Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung

Solusi Ujian Tengah Semester **IF2211 Strategi Algoritma**
Senin, 6 April 2026
Waktu: 100 menit
Dosen: Rinaldi M (K1), Rila Mandala (K2), Fazat Nur Azizah & Tricya Widagdo (K3)

Berdoalah terlebih dahulu agar Anda berhasil dalam mengerjakan ujian ini!

1. **(Brute Force + Greedy, Nilai = 15)** Seorang petugas kebersihan harus memroses *n* tugas. Setiap tugas memiliki durasi (*a*) dan waktu tenggat (*deadline*) (*d*) dan petugas harus memroses setiap tugas berturut-turut (satu tugas harus selesai terlebih dahulu sebelum memulai tugas berikutnya). Bonus untuk suatu tugas adalah selisih *d* - *f* dengan *d* adalah tenggat untuk tugas yang bersangkutan dan *f* adalah waktu penyelesaian tugas tersebut. Waktu dimulai dari 0 dan semua tugas harus diproses walaupun menghasilkan bonus negatif.
   (a) Buatlah algoritma *brute force* (bukan *pseudo-code*) untuk menentukan jumlah maksimum bonus yang dapat diterima dari pengerjaan semua tugas dengan masukan:
   n = banyaknya tugas
   Daftar tugas = {(a_1, d_1), (a_2, d_2), … (a_n,d_n)})}, dengan a_i adalah durasi penyelesaian tugas *i* dan d_i adalah waktu tenggat tugas *i*.
   Berikan ilustrasi algoritma Anda dengan menggunakan kasus berikut: *n* = 6, daftar tugas = {(6,10),(5,12),(4,19),(5,20),(4,17),(6,20 (Keterangan: (6,10) artinya durasi penyelessaian 6 jam dan harus selesai sebelum jam 10)
   Tentukan perkiraan kompleksitas algoritmanya dalam notasi big-O
   (b) Ulangi soal (a) di atas tetapi dengan pendekatan algoritma *greedy*. Berikan ilustrasi algoritma Anda dengan menggunakan kasus berikut: *n* = 6, daftar tugas = {(6,10),(5,12),(4,19),(5,20),(4,17),(6,20)}

Jawaban:

Diketahui:
n = banyaknya tugas
daftar tugas = P = {(a_1, d_1), (a_2, d_2), … (a_n,d_n)}
dengan a_i adalah durasi penyelesaian tugas i dan d_i adalah tenggat tugas i.
Fungsi objektif = memaksimumkan jumlah bonus dari pengerjaan semua tugas, yaitu:

$$\sum_{i=1}^{n} d_i - f_i$$

dengan f_i adalah waktu penyelesaian tugas ke-i

(a) Alternatif solusi dengan *brute-force* (*exhaustive search*)

Ide dasar: mencari permutasi n item yaitu kemungkinan urutan penyelesaian n tugas yang mungkin dan masing-masing dihitung total bonus yang diperoleh. Lalu dipilih total bonus yang terbesar.

Jumlah permutasi pemilihan r item dan n item adalah P(n,r) = n!/(n-r)!
Dalam hal ini karena r = n, maka P(n,n) = n!/(n-n)! = n!

Dengan tambahan waktu pemrosesan perhitungan bonus untuk n item, kompleksitas algoritma dengan pendekatan brute force adalah: O(n.n!) atau waktu pemrosesan perhitungan bonus dapat diabaikan sehingga kompleksitas waktunya adalah O(n!)

<u>Ilustrasi berdasarkan contoh:</u>
n = 6,
daftar tugas = {(6,10),(5,12),(4,19),(5,20),(4,17),(6,20)}

Dengan demikian, ada n! = 6! = 720 kemungkinan urutan pengerjaan tugas.

Karena jumlah yang sangat besar, diperbolehkan hanya memberikan beberapa ilustrasi saja dan tidak harus mengambil hasil yang optimal (atau bisa mengambil dari hasil algoritma greedy sebagai referensi). Minimum berikan 2 contoh.

Berikut diberikan 2 contoh:

Urutan Tugas	Tugas	Start	End	Bonus	Total Bonus
{1,2,3,4,5,6}	1 (6,10)	0	6	4	
	2 (5,12)	6	11	1	
	3 (4,19)	11	15	4	
	4 (5,20)	15	20	0	
	5 (4,17)	20	24	-7	
	6 (6,20)	24	30	-10	-8
{3,2,1,4,5,6}	3 (4,19)	0	4	15	
	2 (5,12)	4	9	3	
	1 (6,10)	9	15	-5	
	4 (5,20)	15	20	0	
	5 (4,17)	20	24	-7	
	6 (6,20)	24	30	-10	-4

Di antara kedua contoh tersebut, lebih optimal urutan tugas ke-2.

