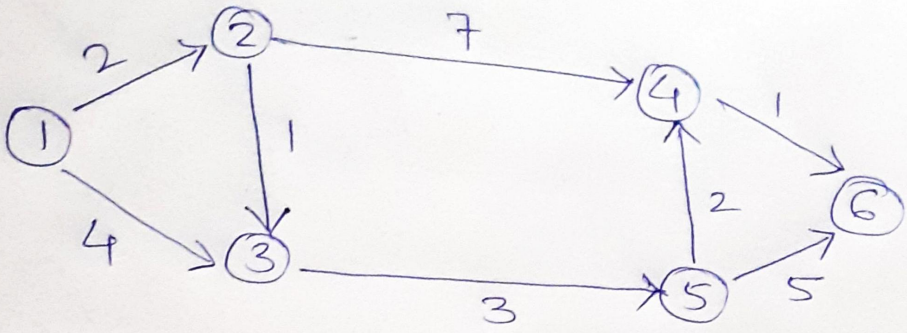
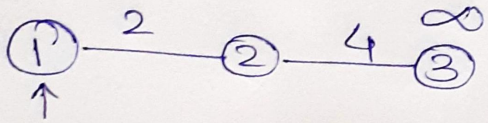
# Dijkstra Algorithm

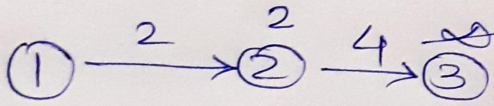


23/4/21

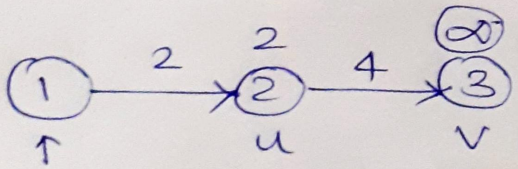



From vertex ① to ②, we have direct path and cost is '2' but vertex ① to ③, we don't have direct path so cost is assumed as '∞'. But we can have path from vertex ① to ③ via vertex ② and cost = 2+4 = 6




This updation in cost from ∞ to 6 is called Relaxation.




$2+4 \leq \infty$

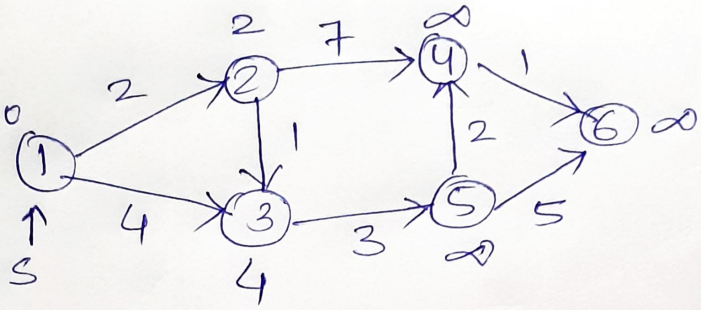
Relaxation

if $(d[u] + c(u,v) < d[v])$

$d[v] = d[u] + c(u,v)$

# Dijkstra Algorithm

23/4/21



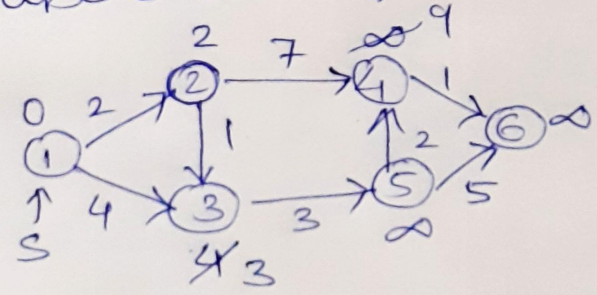

## Relaxation

if(d[u] + c(u,v) < d[v])

d[v] = d[u] + c(u,v)

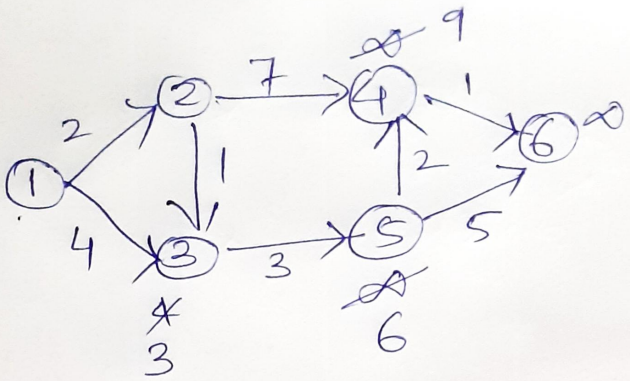
Out of ② and ③, cost of vertex ② is minimum, so select ②, now see adjacent path of ②, 2 to 4 and 2 to 3, from ② to ③ cost is 2+1 = 3 that is less than 4, so update or do Relaxation.

