terhubung dan tidak mengandung sirkuit (siklus). Syarat pohon adalah graf tidak berarah, harus terhubung, dan tidak memiliki sirkuit.

Teorema 4.1: Sifat-Sifat Pohon

Misalkan $G = (V, E)$ adalah graf tak-berarah sederhana dengan jumlah simpul n. Maka, pernyataan berikut ekuivalen:

1. G adalah pohon.
2. Setiap pasang simpul di dalam G terhubung dengan lintasan tunggal.
3. G terhubung dan memiliki $m = n - 1$ sisi.
4. G tidak mengandung sirkuit dan memiliki $m = n - 1$ sisi.
5. G terhubung dan semua sisinya adalah jembatan.

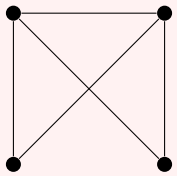
Kumpulan dari berbagai Pohon disebut Hutan. Hutan adalah kumpulan pohon yang saling lepas atau graf tidak terhubung yang tidak mengandung sirkuit.

4.2 Pohon Merentang (*Spanning Tree*)

Pohon merentang dari sebuah graf terhubung adalah subgraf merentang yang berupa pohon. Pohon merentang diperoleh dengan memutus sirkuit di dalam graf.

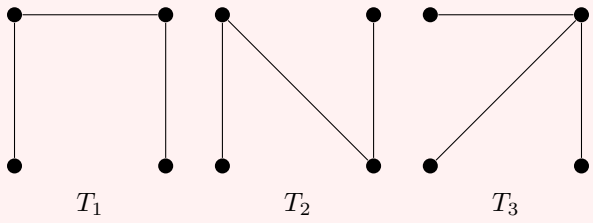
Contoh 4.1: Pemutusan Graf

Gambarkan minimal tiga pohon merentang yang dapat dibuat dari graf G.



G

Jawab:



4.3 Pohon Merentang Minimum (*Minimum Spanning Tree*)

Graf yang berbobot juga dapat dibuat perentangan pohonnya. Pohon dengan bobot total terendah dinamakan **Minimum Spanning Tree (MST)**.

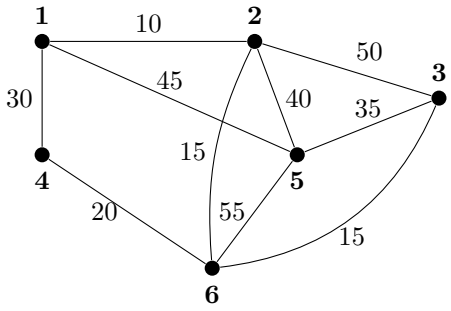


Figure 4.1: Graf Berbobot

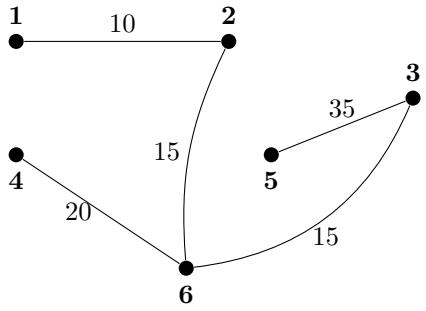


Figure 4.2: Minimum Spanning Tree (MST)

Terdapat dua algoritma yang dapat digunakan dalam menentukan *minimum spanning tree*.

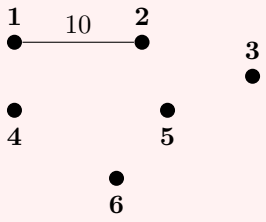
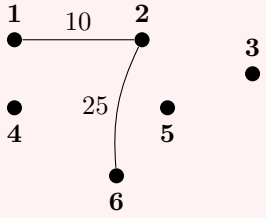
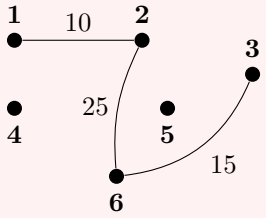
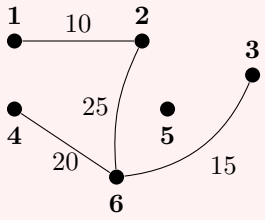
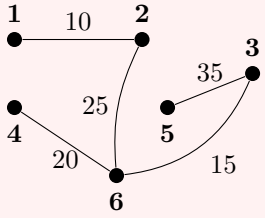
4.3.1 Algoritma Prim