(b) Pendekatan dengan algoritma greedy.

Strategi greedy: kerjakan pekerjaan dengan durasi kecil terlebih dahulu.

Algoritma:

1. Urutkan daftar tugas secara terurut membesar berdasarkan nilai a_i. Jika terdapat lebih dari satu nilai a_i yang sama, tidak perlu mengurutkan nilai d_i (hasilnya akan sama dengan urutan apa pun).
2. Untuk tiap pekerjaan hitung waktu selesai (f_i) dan bonus (d_i-f_i) yang dihasilkan. Jumlahkan bonus dari bonus pekerjaan sebelumnya.
3. Jumlah bonus akhir adalah jumlah bonus yang diterima.

<u>Ilustrasi terhadap contoh kasus:</u>

n = 6,

daftar tugas = {(6,10),(5,12),(4,19),(5,20),(4,17),(6,20)}

1. Urutkan daftar tugas terurut membesar berdasarkan nilai a_i.
   Hasil pengurutan daftar tugas = {(4,19),{4,17),(5,12),(5,20),(6,10),(6,20)}
2. Pengerjaan tugas dan perhitungan bonus:

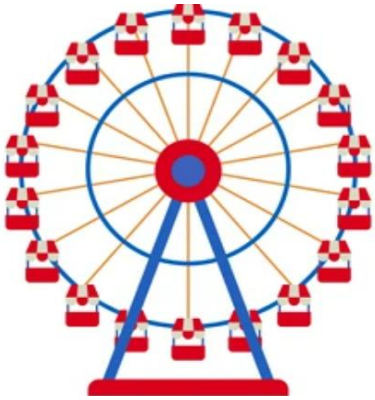
Tugas	Start	End	Bonus	Total Bonus
3 (4,19)	0	4	15	
5 (4,17)	4	8	9	
2 (5,12)	8	13	-1	
4 (5,20)	13	18	2	
1 (6,10)	18	24	-14	
6 (6,20)	24	30	-10	1

3. Total bonus yang didapatkan = 1

Kompleksitas algoritma:

- Kompleksitas waktu pengurutan: $O(n^2)$ atau O(n log n) tergantung algoritma yang digunakan
- Kompleksitas waktu untuk penghitungan bonus (dan total bonus) adalah O(n)

2. (**Greedy, Nilai = 15**) Setiap tahun Pak Budi mengajak siswa dari kelasnya untuk mengunjungi suatu taman bermain. Salah satu permainan yang sangat disukai adalah kincir ria (*ferris wheel*). Setiap gondola di kincir ria bisa dinaiki maksimum 4 orang dan total berat penumpang untuk tiap gondola tidak boleh melebihi suatu nilai *x*. Biaya karcis untuk sebuah gondola sangat mahal sehingga Pak Budi ingin sesedikit mungkin menyewa gondola untuk siswa-siswanya. Buatlah algoritma dengan pendekatan *greedy* untuk menghitung jumlah minimum gondola yang dibutuhkan untuk mengangkut semua siswa, jika diberikan: banyaknya siswa *n*, berat maksimum per gondola *x*, dan daftar berat siswa $w_1, w_2, \ldots w_n$.



Jelaskan ilustrasi algoritma Anda untuk kasus berikut: n = 10; x = 55. Daftar berat badan siswa = [15, 20, 10, 10, 17, 18, 15, 25, 14, 16]. Jelaskan apakah algoritma akan memberikan hasil yang optimal? Buktikan!

Jawaban:

ALTERNATIF SOLUSI:
Algoritma dengan pendekatan greedy
Diketahui:
Banyaknya siswa, n
Berat maksimum sebuah gondola, x
Jumlah penumpang sebuah gondola, y dengan nilai y maksimum, $y_{max} = 4$
Himpunan berat badan siswa, W = [w_1, w_2, ..., w_n] dengan w_i adalah berat seorang siswa, dengan $w_i <= x$ (tambahan asumsi). Total berat badan seluruh siswa, Wtot adalah:

$$Wtot = \sum_{i=1}^{n} w_i$$

Objektif = minimumkan m, yaitu banyaknya gondola yang dibutuhkan untuk mengangkut semua siswa, jika total berat seluruh siswa, Wtot, adalah

$$Wtot = \sum_{i=1}^{m} \sum_{j=1}^{y} w_j$$

dengan kendala $y <= y_{max}$ dan $\sum_{j=1}^{y} w_j <= x$

Strategi greedy: meminimumkan jumlah gondola yang disewa dengan memilih siswa yang terberat terlebih dahulu, menempatkan siswa pada gondola yang masih muat untuk siswa tersebut dimulai dari gondola paling awal.