Now cost to reach vertex ④ is ∞, but via vertex ②, cost is 2+7 = 9, update it.



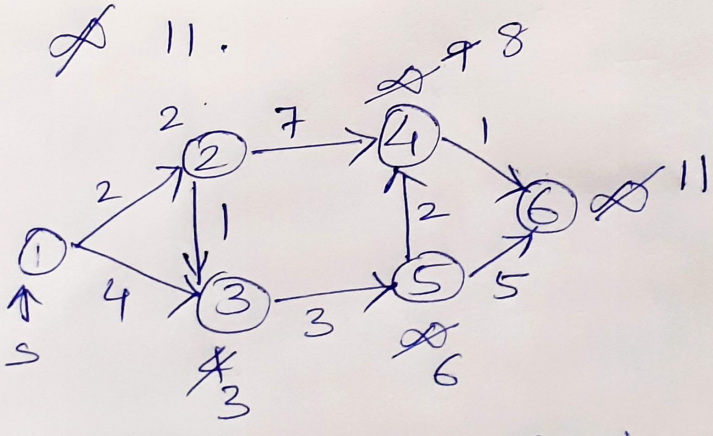
# Dijkstra Algorithm

23/4/21




③ to ⑤ cost 3+3 = 6

Now remaining vertex ④, ⑤ and ⑥, ⑤ is having min cost, from vertex ⑤ to ④ cost = 6+2 = 8 so update ∞ 8. and ⑤ to ⑥ cost ~~∞~~ 6+5 = 11 so update ∞ 11.

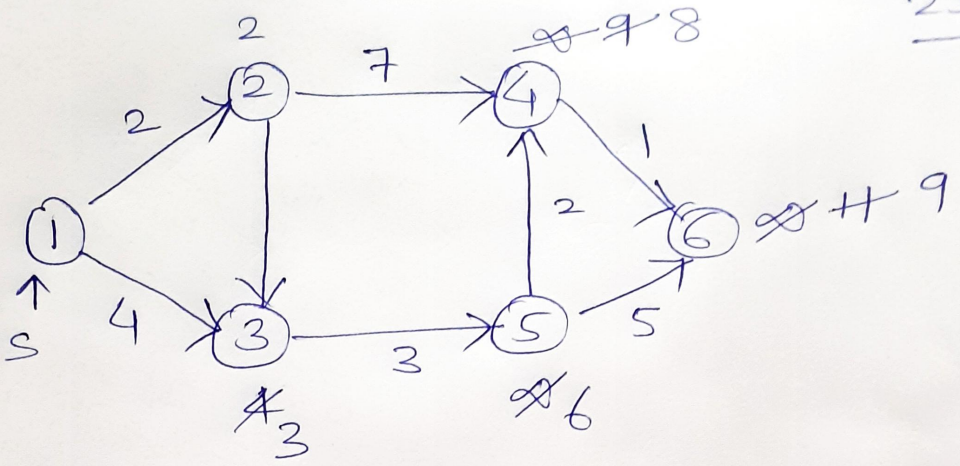



Out of ④ and ⑥, ④ is min and from ④ to ⑥ cost 8+1 = 9 update ∞ 9

# Dijkstra Algorithm



23/4/21




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

worst case time complexity $n = |V||V|$

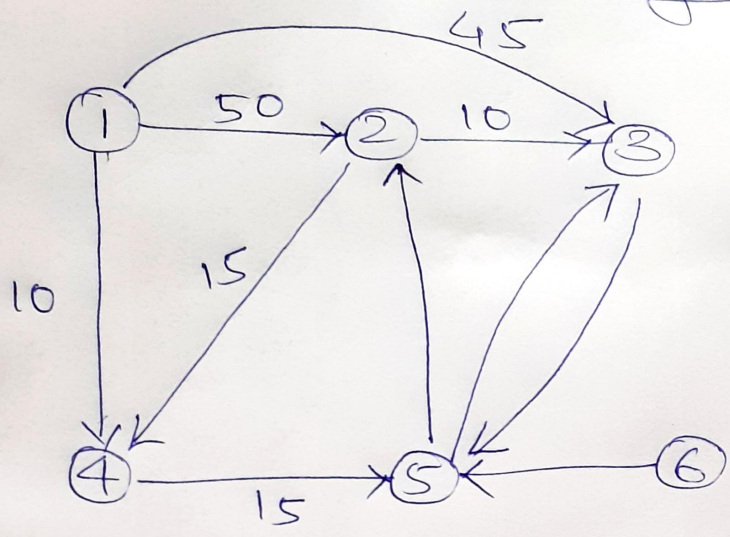
It is finding shortest path for all the vertices and it is relaxing most of the nodes than $n = |V||V|$