Dimulai dari simpul dengan bobot terkecil dan membangun pohon secara iteratif tanpa membentuk sirkuit. Langkah-langkah:

1. Ambil sisi G yang paling minimum, masukkan ke dalam T (Cikal bakal MST).
2. Pilih sisi (u, v) yang bersisian dengan simpul di T dan minimum, namun tak membentuk sirkuit. Masukkan ke T.
3. Langkah 2 diulangi sebanyak $n - 2$ kali.

Contoh 4.2: Algoritma Prim

Perolehlah Figure 1.2 dari Figure 1.1 dengan Algoritma Prim!
Penyelesaian:

Langkah	Sisi	Bobot	Pohon rentang
1	(1, 2)	10	
2	(2, 6)	25	
3	(3, 6)	15	
4	(4, 6)	20	
5	(3, 5)	35	

Bobot total yang dihasilkan: 105.

4.3.2 Algoritma Kruskal

Menyusun sisi berdasarkan bobot terkecil dan menambahkan sisi ke dalam pohon tanpa membentuk sirkuit. Langkah-langkah:

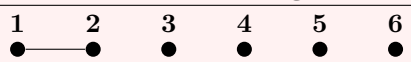
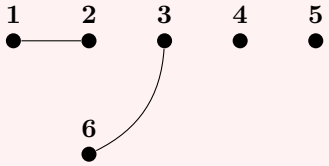
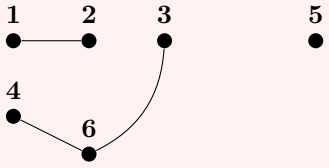
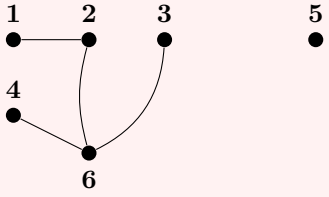
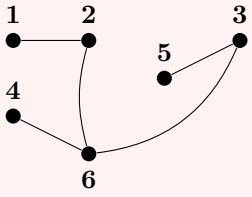
1. Urutkan sisi-sisi dahulu dari yang terkecil.
2. Pilih sisi (u, v) dengan bobot minimum yang tidak membentuk sirkuit. Tambahkan (u, v) ke dalam T.
3. Ulangi langkah 2 sebanyak $n - 1$ kali.

Contoh 4.3: Algoritma Kruskal

Perolehlah Figure 1.2 dari Figure 1.1 dengan Algoritma Kruskal!
Penyelesaian:
Sisi sudah diurutkan.

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

Selanjutnya, dilakukan pengurutan.

Langkah	Sisi	Bobot	Pohon rentang
1	(1,2)	10	
2	(3,6)	15	
3	(4,6)	20	
4	(2,6)	25	
5	(1,4)	30	Ditolak (Siklik)
6	(3,5)	35	

Bobot total yang dihasilkan: 105

4.4 Pohon Berakar

Pohon berakar adalah pohon yang salah satu simpulnya diperlakukan sebagai akar, dengan sisi-sisinya diberi arah.

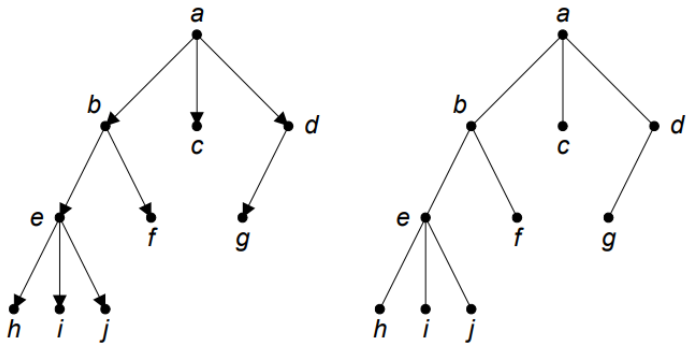


Figure 4.3: Contoh pohon berakar. Sebagai kesepakatan, tanda panah dapat tidak ditulis (namun tetap menyiratkan bahwa arahnya ke bawah)

4.5 Terminologi pada Pohon Berakar

- **Anak (Child):** Simpul yang terhubung langsung dengan orang tua.
- **Orang Tua (Parent):** Simpul yang memiliki anak.