Algoritma:

1. Urutkan daftar berat badan siswa secara terurut membesar, misalnya W‘ = [w'_1, w'_2, ..., w'_n]
2. Misal jumlah gondola yang dibutuhkan adalah m dengan nilai awal m = 0, berat seorang siswa adalah w'_i, total berat penumpang gondola j adalah g_j, dan jumlah penumpang di gondola j adalah c_i. Nilai awal $g_j = 0$ dan $c_j = 0$.
   Mulai dari w'_1:
   2.1. Mulai dari j = 1
   2.2. Jika $g_j + w'_i <= x$ dan $c_j + 1 <= y_{max}$, tempatkan siswa dengan berat w'_i ke gondola j. Update $g_j = g_j + w'_i$ dan $c_j = c_j + 1$.
   2.3. Jika $g_j + w'_i > x$ atau $c_j + 1 > y_{max}$, maka maju ke gondola j+1 dengan total berat penumpang g_{j+1} dan jumlah penumpang c_{j+1} dan ulangi langkah 2.2 dan 2.3 sampai siswa berhasil ditempatkan.
   Ulangi langkah 2.1 dan 2.3 untuk setiap elemen W‘ secara terurut.
3. Jumlah gondola dibutuhkan, m = nilai j terbesar.

Ilustrasi berdasarkan contoh kasus:

Diketahui:

n = 10

x = 55

$y_{max} = 4$

W = [15, 20, 10, 10, 17, 18, 15, 25, 14, 16]

Algoritma:

1. Urutkan daftar berat badan siswa secara terurut mengecil, misalnya W‘

Didapatkan W‘ = [25, 20, 18, 17, 16, 15, 15, 14, 10, 10]

