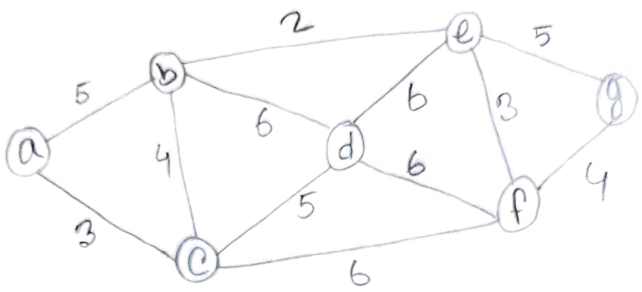
① I will use Kruskal Algorithm with DSU.



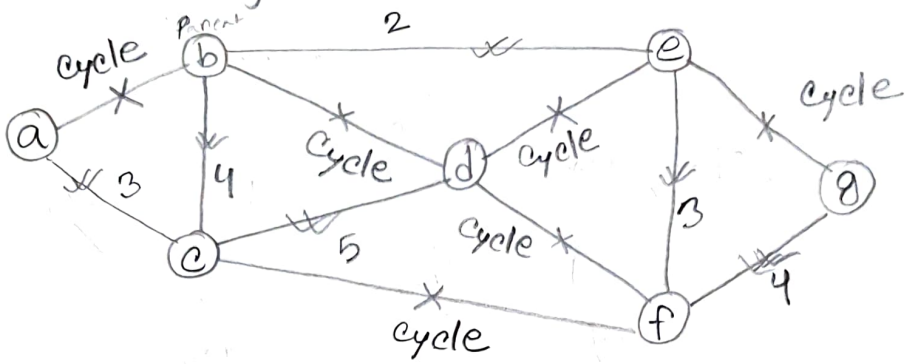
Step 1: Sort edges based on the length of road in ascending order.

length ∝ cost

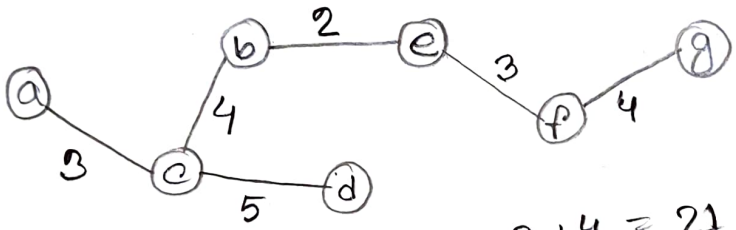
u	v	length/cost
b	e	2
e	f	3
a	c	3
b	c	4
f	g	4
a	b	5
c	d	5
e	g	5
b	d	6
d	e	6
d	f	6
c	f	6

Step 2: Call DSU

* Find parent → if parents will same it will create cycle. We will not repair that road.



So, roads need to be repared,



Total cost = 3+4+5+2+3+4 = 21

②

@ $\underset{a}{0000}$ $\underset{e}{0100}$ $\underset{e}{0100}$ $\underset{i}{1000}$ $\underset{e}{0100}$ $\underset{i}{1000}$ $\underset{e}{0100}$ $\underset{i}{1000}$ $\underset{b}{0001}$

$\underset{f}{0101}$ $\underset{b}{0001}$ $\underset{b}{0001}$ $\underset{g}{0110}$ $\underset{a}{0000}$ $\underset{e}{0100}$ $\underset{j}{1001}$ $\underset{e}{0010}$ $\underset{j}{1001}$

The message: aeeieiejbfbbgaejeij

(b)