- **Daun (Leaf):** Simpul yang tidak memiliki anak.
- **Lintasan (Path):** Arah jalan dari suatu simpul ke simpul anakannya. Sebagai contoh, f dapat dicapai dari a dengan lintasan $a - b - f$.
- **Saudara Kandung (Sibling):** Simpul yang orangtuanya sama dan setingkat. Sebagai contoh, e adalah saudara dari f karena orangtuanya saam (e), tetapi g bukan *sibling* dari f.
- **Upaphon (Subtree):** Anak dari suatu orang tua dapat diklasifikasikan sebagai tree itu sendiri. Karena anak tersebut masih memiliki orang tua, maka ia disebut upapohon. Sebagai contoh, b dan anakannya adalah upapohon dari a dan anakannya.
- **Derajat (Degree):** Jumlah anak pada suatu simpul.
- **n-Ary Tree:** Pohon dengan banyak anak pada semua simpul *maksimum* n.
- **Aras(Level):** Tingkat simpul pada suatu pohon.
- **Depth/Height:** Aras maksimum dari pohon.

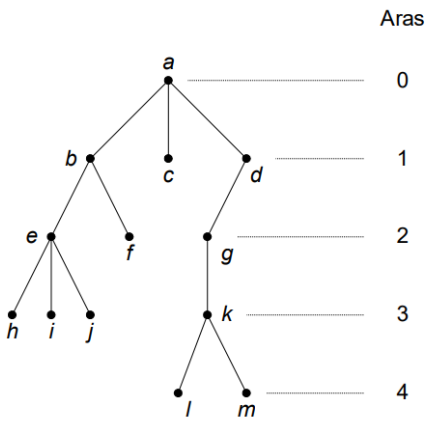


Figure 4.4: Ilustrasi Tingkatan

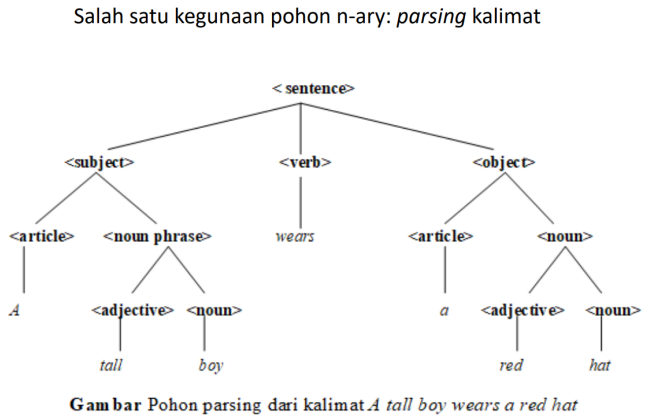


Figure 4.5: Kegunaan n-ary Tree

4.6 Pohon Terurut dan Pohon Biner

- Pohon berakar dengan urutan anak yang penting disebut pohon terurut.
- Pohon biner adalah pohon yang setiap simpulnya memiliki paling banyak dua anak.
  - Anak kiri dan kanan dibedakan.
  - Karena ada perbedaan urutan anak, pohon biner adalah pohon terurut.
  - Pohon ini banyak aplikasinya.
- Jenis-jenis dari *Binary Tree*, mencakup:
  1. **Pohon Biner Condong Kiri/Kanan:** Pohon biner yang semua anaknya berada di bagian kiri atau kanan.
  2. **Pohon Biner Penuh:** Setiap simpul, kecuali simpul pada aras terbawah, tepat mempunyai dua anak.
  3. **Pohon Biner Lengkap:** Setiap aras, mungkin kecuali pada aras terakhir, terisi lengkap, dan semua simpul dipadatkan sejauh mungkin ke bagian kiri.
  4. **Pohon Biner Seimbang:** Pada beberapa aplikasi, diinginkan tinggi upapohon kiri dan tinggi upapohon kanan yang seimbang, yaitu berbeda maksimal 1.

