CSE221 ALGORITHMS

Topic: Dijkstra Algorithm, Bellman Ford Algorithm

Prepared by:
Saad Bin Sohan

BRAC University

Email: sohan.academics@gmail.com

GitHub: https://github.com/saad-bin-sohan

Recall: **Unweighted** graph এ BFS apply করে shortest distance / shortest path বের করা যায়।

But what happens when the graph is weighted? (concept of cost brings changes)

NB: weighted graph এ BFS apply করে shortest distance বের করার concept not applicable.

Weighted graph এ shortest distance between vertices বের করতে Dijkstra Algorithm ব্যবহার করা হয়।

Dijkstra Algorithm → shortest path in weighted graph

BFS → shortest path in unweighted graph

Dijkstra Algorithm

Psuedocode:

For all vertices,

D[v] = inf

Distance array. প্রত্যেকটা vertex এর জন্য একটা distance value এই array তে থাকে। এই distance value টা হলো starting vertex থেকে ঐ vertex এ যেতে minim-um total cost. initially এই value infinity set করা থাকে

P[v] = null

প্রতিটা vertex এর immediate parent কোন vertex সেই value টা parent array তে রাখবো।

d[s] = 0

absolute initial parent এর নিজের থেকে নিজের distance zero set করা হচ্ছে।

PQ = All vertices [Priority Queue]

সবগুলো vertex এখানে থাকবে। difference with usual queue is, এখানে First in first out apply হবে না। value pop হবে vertex গুলোকে queue এ push করার সময়ে প্রতিটা vertex কে একটা value assign করে দেওয়া হয়, ঐ value যে vertex এর সবচে কম, সে vertex আগে pop হবে।

while PQ is not empty:

U = PQ.extractmin()

minimum value র vertex pop বা extract হবে

For each U → V:

pop করা vertex U থেকে যত গুলো vertex (v) এ যাওয়া যায়

saad_bin_sohan

value কে priority দেওয়া
its called priority queue

```
if (v in PQ and
    d[u]+edge_cost<
     d[v]):
    D[v]=d[u]+edge_cost
    P[v] = u
```

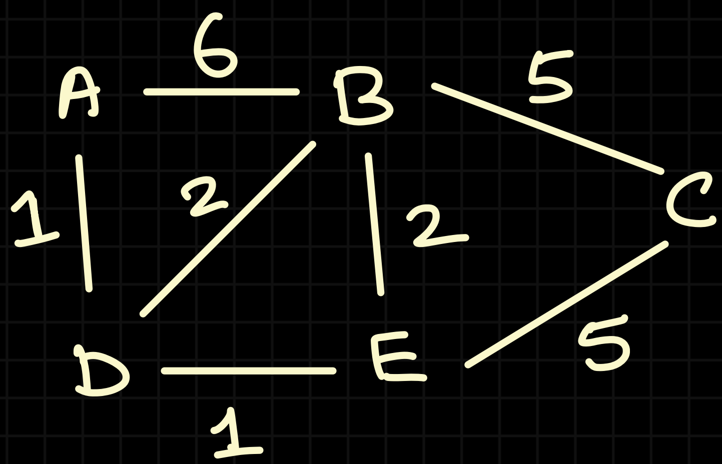
checks if destination v is in Priority Queue and absolute starting vertex থেকে u এর distance (+) u এর edge এর cost to the destination টা absolute starting vertex থেকে target vertex এর cost এর থেকে ছোট কিনা

ছোট হলে means there's a shorter path between absolute initial vertex and destination vertex v. so we update the distance accordingly and also update the parent of v to a newer vertex that gives shorter distance so in short:

Psuedocode:

```
For all vertices,
    D[v] = inf
    P[v] = null
d[s] = 0
PQ = All vertices [Priority Queue]
while PQ is not empty:
    U = PQ.extractmin()
    For each U→V:
    if (v in PQ and d[u]+edge_cost < d[v]):
        D[v] = d[u]+ edge_cost
        P[v] = u
```

simulation of Dijkstra Algorithm:

