Greedy Algorithms

Q.1 What is single source shortest path? Write an algorithm to find single source path using greedy method.

⇒ i) The graph is widely accepted data structure to represent distance map. The distance between cities effectively represented using graph.

ii) Dijkstra proposed an efficient way to find single source shortest path from the weighted graph.

iii) For a given source vertex s, the algorithm finds the shortest path to every vertex v in the graph.

iv) Assumption

⇒ Weight of all edges is non-negative.

v) Steps

⇒

a) Initialize the distance of source vertex to 0 and all other vertex to infinity.

b) Set source node to current node and put all the remaining nodes in the list of unvisited vertex list. Compute the tentative distance of all immediate neighbour vertex of current node.

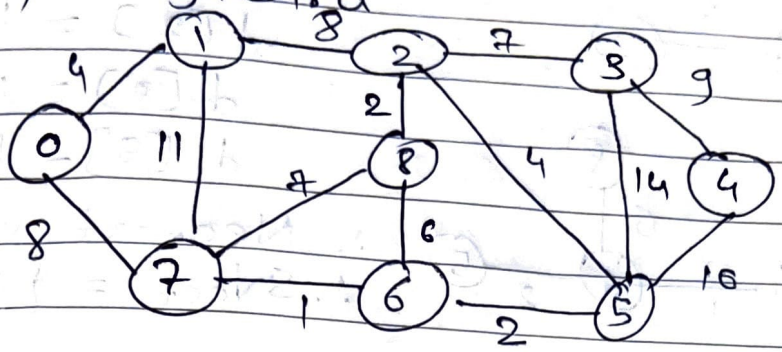
c) If newly computed value is smaller than old value, then update it.

vi) The ~~simple~~ time complexity of Dijkstra's algorithm is $O(|V^2|)$.

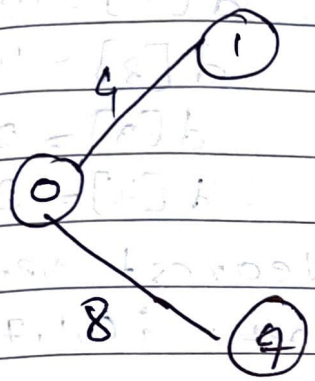
• Algorithm :

```
dist [s] ← 0
Π [s] ← NULL

for each vertex v ∈ V do

if v ≠ s then
    dist [v] ← ∞
    Π [v] ← undefined
end
ENQUEUE (v, Q)
end

while Q is not empty do

    u ← vertex in Q having minimum dist[u]
    if u == t then
    break
    end

    DEQUEUE (u, Q)
    for each adjacent node v of u do

        val ← dist [u] + weight (u,v)

        if val < dist [v] then
            dist [v] ← val
            Π [v] ← u
        end

    end
end
```

8.2 Apply Dijkstra



⇒ Soln :-



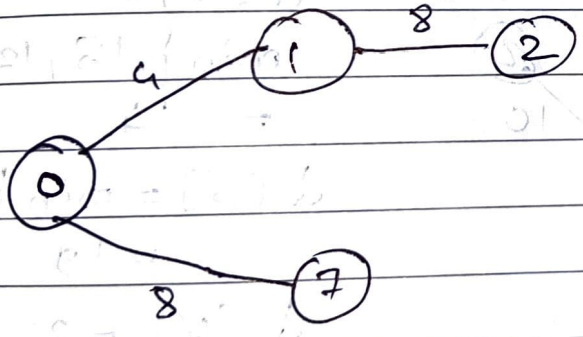
Here, 0 is the source vertex and {1, 7} are its adjacent vertex.

∴ d[1] = 0+4 = 4

∴ d[7] = 0+8 = 8

∴ The nearest vertex is 1.