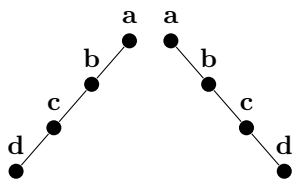


Figure 4.6: Kiri: Pohon Condong Kiri; Kanan: Pohon Condong Kanan

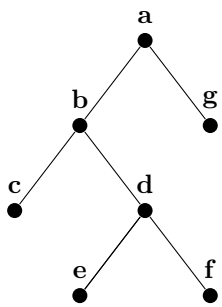


Figure 4.7: Contoh Pohon Biner Penuh

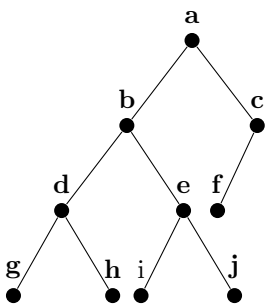
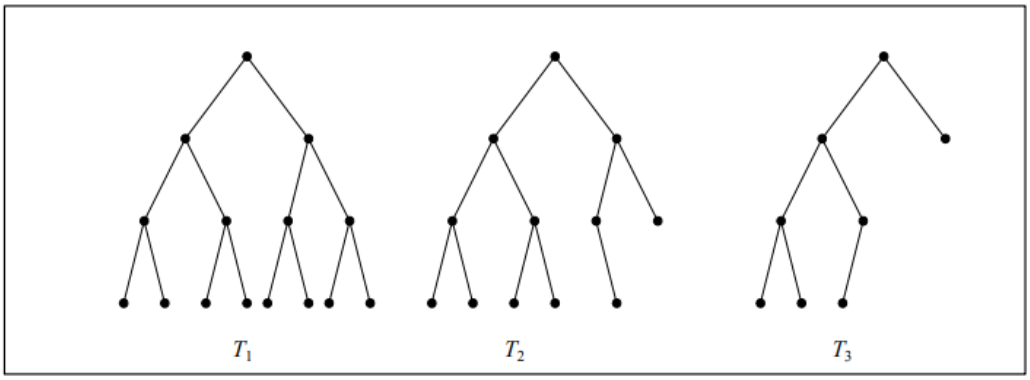


Figure 4.8: Contoh Pohon Biner Lengkap



Gambar T_1 dan T_2 adalah pohon seimbang, sedangkan T_3 bukan pohon seimbang.

4.7 Aplikasi Pohon

- **Pohon Ekspresi:** Digunakan untuk representasi operasi matematika.
- **Pohon Keputusan:** Digunakan dalam pengambilan keputusan, seperti algoritma pembelajaran mesin.
- **Kode Huffman:** Digunakan untuk kompresi data.

4.8 Kode Huffman

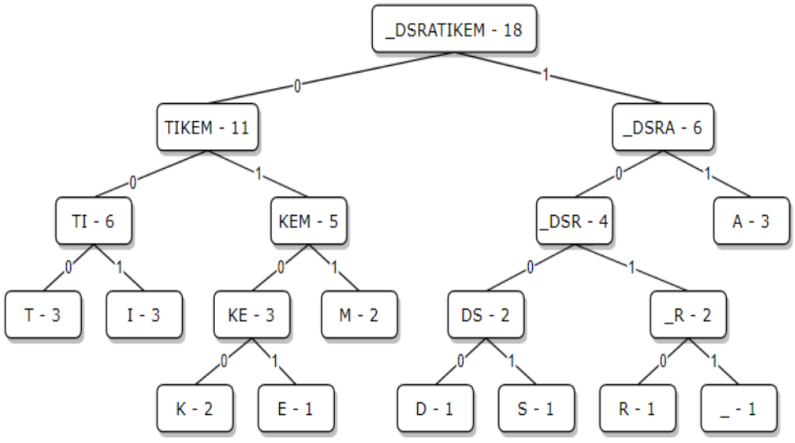


Figure 4.9: Pohon Huffman untuk string `MATEMATIKA DISKRIT`.

