ALGORITHM:

1. Get n of object
2. Get weight & profit for n objects
3. Calculate ratio = Profit / float(weight)
4. Get the total weight of bag, Capacity
5. Bubble sort ie) smallest on the right, use temp i, j = i+1 to sort profit, weight, ratio.

↓

for : i = 0 → n
    for : j = i+1 → n
        if (ratio[i] < ratio[j])

for : i = 0 → n
    if (weight[i] > capacity)
        total = total + ratio * Capacity   LAST CASE.
    else
        total = total + profit
        Capacity = Capacity - Weight.

DIJKSTA'S ALGORITHM

* We have to find the shortest path in a directed, weighted graph b/w 2 points.

* This is a minimisation problem + also a optimisation problem & solved by GREEDY approach.

* It works on directed & undirected graph.

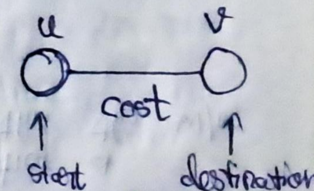
* If there is no direct Edge from source we initially keep the distance as ∞.

↑ INITIAL STEP. !

REPEATING STEP:

Relaxation ⇒ changing the distance w.r.t to the source in order to achieve the shortest path.

if (d(u) + c(u,v) < d(v))
    d(v) = d(u) + c(u,v)



* This is single source shortest problem.

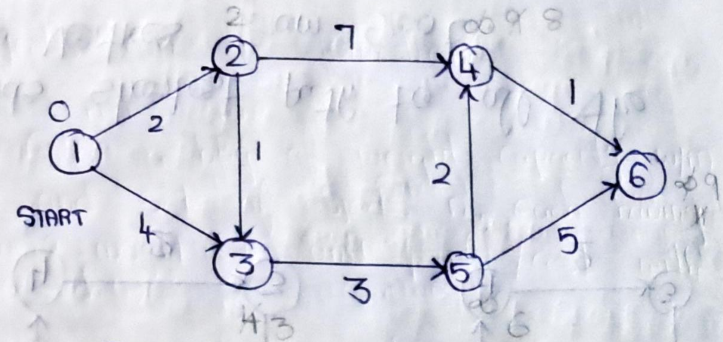
TC: $n = |V|$, no. of vertices.

$n \times n \leftarrow$ vertices relaxed.

↑
vertices

$\Theta(n^2)$ = In COMPLETE GRAPH — SHORTEST PATH TO ALL VERTICES

Eg:

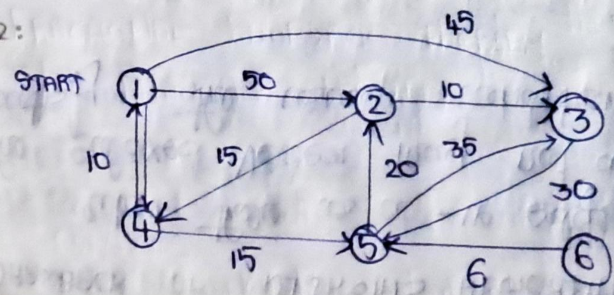


FOR START V[start] = 1

ANS:

V	2	3	4	5	6
d[V]	2	3	8	6	9

Eg 2:



we modify each vertex if we find the relaxation condition satisfying.

START VERTEX : 1

↗ Smallest Value each time.

Selected Vertex	2	3	4	5	6
Select @V = 4	50	45	10	∞	∞
@V = 5	50	45	10	25	∞
@V = 2	45	45	10	25	∞
@V = 3	45	45	10	25	∞
@V = 6	45	45	10	25	∞

} SELECTING THE SMALLEST WEIGHT

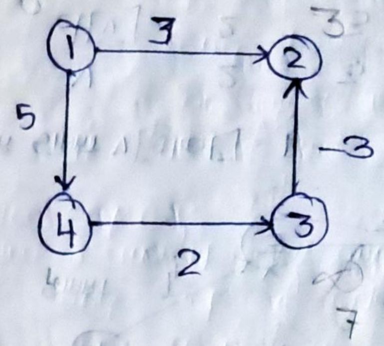
WE CAN'T REACH 6 from 1, coz we don't have a path, to reach 6.