Selected vertex set SVnet = {0, 1}



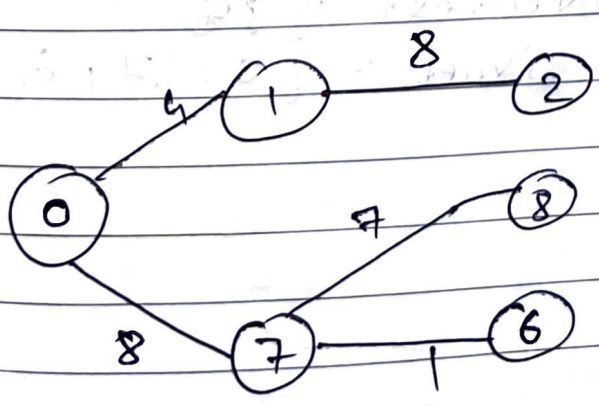
Now,

d[2] = 4+8 = 12

d[7] = 8

∴ Nearest vertex = 7

∴ SVnet = {0, 1, 7}



Now,

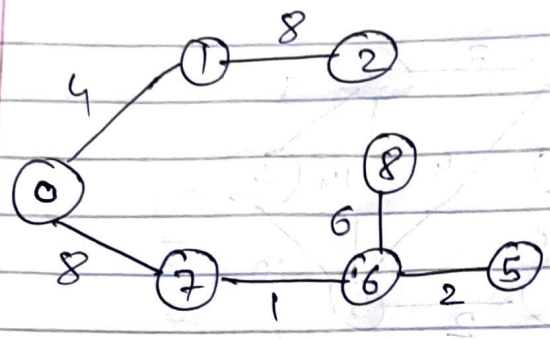
d[6] = 8+1 = 7

d[8] = 8+7 = 15

d[2] = 12

∴ Nearest vertex = 6

∴ SVnet = {0, 1, 7, 6}



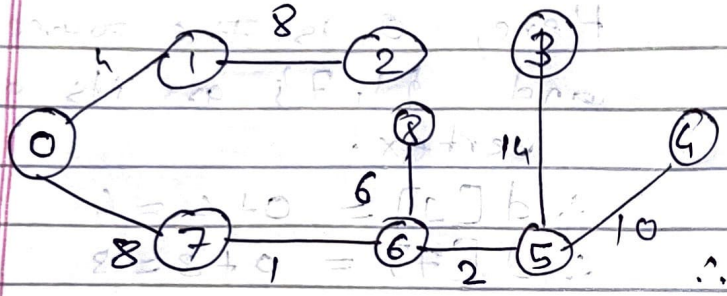
$d[2] = 12$

$d[8] = 15$

$d[5] = 11$

$\therefore$ Nearest Vertex = 5

$\therefore SV_{net} = \{0, 1, 7, 6, 5\}$



$d[2] = 12$

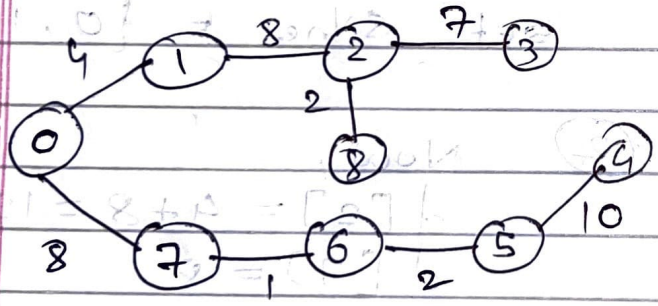
$d[8] = 15$

$d[3] = 25$

$d[4] = 21$

$\therefore$ Nearest vertex = 2

$\therefore SV_{net} = \{0, 1, 7, 6, 5, 2\}$



$d[8] =$

$\min\{15, 12+2\}$

$= 14$

$d[3] = \min\{25, 19\}$

$= 19$

$d[4] = 21$

Nearest = 8

$\therefore SV_{net} = \{0, 1, 7, 6, 5, 2, 8\}$

Q.3 Knapsack Problem

⇒ • Algorithm :

1) Sort the n objects from large to small based on the ratios p_i/w_i. We assume the arrays w[1...n] and p[1..n] stores weight and profit.
2) Initialize array x[1..n] to zeros.
3) wt = 0 , i=1 , profit = 0;
4) while (i ≤ n and wt < M) do
   if (wt + w[i] ≤ M) then
      x[i] = 1
   else
      x[i] = (M - wt) / w[i]
      wt = wt + x[i] * w[i]
      profit = profit + x[i] * p[i]
   i++

Q. n = 7 , m = 15, $(P_1, P_2, P_3, \ldots, P_7)$ = (10, 5, 15, 7, 6, 18, 3),
$(w_1, w_2, \ldots, w_7)$ = (2, 3, 5, 7, 1, 4, 1)

⇒ Soln :-

Item	Weight	Value	Ratio
I_1	2	10	5
I_2	3	5	1.67
I_3	5	15	3
I_4	7	7	1
I_5	1	6	6
I_6	4	18	4.5
I_7	1	3	3

