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

Nama :……
NIM/Kelas:……
T.tangan :……

Solusi Kuis ke-4 IF1220 Matematika Diskrit (3 SKS) – Graf, Pohon, dan Kompleksitas Algoritma
Dosen: Rinaldi M
Selasa, 26 Mei 2026
Waktu: 90 menit

1. **(Nilai 10)** Sebuah negara memiliki 10 bandara. Pemerintah ingin membangun jalur penerbangan langsung sehingga setiap bandara terhubung langsung dengan tepat 5 bandara lainnya.
   (a) Berapa jumlah jalur penerbangan yang terbentuk?
   (b) Apakah jaringan penerbangan seperti ini dapat dibuat? Jelaskan menggunakan *lemma* jabat tangan.
   (c) Tanpa menggambarkan grafnya, apakah jaringan ini planar? Jelaskan dengan teori graf.

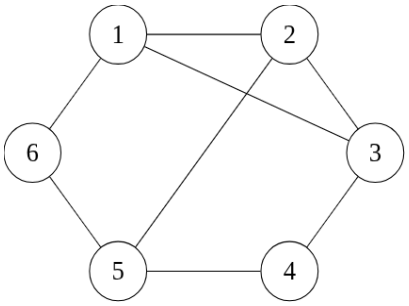
Solusi:

(a) Jumlah derajat = 10 x 5 = 50.
Jumlah derajat = 2 x jumlah sisi
50 = 2e
e = 25

(b) Jumlah total derajat genap jadi graf (yang merepresentasikan jaringan penerbangan) dapat dibuat

(c) $e \leq 3n - 6$
$25 \leq 3(10) - 6$
$25 \leq 24$ (Tidak Memenuhi, tidak planar)

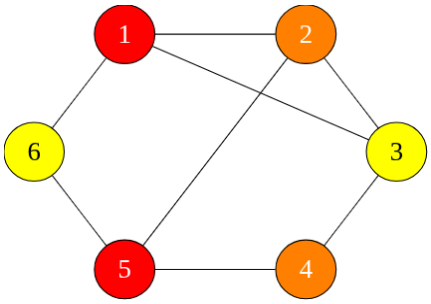
2. **(Nilai 10)** Diberikan sebuah graf G seperti gambar di samping
   (a) Apakah graf G tersebut merupakan graf bipartit? Jelaskan alasannya.
   (b) Tentukan bilangan kromatik ($\chi(G)$) dari graf G dan berikan satu contoh kombinasi pembagian warnanya (gunakan warna mejikuhibiniu)!



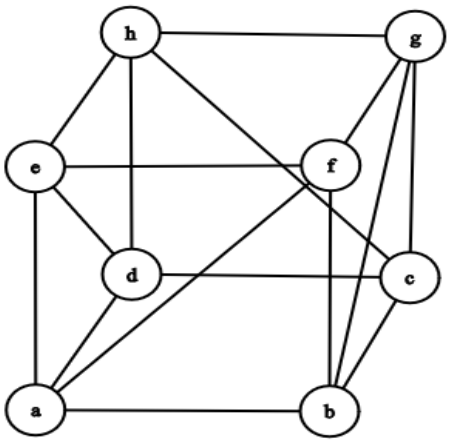
Solusi:

(a) Bukan graf bipartit. Ada banyak alasannya. Salah satunya, sebuah graf dapat disebut graf bipartit jika tidak mengandung sirkuit dengan panjang ganjil (atau bilangan kromatiknya <= 2). Pada graf ini, terdapat sirkuit dengan panjang 3 (ganjil), yaitu sirkuit 1 - 2 - 3 - 1. Oleh karena itu, graf ini bukan graf bipartit. Alasan lainnya, tidak mungkin digambarkan menjadi graf dengan dua buah himpunan simpul V1 dan V2 sedemikian sehingga simpul-simpul di V1 hanya bertetangga dengan simpul-simpul di V2