2. Tiap langkah diperiksa mulai dari j = 1. Nilai awal $g_1 = 0$ dan $c_1 = 0$. Cek tiap elemen W‘:
   i. $w'_1 = 25$, karena $g_1 + w'_1 = 0 + 25 = 25 <= 55$ dan $c_1 + 1 = 0 + 1 = 1 <= 4$ maka w'_1 ditempatkan di gondola j = 1.
   Kondisi terkini: Gondola-1 = [25]
   ii. $w'_2 = 20$, karena $g_1 + w'_2 = 25 + 20 = 45 <= 55$ dan $c_1 + 1 = 1 + 1 = 2 <= 4$ maka w'_2 ditempatkan di gondola j = 1.
   Kondisi terkini: Gondola-1 = [25,20]
   iii. $w'_3 = 18$, karena $g_1 + w'_3 = 45 + 18 = 63 > 55$, maka maju ke gondola j = 2 dengan $g_2 = 0$ dan $c_2 = 0$. Karena $g_2 + w'_3 = 0 + 18 = 18 <= 55$ dan $c_2 + 1 = 0 + 1 = 1 <= 4$, maka w'_2 ditempatkan di gondola j = 2.
   Kondisi terkini: Gondola-1 = [25,20]; Gondola-2 = [18]
   iv. $w'_4 = 17$, karena $g_1 + w'_4 = 45 + 17 = 62 > 55$, maka maju ke gondola j = 2 dengan $g_2 = 18$ dan $c_2 = 1$. Karena $g_2 + w'_4 = 18 + 17 = 35 <= 55$ dan $c_2 + 1 = 1 + 1 = 2 <= 4$, maka w'_4 ditempatkan di gondola j = 2.
   Kondisi terkini: Gondola-1 = [25,20]; Gondola-2 = [18,17]
   v. $w'_5 = 16$, karena $g_1 + w'_5 = 45 + 16 = 61 > 55$, maka maju ke gondola j = 2 dengan $g_2 = 35$ dan $c_2 = 2$. Karena $g_2 + w'_5 = 35 + 16 = 51 <= 55$ dan $c_2 + 1 = 2 + 1 = 3 <= 4$, maka w'_5 ditempatkan di gondola j = 2.
   Kondisi terkini: Gondola-1 = [25,20]; Gondola-2 = [18,17,16]
   vi. $w'_6 = 15$, karena $g_1 + w'_6 = 45 + 15 = 60 > 55$, maka maju ke gondola j = 2 dengan $g_2 = 51$ dan $c_2 = 3$. Karena $g_2 + w'_6 = 51 + 15 = 66 > 55$, maka maju ke gondola j = 3 dengan $g_3 = 0$ dan $c_3 = 0$. Karena $g_3 + w'_6 = 0 + 15 = 15 <= 55$ dan $c_3 + 1 = 0 + 1 = 1 <= 4$, maka w'_6 ditempatkan di gondola j = 3.
   Kondisi terkini: Gondola-1 = [25,20]; Gondola-2 = [18,17,16]; Gondola-3 = [15]
   vii. $w'_7 = 15$, karena $g_1 + w'_7 = 45 + 15 = 60 > 55$, maka maju ke gondola j = 2 dengan $g_2 = 51$ dan $c_2 = 3$. Karena $g_2 + w'_7 = 51 + 15 = 66 > 55$, maka maju ke gondola j = 3 dengan $g_3 = 15$ dan $c_3 = 1$. Karena $g_3 + w'_7 = 15 + 15 = 30 <= 55$ dan $c_3 + 1 = 1 + 1 = 2 <= 4$, maka w'_7 ditempatkan di gondola j = 3.
   Kondisi terkini: Gondola-1 = [25,20]; Gondola-2 = [18,17,16]; Gondola-3 = [15,15]
   viii. $w'_8 = 14$, karena $g_1 + w'_8 = 45 + 14 = 59 > 55$, maka maju ke gondola j = 2 dengan $g_2 = 51$ dan $c_2 = 3$. Karena $g_2 + w'_8 = 51 + 14 = 65 > 55$, maka maju ke gondola j = 3 dengan $g_3 = 30$ dan $c_3 = 2$. Karena $g_3 + w'_8 = 30 + 14 = 44 <= 55$ dan $c_3 + 1 = 2 + 1 = 3 <= 4$, maka w'_8 ditempatkan di gondola j = 3.
   Kondisi terkini: Gondola-1 = [25,20]; Gondola-2 = [18,17,16]; Gondola-3 = [15,15,14]
   ix. $w'_9 = 10$, karena $g_1 + w'_8 = 45 + 10 = 55 <= 55$ dan $c_1 + 1 = 2 + 1 = 3 <= 4$, maka w'_9 ditempatkan di gondola j = 1.
   Kondisi terkini: Gondola-1 = [25,20,10]; Gondola-2 = [18,17,16]; Gondola-3 = [15,15,14]
   x. $w'_{10} = 10$, karena $g_1 + w'_{10} = 55 + 10 = 65 > 55$, maka maju ke gondola j = 2 dengan $g_2 = 51$ dan $c_2 = 3$. Karena $g_2 + w'_{10} = 51 + 10 = 61 > 55$, maka maju ke gondola j = 3 dengan $g_3 = 44$ dan $c_3 = 3$. Karena $g_3 + w'_{10} = 44 + 10 = 54 <= 55$ dan $c_3 + 1 = 3 + 1 = 4 <= 4$, maka w'_{10} ditempatkan di gondola j = 3.
   Kondisi terkini: Gondola-1 = [25,20,10]; Gondola-2 = [18,17,16]; Gondola-3 = [15,15,14,10]

3. Jumlah gondola yang dibutuhkan adalah nilai j terbesar yaitu m = 3, dengan masing-masing gondola berisi :

Gondola-1 = [25,20,10], $g_1 = 55$, $c_1 = 3$

Gondola-2 = [18,17,16], $g_2 = 51$, $c_2 = 3$

Gondola-3 = [15,15,14,10] , $g_3 = 54$, $c_3 = 4$

Untuk kasus di atas m = 3 adalah jumlah minimum gondola yang dibutuhkan.

Karena jika n mod $y_{max} > 0$, maka minimum jumlah gondola yang dibutuhkan adalah $m_{min} = (n \text{ div } y_{max}) + 1$ dan tidak mungkin jumlah gondola lebih kecil dari m_{min} (malah mungkin lebih besar dalam kasus Wtot > x * m_{min}).

Dalam hal ini:

n mod $y_{max} = 10$ mod $4 = 2 > 0$, maka $m_{min} = (n \text{ div } y_{max}) + 1 = (10 \text{ div } 4) + 1 = 3$.

Tidak mungkin jumlah gondola yang dibutuhkan akan < 3, jadi utk kasus di atas m = 3 optimal.

Apakah solusi algoritma greedy di atas optimal?
Walaupun solusi pada contoh mencapai hasil optimal, terdapat *counter example* yang menunjukkan bahwa pada kasus lain, algoritma di atas tidak optimal, misalnya spt contoh berikut.