Kode Huffman digunakan untuk kompresi data dengan mengganti simbol yang sering muncul dengan kode biner yang lebih pendek. Berikut adalah langkah-langkah algoritma Huffman:

1. Urutkan simbol berdasarkan frekuensi kemunculannya dari kecil ke besar.
2. Pilih dua simbol dengan frekuensi terkecil, lalu gabungkan menjadi simpul baru dengan frekuensi yang merupakan jumlah dari kedua simbol tersebut.
3. Ulangi langkah sebelumnya sampai hanya tersisa satu simpul yang mencakup semua simbol.
4. Berikan label '0' untuk cabang kiri dan '1' untuk cabang kanan pada setiap simpul.
5. Telusuri pohon dari akar ke setiap daun untuk mendapatkan kode Huffman untuk setiap simbol.

Contoh 4.4: Algoritma Kompresi Huffman

Diberikan string `ABACCDA` dengan distribusi frekuensi sebagai berikut:

Simbol	Frekuensi
A	3
B	1
C	2
D	1

Proses pembentukan kode Huffman:

1. Gabungkan simbol B dan D menjadi simpul baru BD dengan frekuensi $1 + 1 = 2$.
2. Gabungkan BD dan C menjadi simpul baru CBD dengan frekuensi $2 + 2 = 4$.
3. Gabungkan CBD dan A menjadi simpul akhir ACBD dengan frekuensi $4 + 3 = 7$.

Berikut adalah hasil kode Huffman untuk setiap simbol:

Simbol	Kode Huffman
A	0
B	110
C	10
D	111

String `ABACCDA` setelah dikodekan menjadi: `0110010101110`.

4.9 Traversal Pohon Biner

4.9.1 Preorder: R, T_1, T_2

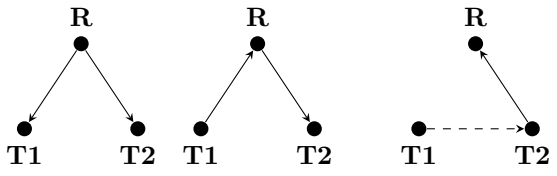
- Kunjungi R
- Kunjungi T_1 secara preorder
- Kunjungi T_2 secara preorder

4.9.2 Inorder: T_1, R, T_2

- Kunjungi T_1 secara inorder
- Kunjungi R
- Kunjungi T_2 secara inorder

4.9.3 Postorder: T_1, T_2, R

- Kunjungi T_1 secara postorder
- Kunjungi T_2 secara postorder
- Kunjungi R



Chapter 5

Kompleksitas Algoritma

5.1 Pendahuluan

- Sebuah algoritma tidak saja harus benar (sesuai spesifikasi persoalan), tetapi juga harus sangkil (efisien).
- Algoritma yang bagus adalah algoritma yang sangkil (*efficient*).
- Kesangkilan algoritma diukur dari waktu (*time*) yang diperlukan untuk menjalankan algoritma dan ruang (*space*) memori yang dibutuhkan oleh algoritma tersebut.
- Algoritma yang sangkil ialah algoritma yang meminimumkan kebutuhan waktu dan ruang memori.
- Kebutuhan waktu dan ruang memori suatu algoritma bergantung pada ukuran masukan (n), yang menyatakan ukuran data yang diproses oleh algoritma.
- Kesangkilan algoritma dapat digunakan untuk menilai algoritma yang bagus dari sejumlah algoritma penyelesaian persoalan.
- Sebab, sebuah persoalan dapat memiliki banyak algoritma penyelesaian. Contoh: persoalan pengurutan (sort), ada puluhan algoritma pengurutan (selection sort, insertion sort, bubble sort, dll).

Definisi 5.1: Kompleksitas Algoritma

Model abstrak pengukuran waktu/ruang memori algoritma harus independen dari pertimbangan mesin (*computer*) dan *compiler* apapun.

Besaran yang dipakai untuk menerangkan model abstrak pengukuran waktu/ruang ini adalah kompleksitas algoritma.

Dua jenis kompleksitas algoritma: Time/Waktu ($T(n)$) dan Space/Ruang ($S(n)$)

5.2 Model Perhitungan Kebutuhan Waktu

Mengapa lebih terfokus pada kompleksitas waktu?