- Arranging items in increasing order of ratio -

Item	Ratio
I_5	6
I_1	5
I_6	4.5
I_7	3
I_3	3
I_2	1.62
I_4	1

- M = 15

Capacity of Bag	Items	Value
15	--	0
14	I_5	6
12	I_5, I_1	16
8	I_5, I_1, I_6	34
7	I_5, I_1, I_6, I_7	37
2	I_5, I_1, I_6, I_2, I_3	52

Now, I_2 arrives but capacity of bag is only 2 and weight is 3 so we are going to take Fraction.

$$\therefore \text{Maximum Profit} = 52 + \frac{2}{3} \times 5$$

$$= 55.33$$

$$\text{Solution Vector} = \begin{bmatrix} I_1 & I_2 & I_3 & I_4 & I_5 & I_6 & I_7 \\ 1 & 2/3 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}$$

Q.4 Difference between greedy and DP.

=>

DP	GREEDY
i) DP gurantees an optimal solution.	i) Greedy does not gurantees an optimal solution.
ii) Sub-problems overlap.	ii) Sub-problems do not overlap.
iii) It does more work	iii) It does little work.
iv) Considers the future choices.	iv) Only considers the current choices.
v) Construct the solution from set of Feasi	v) Construct the solution from set of feasible solution.
v) There is no specilized set of feasible solution.	vi) Select choices which is locally optimum.
vi) Select choices which is globally optimum.	vii) There is no concept of memorization.
vii) Employ memorization.	E.g., Knapsack, Dijkstra Kruskal, Prims
E.g., 0/1 Knapsack, bellman ford, LCS	

Q.3 Job Sequencing.

⇒ i) Job sequencing is a problem in which n jobs are arranged in such way that we get maximum Profit.

ii) The job should complete their task within deadlines.

iii) It is also called as Job Sequencing with deadlines.

iv) Its time complexity is $O(n^2)$.

- Algorithm :

```
JOB_SCHEDULING (J, D, P) {

    D: Array of deadline
    P: Array of profit

    Sort all jobs J in decreasing order
      of profit.

    S ← ∅          // S is set of scheduled job
    SP ← 0         // Sum of profit.

    for i ← 1 to N do
        if J[i] is feasible then
            S ← S ∪ J[i]
            SP ← SP + P[i]
        end
    end
}
```

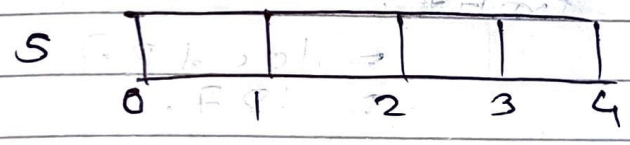
- Example:

Let $n = 4$, $(J_1, J_2, J_3, J_4) = (100, 10, 15, 27)$,
$(D_1, D_2, D_3, D_4) = (2, 1, 2, 1)$

Arranging the jobs in decreasing value -

Index	Jobs	Value	deadlines
1	J_1	100	2
2	J_4	27	1
3	J_3	15	2
4	J_2	10	1

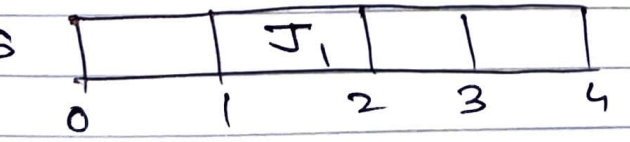
Initially,



$SP = 0$

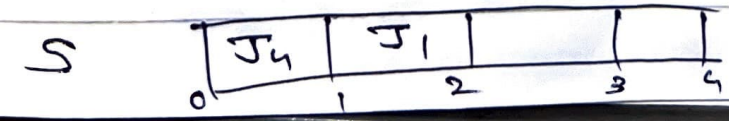
Iteration 1:

Deadline for J_1 is 2 and slot 2 ($t_1 = 1$ to $t = 2$) is Free, so schedule it in slot 2.



$\therefore SP = \{100\}$

Iteration 2: $J_4 \Rightarrow$ deadline is 1 and slot 1 is Free.



$\therefore SP = \{100, 27\}$

Iteration 3: $J_3 \Rightarrow$ deadline = 2

but slot 2 is not free.
So Job J_3 is discarded.

$S = \{J_1, J_4\}$,
$SP = \{100, 27\}$

Iteration 4: $J_2 \Rightarrow$ deadline = 1

but slot 1 is not free.
So Job J_2 is discarded.

$\therefore S = \{J_1, J_4\}$
$SP = \{100, 27\}$

Maximum Profit = $J_1 + J_4$
$= 100 + 27$
$= 127.$