$$O(|V|^2)$$

# Dijkstra Algorithm



23/4/21

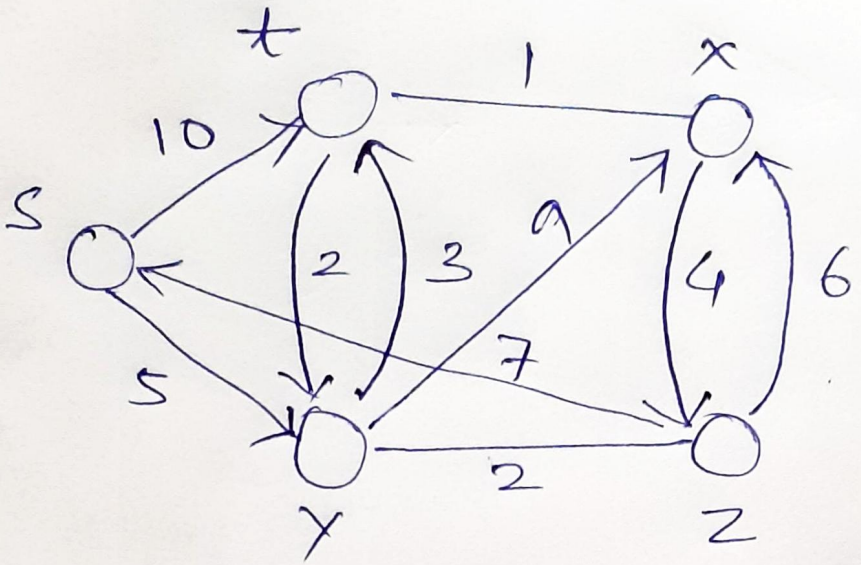



starting vertex 1.

| selected vertex | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|
| Select min cost vertex 4 | 50 | 45 | 10 | ∞ | ∞ |
| select nxt min cost vertex 5 | 50 | 45 | 10 | 25 | ∞ |
| 2 | 45 | 45 | 10 | 25 | ∞ |
| 3 | 45 | 45 | 10 | 25 | ∞ |
| 6 | 45 | 45 | 10 | 25 | ∞ |

# Dijkstra Algorithm

(6)
23/4/21




DIJKSTRA (G, W, S) — graph, weight, single source

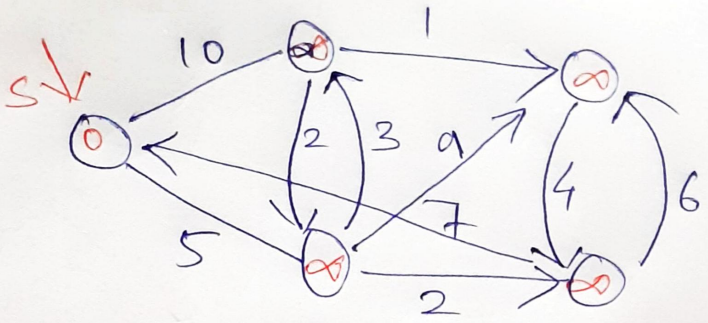
(1)
Algo

1. INITIALIZE-SINGLE-SOURCE(G,S)
2. $S \leftarrow \phi$
3. $Q \leftarrow V[G]$
4. while $Q \neq \phi$
   do $u \leftarrow$ Extractmin(Q)
      $S \leftarrow S \cup \{u\}$
      for each vertex $v \in Adj[u]$
         do RELAX(u,v,w)

# Dijkstra Algorithm

(7)

23/4/21




(2) Algo makes source '0' and other vertex ∞

S = [ ]

(1) Algo step 2 make S empty

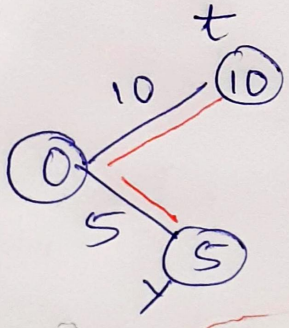
step 3

Q = [s, t, x, y, z]

step ⑤ u ← v[h]