- Materinya di luar lingkup Matdis (walawe),
- Persoalan memori bukan bagian kritis pada gawai modern.

Menghitung kebutuhan waktu algoritma dengan mengukur waktu eksekusi riilnya (dalam satuan detik atau mikrodetik) saat program (yang merepresentasikan sebuah algoritma) dijalankan oleh komputer bukanlah cara yang tepat.
Alasan:

1. Setiap komputer dengan arsitektur berbeda memiliki bahasa mesin yang berbeda → akibatnya, waktu setiap operasi antara satu komputer dengan komputer lain tidak sama.
2. Compiler bahasa pemrograman yang berbeda menghasilkan kode bahasa mesin yang berbeda → akibatnya, waktu setiap operasi antara compiler dengan compiler lain tidak sama.

Seperti yang dijelaskan pada pendahuluan, dipakailah suatu model abstrak yang mampu menggambarkan kebutuhan penggunaan waktu algoritma secara independen.

5.3 Kompleksitas Waktu

- Pekerjaan utama di dalam kompleksitas waktu adalah menghitung (counting) jumlah tahapan komputasi di dalam algoritma.
- Jumlah tahapan komputasi dihitung dari berapa kali suatu operasi dilakukan sebagai fungsi ukuran masukan (n).
- Di dalam sebuah algoritma terdapat banyak jenis operasi:
  - Operasi baca/tulis (contoh: input a, print a)
  - Operasi aritmetika (+, -, *, /) (contoh: a + b, M * N)
  - Operasi pengisian nilai (assignment) (contoh: a = 10)
  - Operasi perbandingan (contoh: a ¡ b, k ¿= 10)
  - Operasi pengaksesan elemen larik, pemanggilan prosedur/fungsi, dll
- Untuk menyederhanakan perhitungan, kita tidak menghitung semua jenis operasi yang ada di dalam sebuah algoritma, tetapi kita hanya fokus menghitung jumlah operasi yang khas (tipikal) yang mendasari algoritma tersebut. Beberapa di antaranya,
  - *Searching*, tipikal: Perbandingan elemen
  - *Sorting*, tipikal: Perbandingan dan pertukaran elemen
  - Perkalian dua buah matriks, tipikal: Perkalian dan penjumlahan
  - Menghitung nilai polinom, tipikal: Perkalian dan penjumlahan

Contoh 5.1: Kompleksitas Dalam Array

Tinjau algoritma menghitung rerata elemen di dalam sebuah larik (array) $a[1..n]$.

```
sum <- 0
for i <- 1 to n do
    sum <- sum + a[i]
endfor
rata_rata <- sum/n
```

Operasi yang mendasar pada algoritma tersebut adalah operasi penjumlahan elemen-elemen larik (yaitu $sum \leftarrow sum + a[i]$) yang dilakukan sebanyak n kali. Total kompleksitas waktunya, $T(n)$.

Contoh 5.2: Mencari Elemen Terbesar

Algoritma untuk mencari elemen terbesar di dalam sebuah larik (array) yang berukuran n elemen.

```
procedure CariElemenTerbesar(input a: array[1..n] of integer,
output maks: integer)

Kamus Lokal
    k: integer
Algoritma
    maks <- a[0]
    k <- 1
    while k <= n do
        if a[k] > maks then
            maks <- a[k]
        endif
        k <- k + 1
    endwhile
```

Kompleksitas waktu algoritma dihitung dari jumlah operasi perbandingan elemen larik ($a_k > maks$). Kompleksitas waktu CariElemenTerbesar : $T(n) = n–1$.