(b) $\chi(G) = 3$

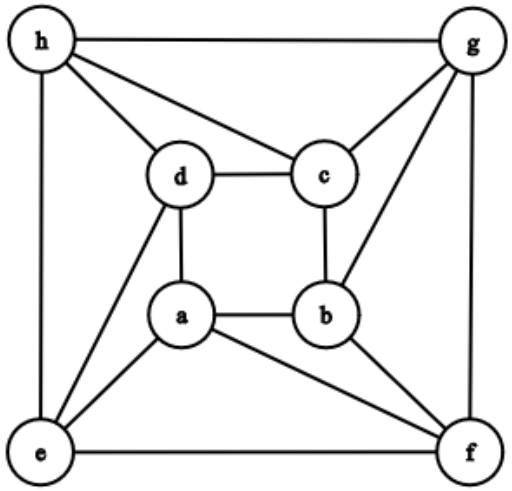


3. **(Nilai 10)** Apakah graf G di kanan ini merupakan graf planar? Jika ya, gambarkanlah dalam bentuk planar. Jika tidak, buktikan dengan teorema Kuratowski.

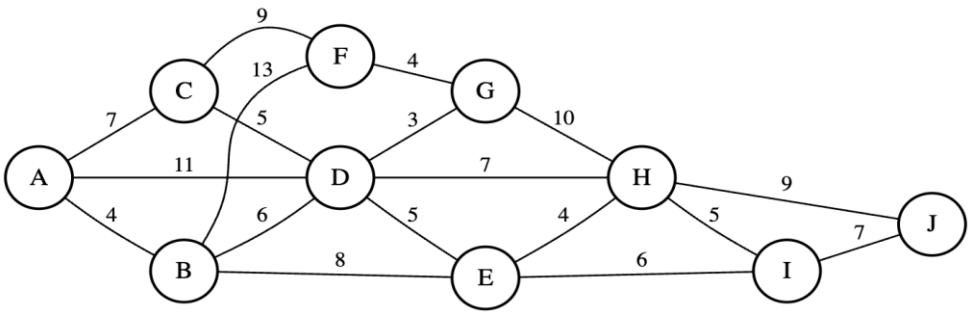


Solusi:

Graf tersebut merupakan graf planar. Graf tersebut dapat diubah menjadi graf berikut. Anggap simpul-simpul merupakan sudut-sudut pada kubus. Lihat kubus dari sisi atas dan graf di bawah ini adalah tampaknya jika dilihat dari sisi atas.