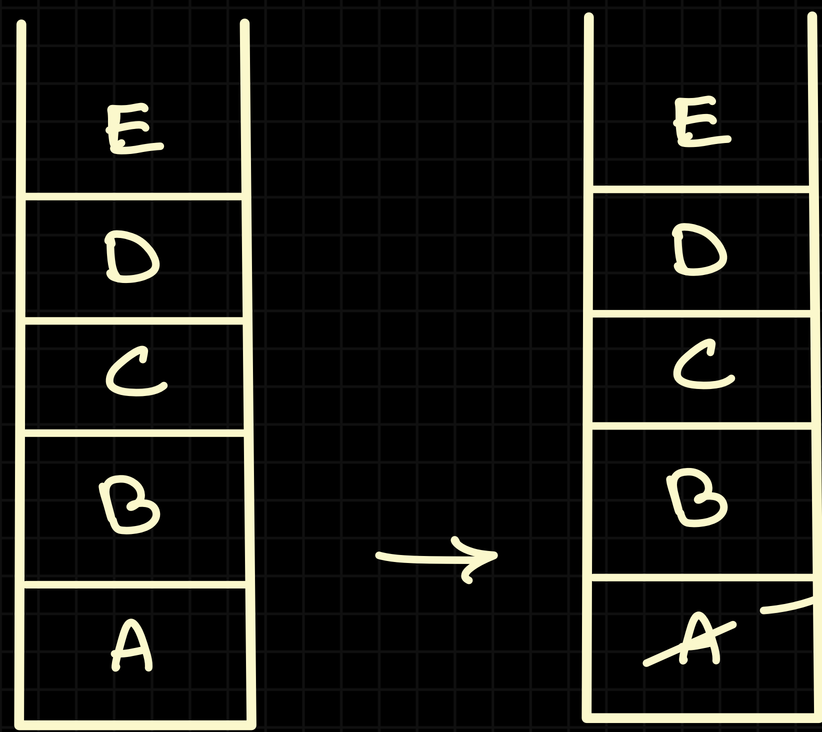


A is the absolute initial vertex.

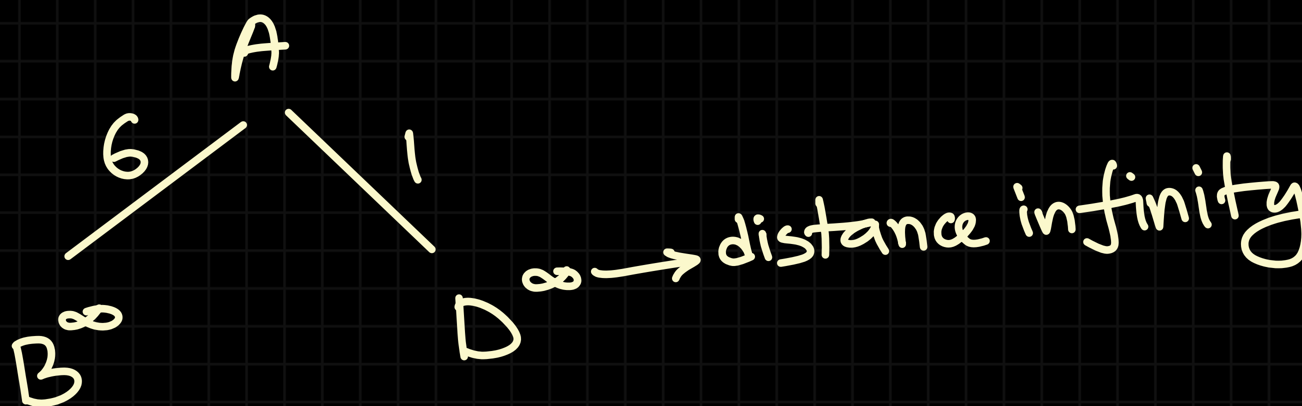
soln:

Vertex	Distance	Parent
A	~~inf~~ → 0	null
B	~~inf~~ → ~~6~~ → 3	~~null~~ → ~~A~~ → D
C	~~inf~~ → 7	~~null~~ → E
D	~~inf~~ → 1	~~null~~ → A
E	~~inf~~ → 2	~~null~~ → D

Priority Queue!



A pop করে সবার শুরুতে। not bc of FIFO but bc of A এর distance সবচে কম (zero). বাকি সবার infi-nity.