Tiga macam kompleksitas waktu:

1. $T_{max}(n)$: Kompleksitas waktu untuk kasus terburuk (maksimum)
2. $T_{avg}(n)$: Kompleksitas waktu untuk kasus rerata (rerata)
3. $T_{min}(n)$: Kompleksitas waktu untuk kasus terbaik (minimum)

Contoh 5.3: ***Sequential Search***

Akan ditunjukkan kompleksitas waktu dari *sequential search*.

```
procedure PencarianBeruntun(input a: array[1..n] of integer,
x : integer, output idx : integer)
    { Mencari elemen x di dalam larik a yang berisi n elemen.
    Jika x ditemukan, maka indeks elemen larik disimpan di dalam
    idx, idx bernilai -1 jika x tidak ditemukan }

Kamus Lokal
    k : integer
    found : boolean

Algoritma
    k <- 0
    found <- false
    while (k < n) and (not found) do
        if a[k] = x then
            ketemu <- true
        else
            k <- k + 1
        endif
    endwhile
    if found then
        idx <- k
    else
        idx <- -1
        endif
```

1. Kasus terbaik: Jika $a[0] = x$, maka $T_{min}(n) = 1$.
2. Kasus terburuk: Jika $a[n-1] = x$ atau x tidak ditemukan, maka $T_{max}(n) = n$.
3. Kasus rerata: Jika x ditemukan pada posisi ke-j, maka operasi perbandingan $(a[k] = x)$ akan dieksekusi sebanyak j kali.

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

5.4 Kompleksitas Waktu Asimptotik

- Seringkali kita kurang tertarik dengan kompleksitas waktu T(n) yang presisi untuk suatu algoritma.
- Kita lebih tertarik pada bagaimana kebutuhan waktu sebuah algoritma tumbuh ketika ukuran masukannya (n) meningkat.
- Contoh, sebuah algoritma memiliki jumlah operasi perkalian sebesar

  $$T(n) = 2n^2 + 6n + 1$$

  Kita mungkin tidak terlalu membutuhkan informasi seberapa presisi jumlah operasi perkalian di dalam algoritma tersebut. Yang kita butuhkan adalah seberapa cepat fungsi $T(n)$ tumbuh ketika ukuran data masukan membesar.
- Kinerja algoritma baru akan tampak untuk n yang sangat besar, bukan pada yang kecil. Kinerja algoritma-algoritma pengurutan seperti selection sort dan bubble sort misalnya, baru terlihat ketika mengurutkan larik berukuran besar, misalnya 10000 elemen.

- Oleh karena itu, kita memerlukan suatu notasi kompleksitas algoritma yang memperlihatkan kinerja algoritma untuk n yang besar. Notasi kompleksitas waktu algoritma untuk n yang besar dinamakan **kompleksitas waktu asimptotik**.
- Gagasannya adalah menghilangkan faktor koefisien pada $T(n)$

5.5 Notasi Big-O

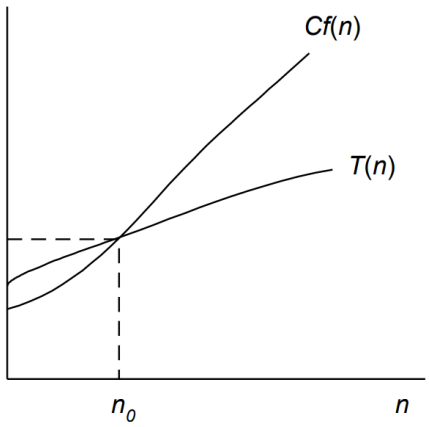
Definisi 5.2: Big-O

$T(n) = O(f(n))$ (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,

$$T(n) \leq Cf(n)$$

untuk $n \geq n_0$.

$f(n)$ adalah batas lebih atas (upper bound) dari $T(n)$ untuk n yang besar.



Fungsi $f(n)$ umumnya dipilih dari fungsi-fungsi standard seperti 1, n^2, n^3, ..., log n, n log n, 2^n, $n!$, dan sebagainya.

6

Figure 5.1: Visualisasi Big-O

Catatan: Ada tak hingga nilai C dan n_0 yang memenuhi $T(n) \leq Cf(n)$, kita cukup menunjukkan satu pasang (C, n_0) yang memenuhi definisi sehingga $T(n) = O(f(n))$.

