DENGAN INI SAYA MENYATAKAN
BAHWA PR INI ADALAH HASIL PEKERJAAN
DIRI SAYA SENDIRI

Luthfi
ALDEN LUTHFI
2206 028 932

1. a. Baris 7 dan 8 diganti:

```
7 | for i = k-1 down to 0
8 |     C[i] = C[i] + C[i+1]
```

b. A = [3,2,3,0,6,7,2,2,0,6,0,2,8,9,3,2] 1 indexed
B = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0] 1 indexed
C = [0,0,0,0,0,0,0,0,0,0] 0 indexed

(i) setelah for loop kedua:
C = [3,0,5,2,0,0,2,1,1,2]

(ii) setelah for loop ketiga (modifikasi a)
C = [16,13,13,8,6,6,6,4,3,2]

(iii) for loop terakhir

① B[13] = 2, C[2] = 12
② B[8] = 3, C[3] = 7
③ B[2] = 9, C[9] = 1
④ B[3] = 8, C[8] = 2
⑤ B[12] = 2, C[2] = 11
⑥ B[16] = 0, C[0] = 15
⑦ B[6] = 6, C[6] = 5
⑧ B[15] = 0, C[0] = 14
⑨ B[11] = 2, C[2] = 10
⑩ B[10] = 2, C[2] = 9
⑪ B[4] = 7, C[7] = 3
⑫ B[5] = 6, C[6] = 4
⑬ B[14] = 0, C[0] = 13
⑭ B[1] = 9, C[9] = 0
⑮ B[9] = 2, C[2] = 8
⑯ B[7] = 3, C[3] = 6

c.

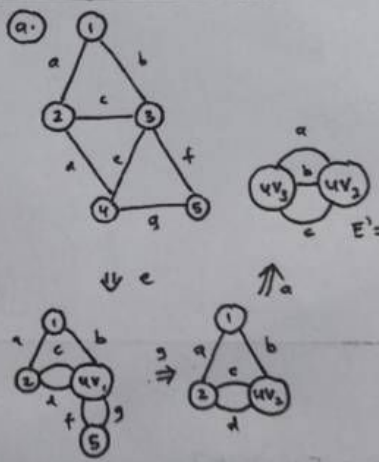
①	②	③	④	⑤
MAKAN	BUNGA	MAKAN	PAGAR	BOTOL
MINUM	BOTOL	PAGAR	MAKAN	BUNGA
BOTOL	MINUM	GELAS	HANTU	DAPUR
KARTU	MAKAN	BUNGA	DAPUR	GELAS
GELAS	DAPUR	BOTOL	KARTU	HANTU
PINTU	PAGAR	KARTU PINTU HANTU	GELAS	KARTU
DAPUR	GELAS		PINTU	MAKAN
HANTU	KARTU		MINUN	MINUM
PAGAR	PINTU	MINUN	BOTOL	PAGAR
BUNGA	HANTU	DAPUR	BUNGA	PINTU

2. a. A = [3,6,1,9,4,2,8,1,7]
① Quicksort (A,1,9)
↳ swap A[4], A[5]
↳ swap A[5], A[6]
↳ swap A[6], A[8]
↳ swap A[7], A[9]
A = [3,6,1,4,2,1,7,9,8]
② Quicksort (A,1,6)
↳ A[1], A[2] swap
↳ A[2], A[6] swap
A = [1,1,3,4,2,6,7,9,8]
③ Quicksort (A,3,6)
↳ swap A[6], A[6]
④ Quicksort (A,3,5)
↳ swap A[3], A[5]
A = [1,1,2,4,3,6,7,9,8]
⑤ Quicksort (A,4,5)
↳ swap A[4], A[5]
A = [1,1,2,3,4,6,7,9,8]
⑥ Quicksort (A,8,9)
↳ swap A[8], A[9]
A = [1,1,2,3,4,6,7,8,9] #

b. algoritma tersebut tidak stabil karena proses swap pada baris 6 dan 7 di PARTITION berpotensi mengacaukan elemen yang identik

c. dalam kasus worst case (terurut / terurut terbalik), Randomized quicksort berpotensi mempercepat running time karena pada quicksort biasa, pivot yang dipilih saat worst case merupakan pivot yang terburuk. randomized quicksort membuka kemungkinan pivot yang terpilih bukan terburuk.