if condition এ,

i) v(B) is in PQ ✓✓

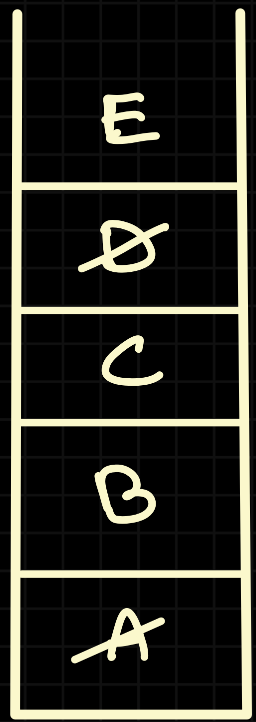
ii) $0+6 < \infty$ ✓✓

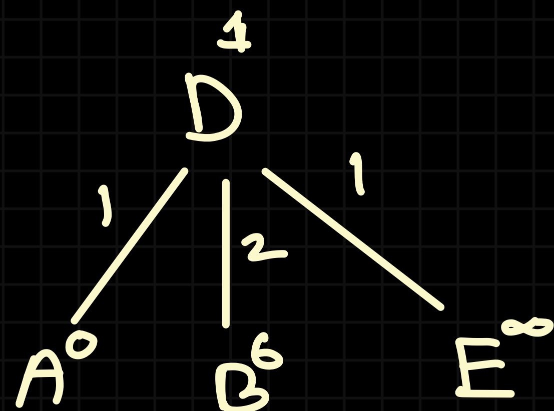
so we update B's distance and Parent

then check for all other vertices that have edge A. e.g. D.

এভাবে keep doing until the priority queue is empty.

next time D pick and pop out হবে bc A already popped out and D এর distance minimum (≈1)





এখন $D \rightarrow A$ এর জন্য condition check

i) v (A) not in PQ ✗

ii) no need to check

then for $D \rightarrow B$, condition check

i) B in the PQ ✓

ii) $1+2 < 6$

$\therefore$ $d[B] = 3$

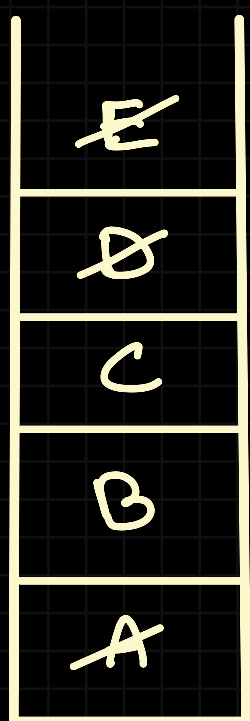
$P[B] = D$

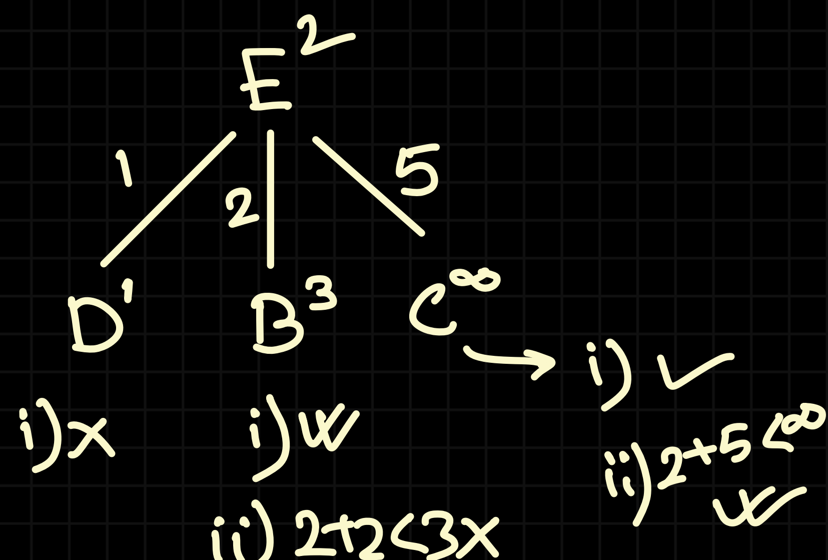
now for D → E :

i) ✓

ii) $1+1 < \infty$ ✓

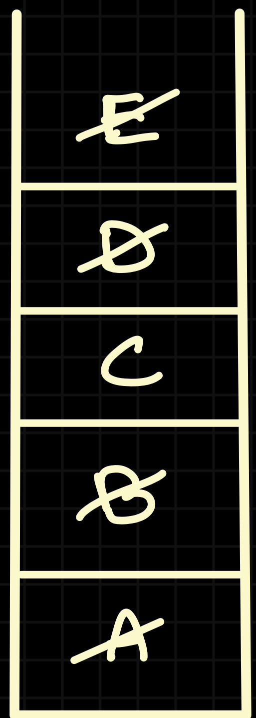
∴ among B, C, E, we choose E (least distance)



