Theory Assignment

Submitted by : TASNIM RAHMAN MOUMITA

ID : 22301689

Section : 14

Course Code : CSE221

Course Title : Algorithms

Date of submission : 23.09.2024

Answer to the Q. NO - 01

Optimum meeting place :

To find the meeting point (x) that minimizes the total travel time.

dist (1, X) + dist (50, x)

Algorithm :

(I) To run Dijsktra's algorithm from my house to get the shortest distances to all vertices.

(II) Run Dijkstra's algorithm from my friend Benji's house (vertex 50) to get the shortest distances to all other vertices.

(III) Here, for every vertex (X) in the graph, we need to calculate dist (1, x) + dist (50, X)

(IV) Now, we need to choose the vertex (x) that minimizes the total travel time.

pseudocode :

```
def    find_opt_place (G):
       dist_1 = dijkstra (G, 1)
       dist_50 = dijkstra (G, 50)

       min_dist = float ('inf')
       meet_p = -1
       for x in range G :
           total_dist = dist_1 [x] + dist_50 [x]
           if total_dist < min_dist :
               min_dist = total_dist
               meet_p = x

       return meet_p
```

→ **Note :** Time Complexity : $O(V+E)$

Answer to the Q. NO - 02

Meet in the middle :

To find the optimal KFC outlet X from K areas that minimizes total travel time.

$dist(1, X) + dist(X, 50)$

Algorithm :

(i) To run Dijkstra's algorithm from my house (1) to get the shortest distance to all other vertices.

(ii) To run Dijkstra's algorithm from my friend's house (50) to get the shortest distance to all other vertices.

(iii) For every outlet (X), we need to calculate $dist(1, X) + dist(X, 50)$

(iv) We need to choose the outlet (X) that minimizes the sum of total travel time.

[P.T.O.]

pseudocode :

```
def find_KFC (G, KFC_loc) :
    dist_1 = dijkstra (G, 1)
    dist_50 = dijkstra (G, 50)

    min_dist = float ("Inf")
    KFC = -1

    for x in KFC_loc :
        total_dist = dist_1 [x] + dist_50[x]
        if total_dist < min_dist :
            min_dist = total_dist
            KFC = x

    return KFC, min_dist
```

* Time Complexity → $O(V+E)$

Answer to the Q. NO- 03 (a)

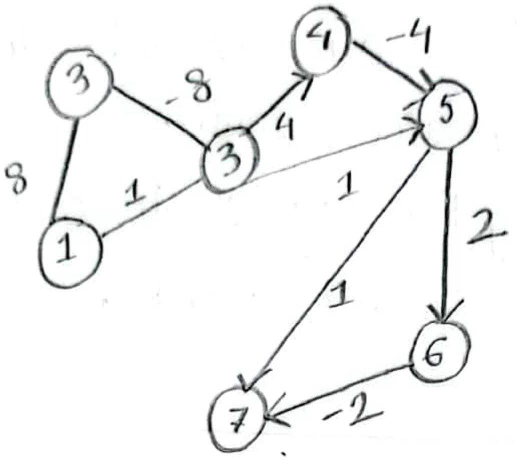


Figure : Directed Graph.

Simulation of Dijkstra's algorithm to find the shortest path costs from vertex 1 to all others:

Source	Destination					
1	2	3	4	5	6	7
0	∞	∞	∞	∞	∞	∞
	8	1	∞	∞	∞	∞
1, 3	8	1	5	2	∞	∞
1, 3, 5	8	1	5	2	4	3
1, 3, 5, 7	8	1	5	2	4	3
1, 3, 5, 7, 6	8	1	5	2	4	2 Since, 3 is updated already, it cannot b changed. And as 3>2. ∴ for Dijkstra's Algorithm, it is 2.

The graph we got from the given information Dijkstra's Algorithm is not appropriate for it. Since 7 is updated already into the queue. So 7 cannot be updated again.

So, it provides negative cycle here and the Dijkstra's Algorithm won't work.

Ans. to the Q. NO- 03 (b)

Dijkstra's algorithm might compute the correct shortest path costs for some vertices in some certain cases. This algorithm basically works under the assumption that all edge weights are non- negative.

when there are negative edge weights, this assumption fails.

If a vertex is not affected by any negative weight edges in its shortest path, the algorithm could determine the shortest path to that vertex correctly.

into the priority queue, only if it was previously visited.

(iii) It does not assume that extracting a vertex from the queue means that its shortest path cost is finalized :

Continuation to explore alternative paths that might offer lower costs.

With implementation of these modifications, this algorithm allows for the possibility of finding shorter paths that include negative - weight edges, hence improving the accuracy of shortest path calculations in graphs with negative weights.

Therefore, in practice, algorithms like Bellman-Ford are often preferred for graphs with negative-edge weights, since they are specifically structured to handle these type of cases efficiently.

In this given graph,

→ If the shortest path to vertex 1 or 2 does not involve any negative - weight edges, the algorithm could find the correct shortest path for these vertices.

→ For vertices that lie on paths containing negative - weight edges, Dijkstra's algorithm may not give the correct result.

Answer to the Q. NO - 03 (c)

Modified Dijkstra's Algorithm with the given changes :-

(i) Initialize Priority Queue with Source Vertex :

Begin with inserting with only the source vertex into the priority queue.