a = 2

b = 3

c = 1

d = 0

e = 5

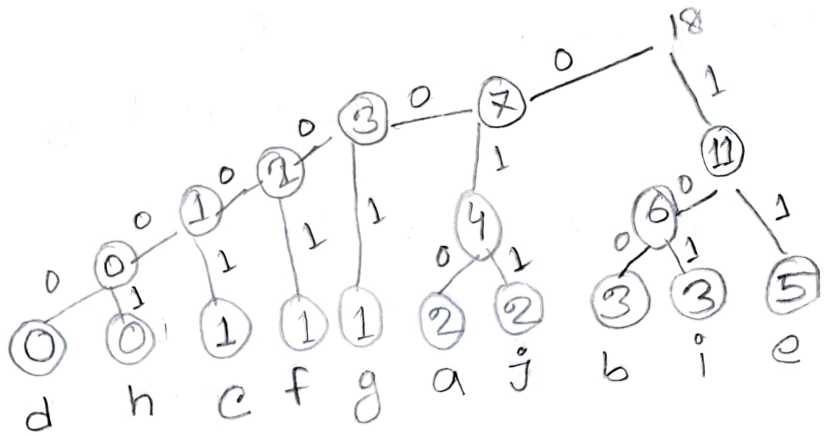
f = 1

g = 1

h = 0

i = 3

j = 2



(c)

a = 010

b = 100

c = 00001

e = 11

f = 0001

g = 001

i = 101

j = 011

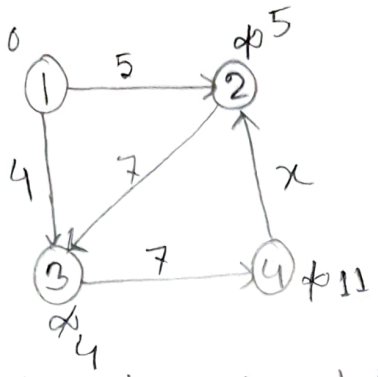
∴ Required bits =

(3×2) + (3×3) + (5×1) + (2×5) + (4×1)

+ (3×1) + (3×3) + (3×2)

= 52 bits

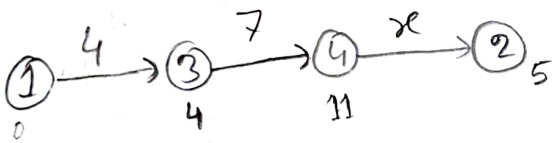
③



①

Current	1	2	3	4
Parent		1	1	3
distance	0	∞ 5	∞ 4	∞ 11

1	2	3	4
	∞	∞	∞
1	5	4	∞
1,3	5	4	11
1,3,2	5	4	11
1,3,2,4	5	4	11



$4+7+x \geq 5$ [sum must be ≥ 5 as it is already relaxed]

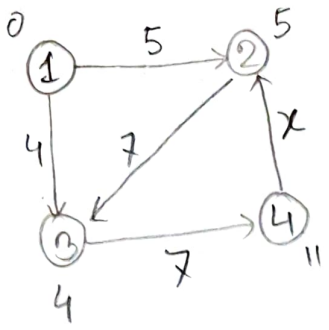
$\Rightarrow 11+x \geq 5$

$\therefore x \geq -6$

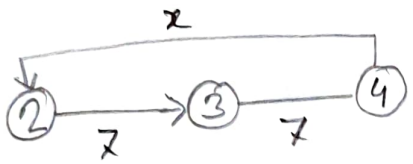
$\therefore -6 \leq x < 0$

So, if we take any value of x which is greater than or equal to -6 and less than 0, Dijkstra's Algorithm will work.

(11)



If the value of $x < -6$ [from ①] Dijkstra's Algorithm will not work.



As it forms a cycle, the value of x must be

$$7+7+x \geq 0$$

$$\Rightarrow x \geq -14$$

$$\therefore \quad -14 \leq x < -6$$

So, if we take any value of x which is greater than or equal to -14 and less than -6, Bellman Ford Algorithm will work.

(11) Bellman Ford's Algorithm doesn't work for negative sum cycle.

$$7+7+x<0$$

$$\therefore x<-14$$

If $x<-14$, then both Dijkstra and Bellman Ford Algorithm will not work. Normally, Dijkstra doesn't work for negative weighted edge graph. Besides, Bellman Ford Algorithm iterates over all edges for |vertex-1| times and finds the shortest path. But, if negative sum cycle exists in the graph, on the v^{th} iteration the distance may continueosly decrease. So, the correct shortest path cannot be found.