Diketahui:
n = 6
x = 55
y_{max} = 4
W = [20,17,18,25,14,16]
Setelah diurutkan menjadi: W' = [25,20,18,17,16,14]

Dengan menggunakan algoritma di atas didapatkan m = 3 dengan komposisi gondola:
Gondola-1 = [25,20], g_1 = 45, c_1 = 2
Gondola-2 = [18,17,16], g_2 = 51, c_2 = 3
Gondola-3 = [14] , g_3 = 14, c_3 = 1

Padahal terdapat solusi yang lebih optimal untuk kasus tersebut, dengan m = 2, misalnya sbb.
Gondola-1 = [25,16,14], g_1 = 55, c_1 = 3
Gondola-2 = [20,18,17], g_2 = 55, c_2 = 3

Untuk kasus di atas m = 2 adalah jumlah minimum gondola yang dibutuhkan (tidak mungkin kurang dari 2 gondola).
n mod y_{max} = 6 mod 4 = 2 > 0, maka m_{min} = (n div y_{max}) + 1 = (6 div 4) + 1 = 2.
Karena ditunjukkan ada counter example yang menghasilkan m = 2, solusi algoritma greedy di atas tidak optimal.

3. **(Divide and Conquer, Nilai = 12)** Diketahui sebuah persoalan *rahasia*. Persoalan *rahasia* tersebut diselesaikan dengan menggunakan algoritma *divide and conquer* sebagai berikut

```
Procedure rahasia (input n : integer)
Algoritma
    if n =1 solve()
    else
        rahasia(n/2)
        rahasia(n/4)
        rahasia(n/4)
        berubah()
    end if
```

Jika kompleksitas waktu dari procedure *solve()* adalah 1 dan kompleksitas dari procedure *berubah()* adalah *n*, jawablah pertanyaan di bawah ini :

(a) Tuliskan kompleksitas waktunya *T*(*n*) secara rekuren.
(b) Jika tiap simpul dalam pohon rekurensi tersebut adalah cost untuk menghitung procedure *berubah()* nya saja, Gambarkan pohon rekurensinya dengan *root* adalah persoalan dengan ukuran *n* (sehingga *cost* di level *root* adalah *n*) dan anak dari setiap simpul adalah ekspansi dari rekurensinya. Gambarkan pohonnya hanya sebanyak 3 level saja (level *root*, level *root*+1 dan level *root*+2).
(c) Jika tiap simpul dalam pohon rekurensi tersebut adalah *cost* untuk menghitung procedure *berubah()* nya saja, hitunglah *cost* nyatanya untuk di level *root*, level *root*+1, dan level *root*+2. (kalian harus menulis dengan penurunannya, bukan hanya jawaban akhirnya saja).
(d) Hitunglah kompleksitas *T*(*n*) akhirnya setelah penjabaran rekurensi nya. (kalian harus menulis dengan penurunannya, bukan hanya jawaban akhirnya saja).

Jawaban:

(a) Relasi rekurens time complexity nya adalah :

T(n) = 1 jika n = 1.

= T(n/2) + 2T(n/4) + n jika n > 1

(b) pohon rekurensinya adalah :

<u>level root :</u>

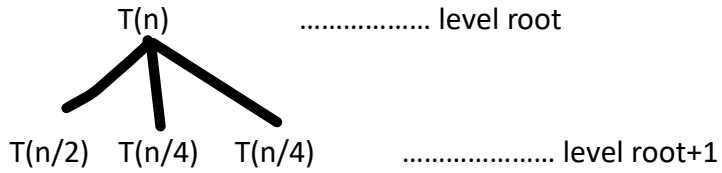
Ukuran: n

Node: 1

<u>level root+1 :</u>

Dari T(n):

- 1 node → T(n/2)
- 2 node → T(n/4)



<u>Level root+2</u>

Ekspansi masing-masing:

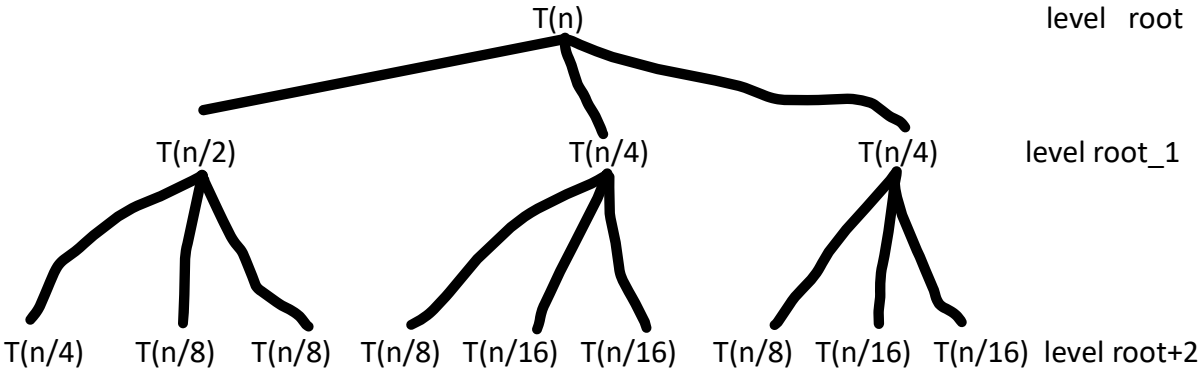
Level root+2

Dari T(n/2):

→ T(n/4), T(n/8), T(n/8)

Dari masing-masing T(n/4):

→ T(n/8), T(n/16), T(n/16)



(c) Cost tiap level (hanya biaya berubah() = ukuran submasalah)

Level root :

- 1 node ukuran n: cost $= n$

Level root+1:

- T(n/2): cost = n/2
- 2 × T(n/4): cost = 2 × (n/4) = n/2

→Total cost level root$+1 = n/2 + n/2 = n$

Level root+2:
Hitung cost satu per satu:
Dari T(n/2):

- T(n/4) → cost : n/4
- 2 × T(n/8) →cost : 2 × (n/8) = n/4
  → total cost = n/4 + n/4 = n/2

Dari 2 × T(n/4):
Setiap T(n/4):

- T(n/8) → cost : n/8
- 2 × T(n/16) → cost : 2 × (n/16) = n/8
  → total per node = n/8 + n/8 = n/4

Karena ada 2 node:
→ total = 2 × (n/4) = n/2

Total Level root+2: $n/2 + n/2 = n$

(d) Dari pohon rekurensi:

- Setiap level cost = n
- Tinggi pohon = ?

Ukuran masalah terkecil mengikuti jalur terbesar pembagian:

$$n \to n/2 \to n/4 \to n/8 \to \cdots \to 1$$

Tinggi pohon: $\log n$

Total biaya: $T(n) = n + n + n + \cdots$ ($\log\ n$ level) = $n \cdot \log\ n$

4. **(Decrease and Conquer, Nilai = 13)** Diketahui sebuah program misterius yang menyelesaikan persoalan misterius dengan menggunakan algoritma *decrease and conquer* sbb :

```
procedure Misterius (input A : LarikInteger,  i, j : integer; K : integer; output id : integer)
Kamus
    p : integer

Algoritma:
    if i > j then
        id ← −1
    else
        p ← i + (j − i) *(K − A(i))/ (A(j) − A (i))
        if A(p) = K then
            id ← p
        else
            if A(p) > K then
                Misterius(A, i, p − 1, K, id)
            else
                Misterius(A, p + 1, j, K, id)
            endif
        endif
    endif
```

(a) Hitunglah *time complexity* dalam notasi Big-O dari procedure *Misterius* di atas untuk kasus terbaik.
(b) Hitunglah *time complexity* dalam notasi Big-O dari procedure *Misterius* di atas untuk kasus terburuk.
(c) Hitunglah *time complexity* dalam notasi Big-O dari procedure *Misterius* di atas untuk kasus rata-rata.

Jawaban:

a. Kasus terbaik: O(1), jika nilai variable K = nilai larik A[p]

b. Kasus terbaik: O(n), jika data sangat tidak merata, misalkan nilai A[j] nilai yang sangat besar jauh melampaui nilai elemen larik A yang lainnya, dan nilai variable K = nilai A[j]

c. Kasus rata-rata :

- Jika data terdistribusi uniform, maka perkiraan posisi sangat akurat. Akibatnya ukuran masalah menyusut sangat cepat.
- Pada *binary search*, ukuran larik berkurang setengah kali ukuran sebelumnya :

  $n \rightarrow \frac{n}{2} \rightarrow \frac{n}{4} \rightarrow \ldots$ (Jumlah langkah: 2log n kali)
- Pada *interpolation search*, karena posisi diprediksi berdasarkan nilai, ukuran larik menyusut kira-kira menjadi: $\sqrt{n}$

sehingga: $n \rightarrow \sqrt{n} \rightarrow \sqrt{\sqrt{n}} \rightarrow \ldots$