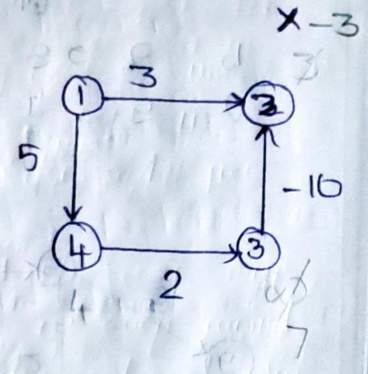
If uncomfortable with undirected
Convert undirected E → 2 directed anti-∥el edges.

No need to check the vertex if it is already relaxed.

DRAWBACK OF DIJKSTA'S ALGORITH

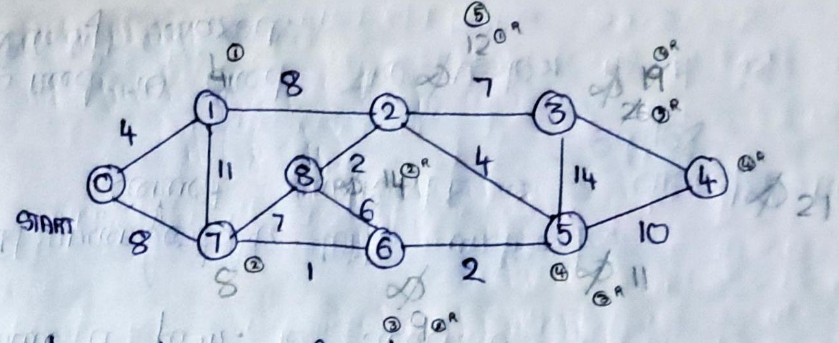
When there is a -ve edge/weight the relaxed Vertex may not always satisfy the usual condition





Finds shortest path to all the the Vertices from root.

GfG has it all.



Shortest path tree
Given/takken src or root

Take a G[ ][ ] adj matrix Weighted
Visited set
Shortest path set.

WORKING:

Dijkstras algorithm select the node with a shortest path and update the shortest path to other vertices

We do relaxation for the Vertices, whenever we choose a shortest path we try to relax the other vertices.

$$\begin{array}{cccccc} a & b & \overset{\downarrow i}{c} & a & b & c \\ 3 & 4 & 5 & 6 & 7 & 8 \end{array}$$

$$\begin{array}{cccccc} 0 & a^1 & \overset{\downarrow j}{b^2} & a^3 & b^4 & d^5 \\ & 0 & 0 & 1 & 2 & 0 \end{array}$$

Now after moving j
Compare i→j+1

Here also it not equals then it moves to b's π value of table

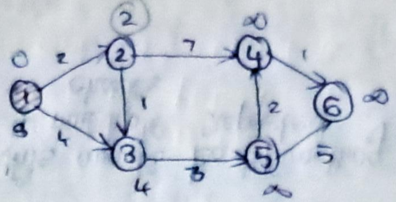
$$\begin{array}{ccccccc} \overset{\downarrow i}{c} & a & b & c & a & b & a \\ 5 & 6 & 7 & 8 & 9 & 10 & 11 \end{array}$$

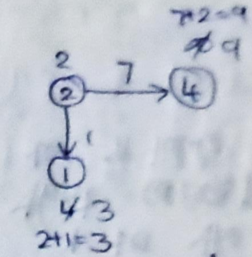
$$\begin{array}{cccccc} \overset{\downarrow j}{0} & a^1 & b^2 & c^3 & b^4 & d^5 \\ & 0 & 0 & 1 & 2 & 0 \end{array}$$

Here we compar i & j+1, still it doesn't match but we can't take it back further so, we increase i by 1

i only moved in forward direction
n times for passing through
m creating table
m+n = O(n).

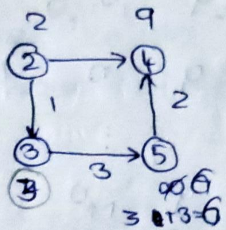
WORKING OF DIJKSTRA'S ALGORITHM



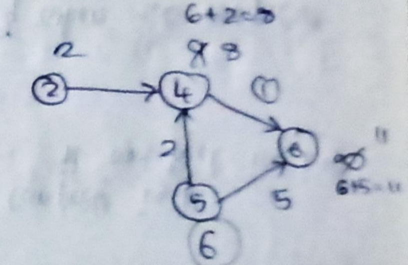


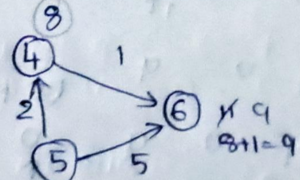
Relaxation of ① + ④

INITIAL STEP



We choose the shortest wt among all.





V	2	3	4	5	6
d[V]	2	3	8	6	9