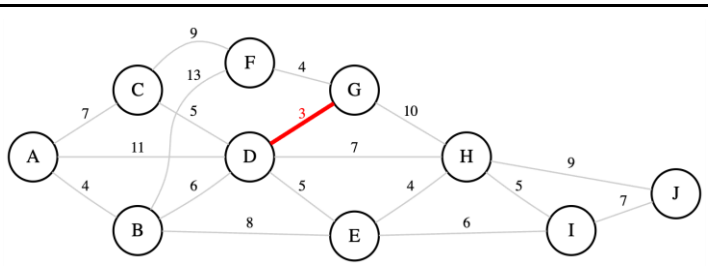
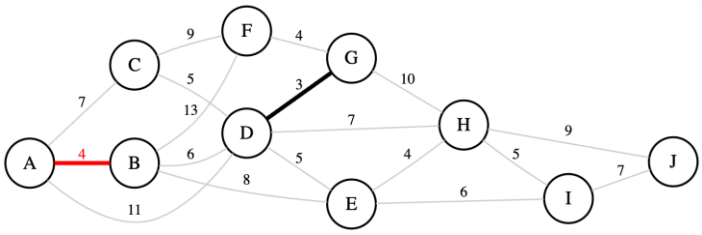
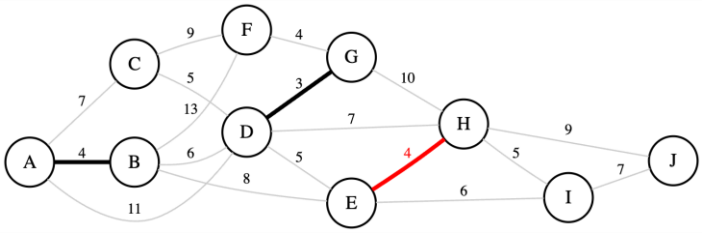
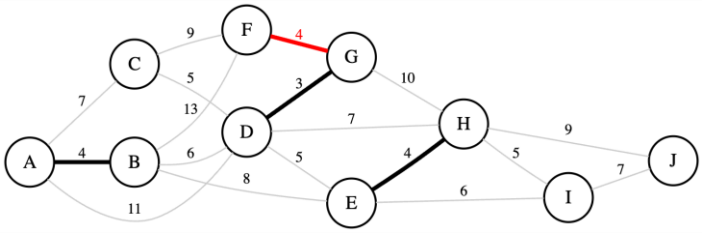
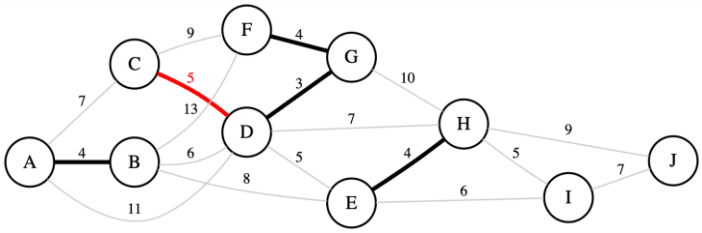
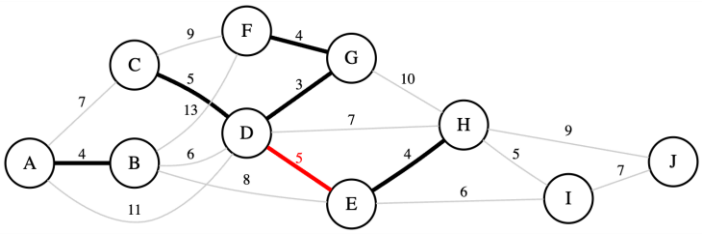


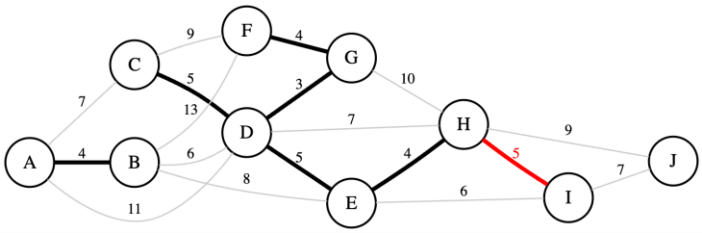
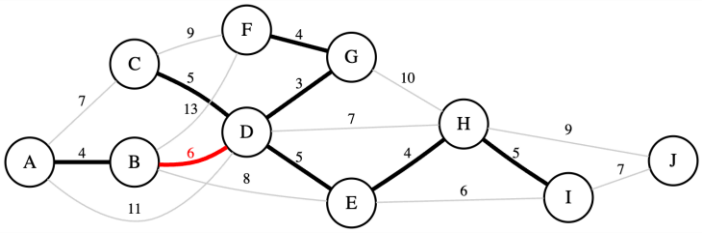
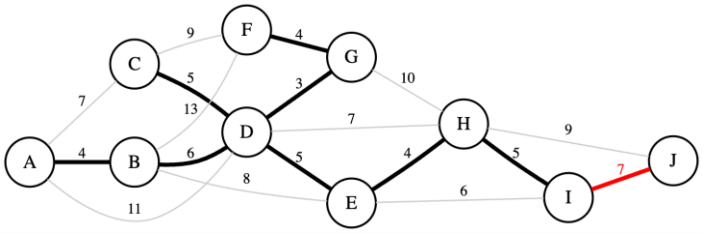
4. **(Nilai 15)** Tentukanlah pohon merentang minimum serta bobot total akhir dari graf pada Gambar di bawah dengan menggunakan Algoritma Kruskal! Untuk setiap langkah, tuliskan sisi yang ditambahkan, bobot dari sisi tersebut, beserta gambar pohon rentang yang terbentuk sampai dengan langkah tersebut. Jika terdapat beberapa sisi dengan bobot yang sama, pilihlah sisi yang nama simpul-simpulnya lebih awal secara alfabetis (misalnya, A-B dipilih sebelum A-C, dan A-C dipilih sebelum B-C). Jawaban tanpa langkah yang lengkap tidak akan mendapatkan nilai penuh.