- Misalkan ukuran larik setelah *k* Langkah adalah $n^{(1/2^k)}$

Pencarian selesai saat ukuran larik mendekati 1, ambil sekitar 2

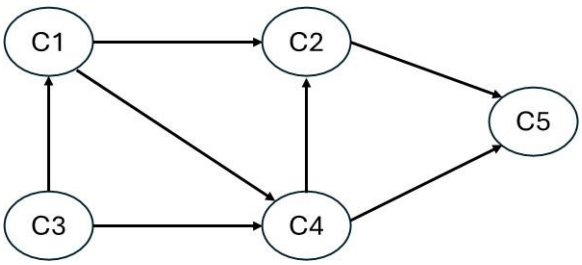
$$\rightarrow n^{(1/2^k)} \approx 2 \rightarrow {}^2\log(n^{(1/2^k)}) \approx 2\log 2 \rightarrow \frac{1}{2^k}\ 2\log n \approx 1 \rightarrow 2^k \approx 2\log n \rightarrow k \approx \log(\log n)$$

Sehingga kompleksitasnya: $O(\log \log n)$

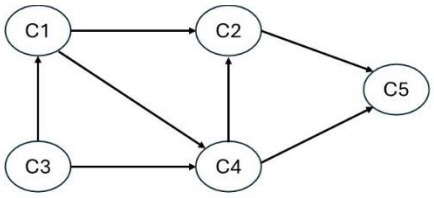
5. **(Decrease and Conque**r, **Nilai = 15)** Sebuah kode_kuliah mungkin memiliki nol atau lebih prasyarat kuliah (*prerequisite*). Kode_kuliah bisa diambil pada suatu semester jika semua prasyaratnya sudah pernah diambil pada semester sebelumnya (tidak harus 1 semester sebelumnya, bisa saja 2 semester sebelumnya). Asumsi semua kuliah bisa diambil pada sembarang semester, baik semester ganjil maupun semester genap. Sebagai contoh, terdapat 5 kuliah: C1, C2, C3, C4, dan C5, yang harus diambil seorang mahasiswa dengan daftar *prerequisite* sebagai berikut:
   C1, Prerequisite: C3.
   C2, Prerequisite: C1, C4.
   C3, Prerequisite: tidak ada
   C4, Prerequsite: C1, C3.
   C5, Prerequisite: C2, C4..
   (a) Gambarkan graf *Directed Acyclic Grap*h (DAG) yang merepresentasikan kelima kuliah di atas
   (b) Lakukan *topological sort* untuk menentukan kuliah apa saja yang diambil pada setiap semester (semester 1, 2, 3, 4, dst). Perlihatkan proses *topological sort* berdasarkan graf dari jawaban (a) di atas

Jawaban:

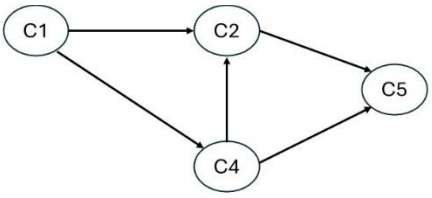
(a)



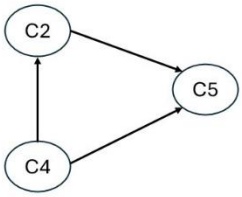
(c) Buang simpul dengan derajat-masuk 0 secara berturut-turut sbb:



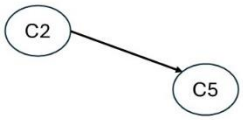
(1) Graf awal



(2) Hapus C3



(3) Hapus C1



(4) Hapus C4

(5) Hapus C2

Hasil topological sort:

Semester 1: C3
Semester 2: C1
Semester 3: C4
Semester 4: C2
Semester 5: C5

6. **(Divide and Conquer, Nilai = 15)** Diberikan sebuah matriks integer ukuran n x n, asumsikan n merupakan perpangkatan dua. Kita akan menghitung selisih nilai maksimum dan nilai minimum matriks dengan algoritma divide and conquer
   (a) Tuliskan garis besar algoritmanya (boleh dalam *pseudo-code*)
   (b) Tuliskan kompleksitas waktunya, $T(n)$, dalam bentuk relasi rekurens
   (c) Hitung notasi Big-Oh dengan menggunakan Teorema Master

Jawaban:

(a) Algoritma divide and conquer:

Misalkan A adalah matriks berukuran n x n

Jika n = 1, maka min=max = A(n,n)