(1) Algo step 4 while Q is not empty do u ← Extract-min(Q)

step 4 S ← S ∪ {u} start with source




S = [s, ···  Q = [t, x, y, z]

---

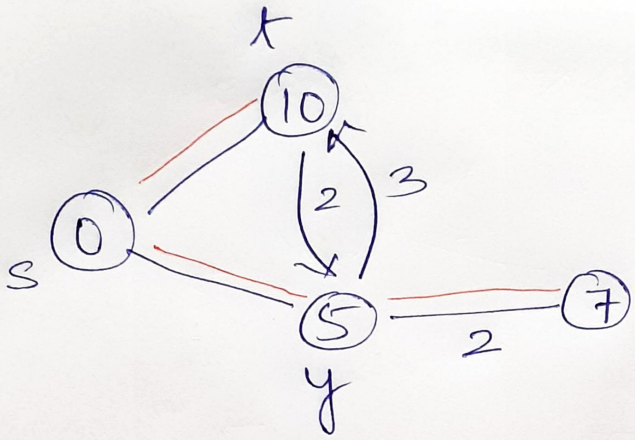
Now min is (5) at vertex 'y'

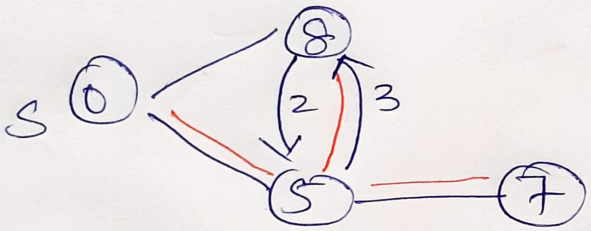
S = [s, y ···  Q = [t, x, z]

# Dijkstra Algorithm

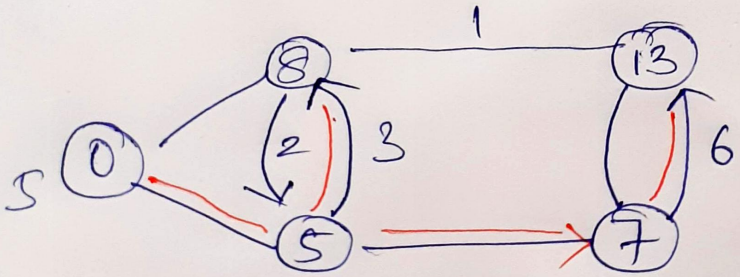


23/4/21

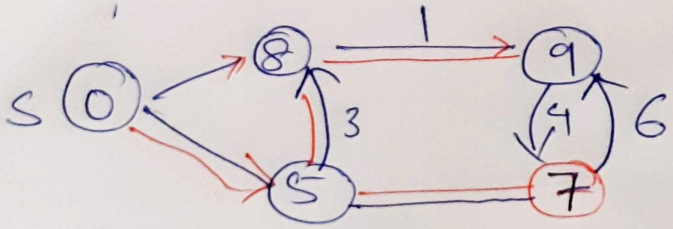



$S = [s, y, z$ $\quad Q = [t, x$




$S = [s, y, z, t$ $\quad Q = [x]$

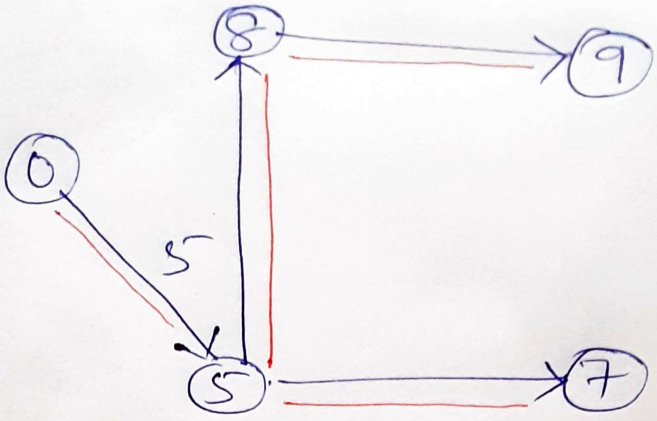



$S = [x, y, z, t, x]$ $\quad Q = [\,]$.

# Dijkstra Algorithm



23/4/21




Final Diagram