Solusi:

Sisi	D-G	A-B	E-H	F-G	C-D	D-E	H-I	B-D	E-I	A-C	D-H	I-J	B-E	C-F	H-J	G-H	A-D	B-F
Bobot	3	4	4	4	5	5	5	6	6	7	7	7	8	9	9	1O	11	13

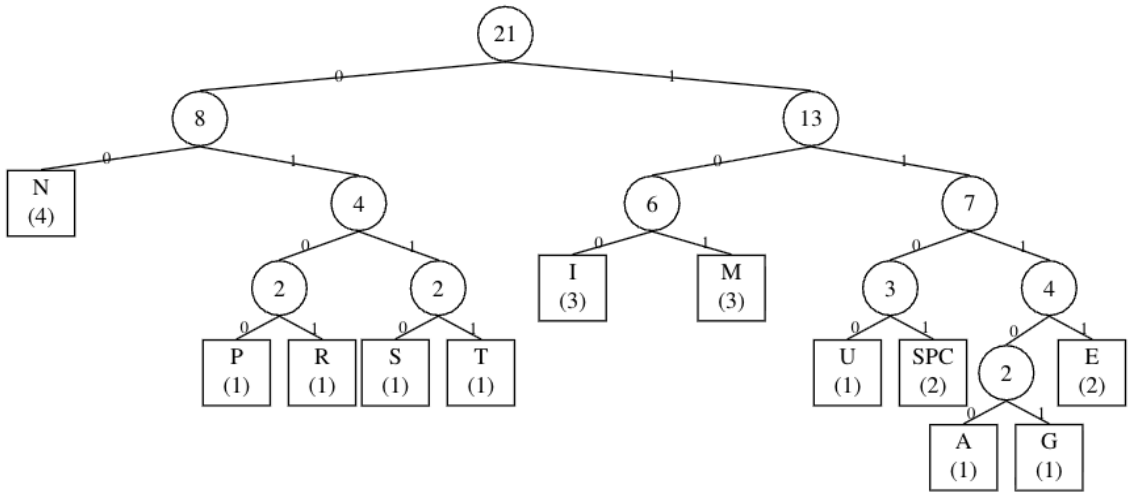
Langkah	Sisi	Bobot	Pohon Rentang
1	D-G	3	
2	A-B	4	
3	E-H	4	
4	F-G	4	
5	C-D	5	
6	D-E	5	

7	H-I	5	
8	B-D	6	
9	I-J	7	

Bobot total: 3+4+4+4+5+5+5+6+7=43

5. **(Nilai 15)** Budi ingin mengirimkan spesifikasi algoritma kepada rekannya secara efisien menggunakan Kode Huffman. Jika pesan yang dikirim adalah 'MINIMUM SPANNING TREE', tentukan kode Huffman untuk setiap karakter (termasuk spasi) dan hitung panjang total kode pesan dalam satuan bit!

Solusi:

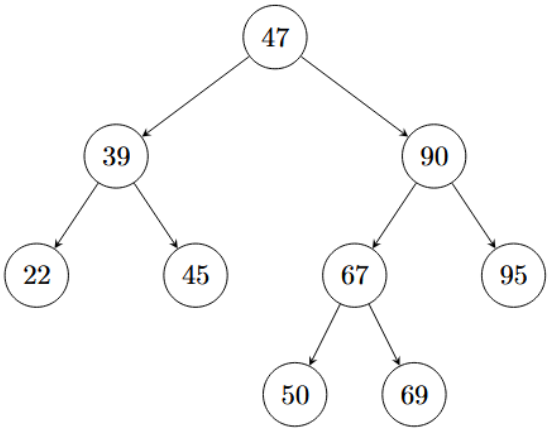


Simbol	Frekuensi	Kode Huffman	Panjang Kode	Total Bit
N	4	00	2	8
I	3	100	3	9
M	3	101	3	9
(spasi)	2	1101	4	8
E	2	1111	4	8
P	1	0100	4	4
R	1	0101	4	4
S	1	0110	4	4
T	1	0111	4	4
U	1	1100	4	4
A	1	11100	5	5
G	1	11101	5	5
			Total Panjang Kode	**72 bit**

6. **(Nilai 10)** Diberikan data: 47, 90, 67, 50, 95, 69, 39, 45, 22.
   (a) Buatlah *Binary Search Tree* (BST) dari data tersebut dengan 47 sebagai akar.
   (b) Dari BST yang dibuat pada bagian (a), berapa banyak perbandingan yang dilakukan untuk menemukan 67? Tuliskan langkah-langkahnya.
   (c) Dari BST yang dibuat pada bagian (a), berikan sekuens penelusuran dengan cara *postorder*.

Solusi:

(a) Pohon dari sekuens angka tersebut adalah sebagai berikut.



(b) Langkah-langkah untuk menemukan 67 adalah sebagai berikut.

i. Bandingkan akar 47 dengan 67, 67 > 47 sehingga ke child kanan

ii. Bandingkan 90 dengan 67, 90 > 67 sehingga ke child kiri

iii. Bandingkan 67 dengan 67, 67 = 67 sehingga 67 ditemukan.

Sehingga diperlukan **3 perbandingan** untuk menemukan 67.

(c) Penelusuran sekuens secara postorder: 22, 45, 39, 50, 69, 67, 95, 90, 47.

7. **(Nilai 15)** Diberikan dua buah kode program Python di bawah ini:

```
def algo_P(n):
    total = 0
    for i in range(1, n + 1):
        for j in range(1, n + 1):
            total += 1
    return total
```

```
def algo_Q(n):
    total = 0
    i = 1
    while i <= n:
        for j in range(1, n + 1):
            total += 1
        i *= 2
    return total
```

(a) Tentukan T(n) secara eksak untuk kedua algoritma di atas!

(b) Hitung T(8) masing-masing!

(c) Mulai dari *n* berapa algoritma_Q lebih sangkil (efisien) dari algoritma_P?

(d) Tentukan Big-O, Big-Ω, Big-Θ masing-masing algoritma.

Solusi:

(a) Algoritma_P: *outer loop* n kali, *inner loop* n kali $\rightarrow T_P(n) = n^2$

Algoritma_Q: *outer loop* $[log_2 n] + 1$, *inner loop* n kali $\rightarrow T_Q(n) = n * ([log_2 n] + 1)$

(b) Untuk n = 8, didapatkan:

$T_P(8) = 8^2 = 64$

$T_Q(n) = n * ([log_2 n] + 1) = 8 * (3 + 1) = 32$

(c) $T_Q(n) < T_P(n) \Leftrightarrow n(log_2 n + 1) < n^2 \Leftrightarrow log_2 n + 1 < n$

Lalu dicek:

n = 2, didapatkan $1 + 1 < 2$
n = 3, didapatkan $1 + 1 < 3$
Maka, mulai dari n = 3, algo_Q lebih efisien dari algo_P

(d) Berikut notasinya:

	algo_P	algo_Q
Big-O	$O(n^2)$	$O(n\ log\ n)$
Big-Ω	$\Omega(n^2)$	$\Omega(n\ log\ n)$
Big-Θ	$\Theta(n^2)$	$\Theta(n\ \ log\ n)$