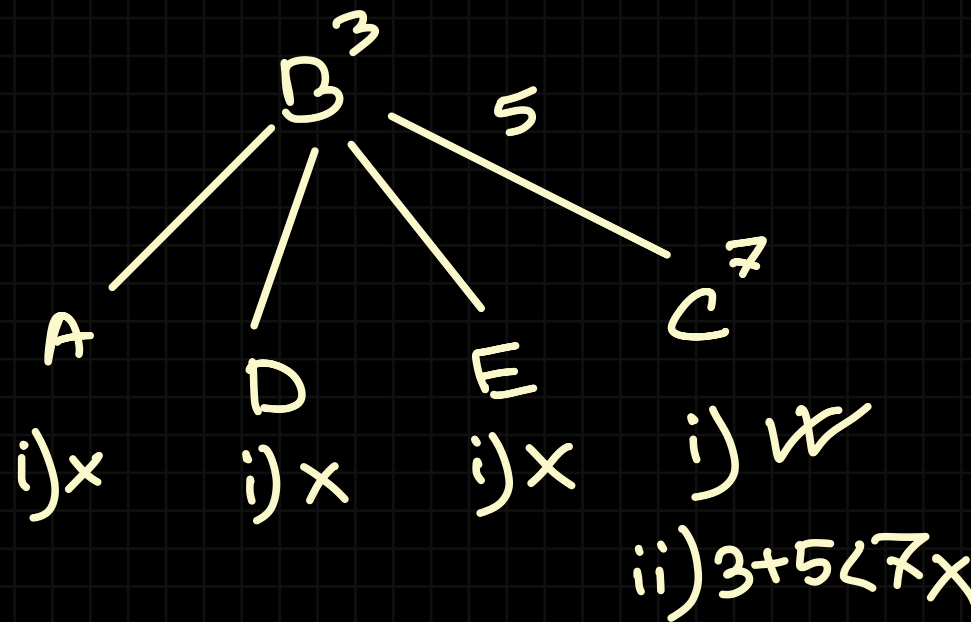


$d[C] = 7$

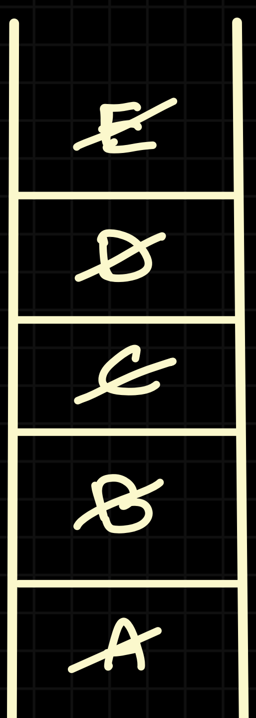
$P[C] = E$



B, C এর মধ্যে we choose B



then



since no C→vertex is in the priority queue, loop will break and C is popped out.

final answer,

Vertex	Distance	Parent
A	~~inf~~ → 0	null
B	~~inf~~ → ~~6~~ → 3	~~null~~ → ~~A~~ → D
C	~~inf~~ → 7	~~null~~ → E
D	~~inf~~ → 1	~~null~~ → A
E	~~inf~~ → 2	~~null~~ → D

now, que: previous graph এ A থেকে B এর minimum distance কোনটা?

answer:

step- I) look at the final table

ii) B এর parent কে shortest distance map এ খোঁজো

B ← D

iii) then D এর parent কে shortest distance map এ খোঁজো and carry on the same process until you reach the absolute initial source vertex.

B ← D ← A

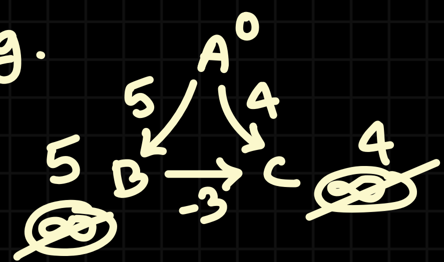
so shortest path:

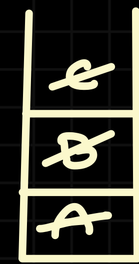
A → D → B

Drawback of Dijkstra Algorithm:

→ কোনো edge এর cost negative হলে Dijkstra Algorithm টা correct shortest path দিতে পারে না।

e.g.





C থেকে যাওয়ার মত কোনো vertex নাই। so no condition checking.

problem: Dijkstra apply করে C এর distance পাই 4। But actual shortest distance 2 (A → B → C)

এই issue resolve করে Bellman-Ford Algorithm