Jika n > 1 maka

(i) bagi matriks A menjadi 4 buah matriks berukuran n/2 x n/2, misalkan A1, A2, A3, dan A4
(ii) min1, max1 ← cari min dan max pada matriks A1 berukuran n/2 x n/2
(iii) min2, max2 ← cari min dan max pada matriks A2 berukuran n/2 x n/2
(iv) min3, max3 ← cari min dan max pada matriks A3 berukuran n/2 x n/2
(v) min4, max4 ← cari min dan max pada matriks A4 berukuran n/2 x n/2
(vi) min ← MIN(min,min2, min3, min4)
(vii) max ← MAX(max1, max2, max3, max4)

atau dalam pseudo-code:

```
function MinMax(A, n, n)
if n = 1 then
  return (A[n,n], A[n,n])
else
   bagi matriks menjadi 4 bagian: A1, A2, A3, A4
   (min1, max1) ← MinMax(A1, n/2, n/2)
   (min2, max2) ← MinMax(A2, n/2, n/2)
   (min3, max3) ← MinMax(A3, n/2, n/2)
   (min4, max4) ← MinMax(A4, n/2, n/2)

   min_val ← min(min1, min2, min3, min4)
   max_val ← max(max1, max2, max3, max4)

   return (min_val, max_val)
```

Pemanggilan fungsi:
[min_val, max_val] ← MinMax(A, n, n)
selisih = max_val – min_val

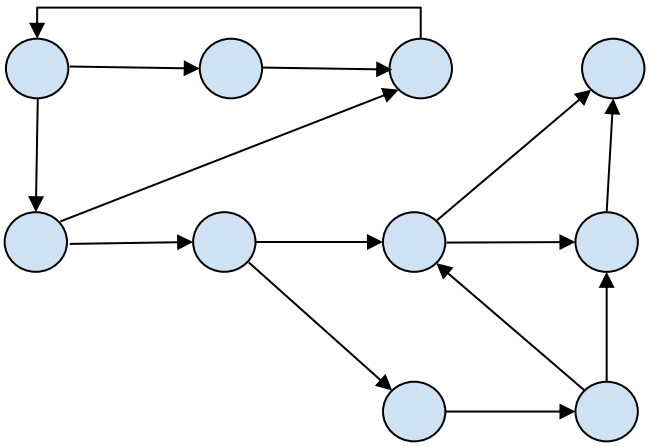
(b) T(n) = c, n = 1
= 4T(n/2) + d, n > 1

(c) Teorema Master: T(n) adalah $\begin{cases} O(n^d) & \text{jika } a < b^d \\ O(n^d \log n) & \text{jika } a = b^d \\ O(n^{\log_b a}) & \text{jika } a > b^d \end{cases}$

T(n) = 4T(n/2) + d $\rightarrow$ a = 4, b = 2, d = 0 $\rightarrow$ a > 2^0 $\rightarrow$ case 3 $\rightarrow$ O($n^{\log 4}$) = O(n^2)

7. **(BFS, DFS, dan DLS, Nilai = 15)** Berikut ini adalah senarai ketetanggaan yang memuat informasi graf berarah.
*Elemen **x : y, z** menyatakan terdapat sisi dari simpul x ke simpul y dan simpul z.*
{ A : B, C; B : E; C : D, E; D : F, G; E : A; F : H; G : I, J; H : G, I; I : J }
   (a) Gambarkan graf berarah sesuai informasi pada senarai ketetanggaan tersebut.
   (b) Jika dari simpul A ingin menuju simpul G, tentukan **jalur penelusuran/pencarian** menggunakan ***masing-masing algoritma berikut ini*** dan **tentukan jalur hasil** pencarian untuk masing-masing algoritma. Perhatikan, jika terdapat lebih dari satu simpul tetangga, pemilihan simpul sesuai urutan abjad.
      i. BFS
      ii. DFS
      iii. DLS dengan batas kedalaman 4

Jawaban:
(a) Gambar graf berarah:



(b) Jalur penelusuran/pencarian dan jalur hasil pencarian dari simpul A ke F:

i. BFS:
Jalur penelusuran: A, B, C, E, D, F, G (jika dituliskan dalam bentuk pohon, harus jelas urutan pembangkitan simpulnya).
Jalur hasil: A - C - D - G

ii. DFS:
Jalur penelusuran: A, B, E, C, D, F, H, G (jika dituliskan dalam bentuk pohon, harus jelas urutan pembangkitan simpulnya).
Jalur hasil: A - C - D - E - F - G

iii. DLS dengan batas kedalaman 4:
Jalur penelusuran: A, B, E, C, D, F, H, G (jika dituliskan dalam bentuk pohon, harus jelas urutan pembangkitan simpulnya).
Jalur hasil: A - C - D - G