3. a.



b. Algoritma tersebut adalah algoritma monte carlo karena tidak selalu benar, contoh graf menyisakan 3 edge padahal solusi optimalnya 2 edges.

c. ① untuk graf dengan ukuran solusi optimal min_cut() k, setiap verteks berderajat minimal k (Bukti: jika ada verteks yang derajatnya < k, k tidak optimal karena tinggal menghapus semua adj. edges verteks tsb yang < k)

② semua edge pada solusi min_cut() C optimal tidak pernah dipilih

Maka peluang mendapat solusi optimal = peluang edge pada C tidak pernah dipilih.

misal |C| = k

→ jumlah edge pada graf dengan setiap verteks berderajat minimal k adalah $\geq \frac{ik}{2}$ dengan i jumlah verteks

→ peluang edge pada C terpilih diantara semua verteks = $\frac{k}{\frac{ik}{2}} = \frac{2}{i}$

→ peluang edge pada C^2 tidak terpilih dari N verteks ke 2 verteks

$$\prod_{n=i}^{3} 1 - \frac{2}{i} = \frac{i-2}{i}$$

$$= \frac{n-2}{n} \cdot \frac{n-3}{n-1} \cdot \frac{n-4}{n-2} \cdots \frac{2}{4} \cdot \frac{1}{3} = \frac{2}{n(n-1)}$$

$$\geq \frac{1}{n^2}$$

$\mathcal{X}$: Jumlah eksekusi algoritma sebelum ditemukan solusi optimal

$\mathcal{X} \sim Geo\left(\frac{1}{n^2}\right)$

$P(\mathcal{X} \leq n) = 1 - \left(1 - \frac{1}{n^2}\right)^n$

(4.) (a.) Haskell / Python Pseudocode :

```
mt (n, i)
 | i > N   = 0
 | wi > n  = mt (n, i-1)
 | otherwise = max (mt (n-wi, i-2), mt (n, i-1))
                    +pi
```

w_i : weight di indeks i
p_i : point di indeks i
N : size(w) and size(p)
n : knapsack limit

(b.) Python :

```
def f(n) :
    dp = [[0]*(n+1) for _ in range(N+2)]

    for i in range(2, N+2):
        for j in range(n+1):
            if w[N+1-i] > j :
                dp[i][j] = dp[i-1][j]
            else :
                dp[i][j] = max(dp[i-1][j], dp[i-2][j-w[N+1-i]]+p[N+1-i])
    solution = []
    i = N+1
    while i > 1 :
        if dp[i][n] = dp[i-1][n]
            i -= 1
        else :
            solution.append(N+1-i)
            n -= w[N+1-i]
            i -= 2
    return solution
```

(c.) $T(n) \in O(Nn)$

(d.) final dp table

i \ n	0	1	2	3	4	5	6	7	8	9	10	
0	0	0	0	0	0	0	0	0	0	0	0	
1	0	0	0	0	0	0	0	0	0	0	0	
2	0	0	0	0	6	(6)	6	6	6	6	6	→ 7
3	0	0	0	0	6	(6)	6	7	7	7	7	
4	0	0	0	0	6	(6)	6	7	7	10	10	
5	0	0	0	0	6	6	8	8	8	10	14	
6	0	3	3	3	6	9	(9)	9	10	10	14	→ 3
7	0	3	3	3	6	9	(9)	9	10	10	14	
8	0	3	5	8	8	9	11	14	14	14	15	
9	0	3	5	10	13	15	15	16	17	19	(19)	→ 0

solusi = [0, 3, 7]
max points : 19

(5.) Python :

```
def f(s, w, memo = {}):
    if s in memo :
        return memo[s]

    if not s :
        return d.get(w,0)

    if s[-1]+w not in d:
        result = f(s[:-1], s[-1]+w)

    else :
        result = max(f(s[:-1], s[-1]+w), f(s[:-1],"")+d[s[-1]+w])

    if not w:
        memo[s] = result

    return result
```

note: d itu dictionary poin kata-kata