(ii) Distance of a vertex is updated, it inserts the vertex into the priority - queue :

Each time the distance to a vertex is updated (due to edge relaxation), & insert that vertex

Answer to the Q. NO- 3(d)

Time-complexity of this modified algorithm:

M.A (Grp, V, E, Source):

for each vertex V∈ Grp] → O(V)
  distance [V]=α
  dis [Source] = 0

for i = 1 to |V| - 1
  for each edge (u,v) ∈ Grp
    Relax (u,v,ω) ⟶ O(1)

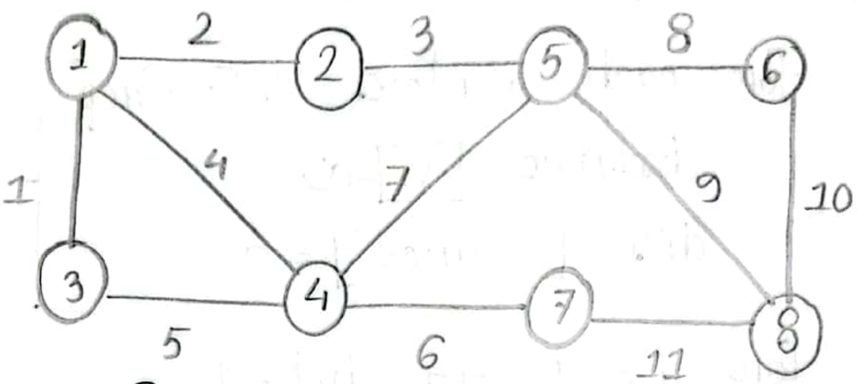
      for each edge (u,v) ∈ Grp
        if (dist [u] + ω (u,v) < dist [V])
          return "Graph contains negative weights"

return distance

In this code, it is relaxing every edge for vertex, V times. So, the time complexity will be = O(V×E)

Answer to the Q. NO- 04

Counter Graph →



$G_{left} = \{1, 2, 3, 4\}$

edge $= \{(1,2), (1,3), (3,5), (1,4)\}$

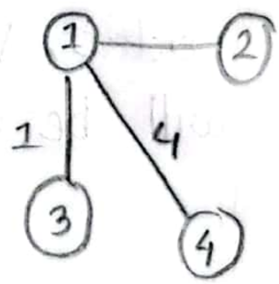
$G_{right} = \{5, 6, 7, 8\}$

edge $= \{(5,6), (5,8), (6,8), (7,8)\}$

$G_{remain} = \{(2,5), (4,5), (4,7)\}$

Applying Kruskal's algorithm with devide and conquere approach:

G_{left} = 1) (1,3), cost = 1
2) (1,2), cost = 2
3) (1,4), cost = 4
4) (3,4), cost = 5



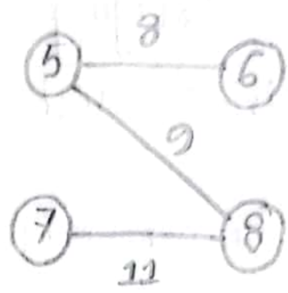
∴ MST = 1+2+4 = 7

G_{right} = 1) (5,6), cost = 8

2) (5,8), cost = 9

3) (6,8), cost = 10

4) (7,8), cost = 11



$\therefore$ MST = 8+9+11 = 28

G_{remain} = 1) (2,5), cost = 3

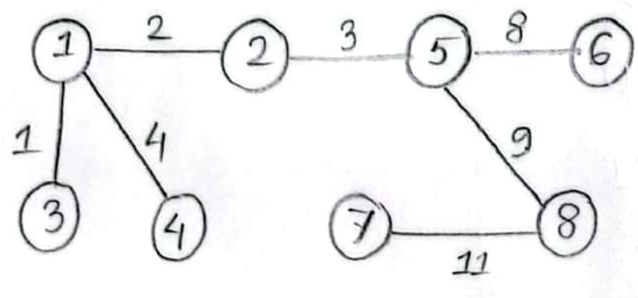
2) (4,5), cost = 7

3) (4,7), cost = 6

$\therefore$ MST = 3

Here,

Connecting the G_{left} and G_{right} with G_{remain} :



$\therefore$ MST = 1+2+3+4+8+9+11 = 38

[P.T.O.]

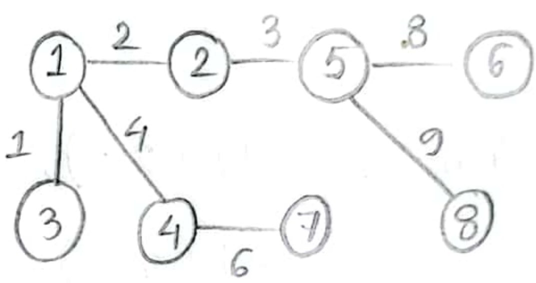
Again,

Applying normal Kruskal's algorithm without divide and conquer approach →

Sorted edges :-

1) (1, 3), cost = 1
2) (1, 2), cost = 2
3) (2, 5), cost = 3
4) (1, 4), cost = 4
5) (3, 4), cost = 5
6) (4, 7), cost = 6
7) (4, 5), cost = 7
8) (5, 6), cost = 8
9) (5, 8), cost = 9
10) (6, 8), cost = 10
11) (7, 8), cost = 11

Now,



$\therefore$ MST = 1+2+3+4+6+8+9
= 33

$\therefore$ MST of the standard Kruskal Algorithm is less cheaper than the MST of divide and conquer Kruskal Algorithm. And they will not always give the correct answer.

(proved)