8. **(Nilai 10)** Diberikan empat blok kode program Python berikut yang menerima input bilangan bulat positif n > 1. Dalam analisis ini, hitunglah kompleksitas berdasarkan berapa kali blok di dalam perulangan terdalam dieksekusi.
   (a) Tentukan rumus T(n) untuk menentukan jumlah operasi secara eksak untuk setiap algoritma!
   (b) Tentukan notasi asimtotik Big-O, lalu urutkan dari paling efisien hingga tidak efisien!

```
def fungsi_A(n):
    res = 0
    i = 0
    while i < 2 * n:
        res += (i % 2)
        i += 2
    return res
```

```
def fungsi_B(n):
    res = 0
    for i in range(n, 0, -1):
        j = 1
        while j <= n:
            res += (i * j)
            j += 1
    return res
```

```
def fungsi_C(n):
    res = 0
    k = n
    while k > 0:
        res += (k % 2)
        k = k // 2
    return res
```

```
def fungsi_D(n):
    res = 0
    for i in range(1, n + 1):
        p = 1
        while p < n:
            res += i
            p = p * 3
    return res
```

Solusi:

(a) T(n)

i. fungsi_A:
Perulangan while diinisialisasi dari i = 0 dan syarat berhentinya adalah i < 2n. Namun, pada setiap iterasi, nilai i ditambah dengan 2 (i += 2).
Deret nilai i adalah: 0, 2, 4, 6, ..., 2n-2.
Jumlah iterasi = 2n/2 = n.
T(n) = n

ii. fungsi_B:
Terdapat *nested loop*. Perulangan luar (for i) berjalan mundur dari n hingga 1, yang berarti dieksekusi sebanyak n kali. Perulangan dalam (while j) berjalan maju dari 1 hingga n, yang juga dieksekusi sebanyak n kali untuk setiap iterasi luar.
Jumlah iterasi = n * n.
$T(n) = n^2$

iii. fungsi_C:
Perulangan while membagi nilai k dengan 2 secara bertahap (k // 2). Deret nilai k akan terus dibelah dua (n, n/2, n/4, ...) hingga mencapai 0.
Jumlah iterasi bergantung pada berapa kali bilangan bisa dibagi 2, yang secara matematis adalah logaritma basis 2.
T(n) = floor(2log n)+ 1

iv. fungsi_D:
Perulangan luar (for i) berjalan linear sebanyak n kali.
Perulangan dalam (while p) mengalikan nilai p dengan 3 pada setiap iterasi ($p = 1, 3, 9, 27, ...) hingga p mencapai atau melampaui n. Karena dikali 3, ini adalah perulangan logaritmik dengan basis 3.
T(n) = n * ceil(3log n)

(b) Big-O
Fungsi_A : O(n)
Fungsi_B : $O(n^2)$
Fungsi_C : O(log n)
Fungsi_D : O(n log n)

9. **(Nilai 5)** Diberikan 5 algoritma dengan jumlah operasi T(n) sebagai berikut. Nyatakan Big-O masing-masing, lalu urutkan dari paling efisien ke paling tidak efisien!

$A: T_A(n) = (n+2)^2 - 4n$
$B: T_B(n) = n\ log(n^4) \quad + n^{1.5}$
$C: T_C(n) = \sum_{i=1}^{n} \left[\frac{n}{i}\right]$
$D: T_D(n) = 6(log\ n)^3 + n$
$E: T_E(n) = 4^{log_2 n} + n\ log\ n$

Solusi:
$A: (n+2)^2 - 4n = n^2 + 4n + 4 - 4n = n^2 + 4 -> O(n^2)$
$B: n\ log(n^4) + n^{1.5} = 4n\ log\ n + n^{1.5} -> O(n^{1.5})$
$C: \sum_{i=1}^{n} \left[\frac{n}{i}\right] \approx n * H(n)\ (Deret\ Harmonik) -> O(n\ log\ n)$
$D: (log\ n)^3 + n -> O(n)$
$E: 4^{log_2 n} + n\ log\ n = (2^2)^{log_2 n} + n\ log\ n = (2^{log_2 n})^2 + n\ log\ n = n^2 + n\ log\ n -> O(n^2)$

Dengan demikian, urutannya adalah: D → C → B → A = E (Dari paling efisien ke paling tidak efisien)