④

ⓐ

index		0	1	2	3	4	5	6	7	8	9	10
		–	P	O	L	Y	N	O	M	I	A	L
0	–	0	0	0	0	0	0	0	0	0	0	0
1	E	0	0	0	0	0	0	0	0	0	0	0
2	X	0	0	0	0	0	0	0	0	0	0	0
3	P	0	1	1	1	1	1	1	1	1	1	1
4	O	0	1	2	2	2	2	2	2	2	2	2
5	N	0	1	2	2	2	3	3	3	3	3	3
6	E	0	1	2	2	2	3	3	3	3	3	3
7	N	0	1	2	2	2	3	3	3	3	3	3
8	T	0	1	2	2	2	3	3	3	3	3	3
9	I	0	1	2	2	2	3	3	3	4	4	4
10	A	0	1	2	2	2	3	3	3	3	5	5
11	L	0	1	2	2	2	3	3	3	3	5	6

∴ The length of the LCS is 6.

(b)

The value of M[3][4] = 1

It means the length of LCS of strings "POLY" and "EXP" is 1.

(c)

If I backtrack from the LCS table which is created in "a", I will find the LCS string. The steps have been shown on the table.

I get "PONIAL", this is the LCS string.

(d)

```
S1 = POLYNOMIAL
S2 = EXPONENTIAL
l1, l2 = len(S1), len(S2)
table = [[0] * (l2+1) for i in range (l1+1)]
for i in range (l1):
    for j in range(l2):
        if S1[i] == S2[j]:
            table [i+1][j+1] = table [i][j]+1

        else:
            table [i+1][j+1] = max(table [i][j+1], table [i+1][i])

lcs = ""
i, j = l1, l2
```

```
while i>0 and j>0:
    if s1[i-1] == s2[j-1]:
        lcs = s1[i-1] + lcs
        i,j = i-1, j-1
    elif table[i-1][j] > table[i][j-1]:
        i-=1
    else: j-=1
print(table[11][12])
print(lcs)
```

⑤ Diamond (D), Jewelry (J), Sculpture (S), Painting (P), Gold Crest (G)

ⓐ

Item →	D	J	S	P	G
Profit →	3	4	12	9	12
Weight →	1	2	8	4	5

	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
D	0	3	3	3	3	3	3	3
J	0	3	4	7	7	7	7	7
S	0	3	4	7	7	7	7	7
P	0	3	4	7	9	12	13	16
G	0	3	4	7	9	12	15	16

∴ Maximum profit Denver can make 16$

ⓑ The array which stores the items Denver took.

D	J	S	P	G
1	1	0	1	0

7-4=3 ; 3-2=1 ; 1-1=0

∴ Denver will take Painting, Jewelry, Diamond.

(c)

Item →	D	J	S	P	G
Profit →	3	4	12	9	12
Weight →	1	2	8	4	5
Profit/weight →	3	2	1.5	2.25	2.4

item	Profit	weight	Remaining
D	3	1	7-1 = 6
G	12	5	6-5 = 1
P	$9 \times \frac{1}{4} = 2.25$	1	1-1 = 0

∴ Total profit = 17.25

Nairobi's profit (17.25) > Denver's profit (16).

So, Nairobi's belief remain valid after the robbery.

(d) DP equation for Denver's approach

if i = 0 or W = 0

DP[i][W] = 0

if $w_i > W$

DP[i][W] = D[i-1][W]

else dp[i][W] = max (DP[i-1][W], DP[i-1][W - W[i]] + profit[i])

(e)

```
weight = [1, 2, 8, 4, 5]
profit = [3, 4, 12, 9, 12]

c = 7

def knapsack(c, weight, profit):
    n = len(profit)
    dp = [[0]*(c+1) for i in range(n+1)]

    for i in range(n+1):
        for w in range(c+1):
            if i==0 or w==0:
                dp[i][w] = 0

            elif weight[i-1] > w:
                dp[i][w] = dp[i-1][w]

            else:
                dp[i][w] = max(dp[i-1][w], dp[i-1][w-weight[i-1]]
                               + profit[i-1])


    takenItems = []
    for i in range(n,0,-1):
        if dp[i][c] != dp[i-1][c]:
            takenItems.append(items[i-1])
            c = c - weight[i-1]

    print(dp[n][W])
    print(takenItems)

knapsack(c, weight, profit)
```