Contoh 5.4: Tunjukkan bahwa $2n^2 + 6n + 1 = O(n^2)$

Tunjukkan bahwa $2n^2 + 6n + 1 = O(n^2)$!
Jawab: $2n^2 + 6n + 1 = O(n^2)$ karena

$$2n^2 + 6n + 1 \leq 2n^2 + 6n^2 + n^2 = 9n^2$$

untuk semua $n \geq 1$. $(C = 9,\ f(n) = n^2,\ n_0 = 1)$

Contoh 5.5: Tunjukkan bahwa $5 = O(1)$

Tunjukkan bahwa $5 = O(1)$!
Jawab: $5 = O(1)$ karena

$$5 \leq 6 \cdot 1$$

untuk semua $n \geq 1$. $(C = 6,\ f(n) = 1,\ n_0 = 1)$

Contoh 5.6: Tunjukkan bahwa $6 \cdot 2^n + 2n^2 = O(2^n)$

Tunjukkan bahwa $6 \cdot 2^n + 2n^2 = O(2^n)$!
Jawab: $6 \cdot 2^n + 2n^2 = O(2^n)$ karena

$$6 \cdot 2^n + 2n^2 \leq 6 \cdot 2^n + 2 \cdot 2^n = 8 \cdot 2^n$$

untuk semua $n \geq 4$. $(C = 8,\ f(n) = 2^n,\ n_0 = 4)$

Contoh 5.7: Tunjukkan bahwa $8n^2 = O(n^3)$

Tunjukkan bahwa $8n^2 = O(n^3)$!
Jawab: $8n^2 = O(n^3)$ karena

$$8n^2 \leq n^3$$

untuk semua $n \geq 8$. $(C = 1,\ f(n) = n^3,\ n_0 = 8)$

Teorema 5.1: Polinom

Bila $T(n) = a_m n^m + a_{m-1} n^{m-1} + \cdots + a_1 n + a_0$ adalah $a_1 n + a_0$ adalah polinom derajat $\leq n$, maka $T(n) = O(n^m)$

Cukup melihat pangkat terbesar dari $T(n)$. Teorema 1 tersebut digeneralisasi:

1. Eksponensial mendominasi sembarang perpangkatan (yaitu, $y^n > n^p, y > 1$)
2. Perpangkatan mendominasi $ln(n)$ (yaitu $n^p > \ln n$)
3. Semua logaritma tumbuh pada laju yang sama (yaitu $a \log n = b \log n$)
4. $n \log n$ tumbuh lebih cepat daripada n tetapi lebih lambat daripada n^2

Teorema 5.2

Misalkan $T_1(n) = O(f(n))$ dan $T_2(n) = O(g(n))$, maka

1. $T_1(n) + T_2(n) = O(f(n)) + O(g(n)) = O(\max(f(n), g(n)))$
2. $T_1(n)T_2(n) = O(f(n))O(g(n)) = O(f(n)g(n))$
3. $O(cf(n)) = O(f(n))$, c adalah konstanta
4. $f(n) = O(f(n))$

Contoh 5.8

Misalkan $T_1(n) = O(n)$ dan $T_2(n) = O(n^2)$, maka:

1. $T_1(n) + T_2(n) = O(\max(n, n^2)) = O(n^2)$
2. $T_1(n)T_2(n) = O(n \cdot n^2) = O(n^3)$

Contoh 5.9

$$O(5n^2) = O(n^2)$$
$$n^2 = O(n^2)$$

5.6 Notasi Big-Omega (Ω) dan Big-Theta (Θ)

Definisi 5.3

- $T(n) = \Omega(g(n))$: $T(n)$ berorde paling kecil $g(n)$ jika terdapat konstanta C dan n_0 sehingga $T(n) \geq Cg(n)$ untuk $n \geq n_0$.
- $T(n) = \Theta(h(n))$: $T(n)$ berorde sama dengan $h(n)$ jika $T(n) = O(h(n))$ dan $T(n) = \Omega(h(n))$.

Contoh 5.10

Contoh: Tunjukkan bahwa $2n^2 + 6n + 1 = \Theta(n^2)$.
Penyelesaian:
Dari definisi Big-O dan Big-Omega:
$2n^2 + 6n + 1 \leq 9n^2$ untuk $n \geq 1$ (Big-O).
$2n^2 + 6n + 1 \geq 2n^2$ untuk $n \geq 1$ (Big-Omega).
Maka, $2n^2 + 6n + 1 = \Theta(n^2)$.