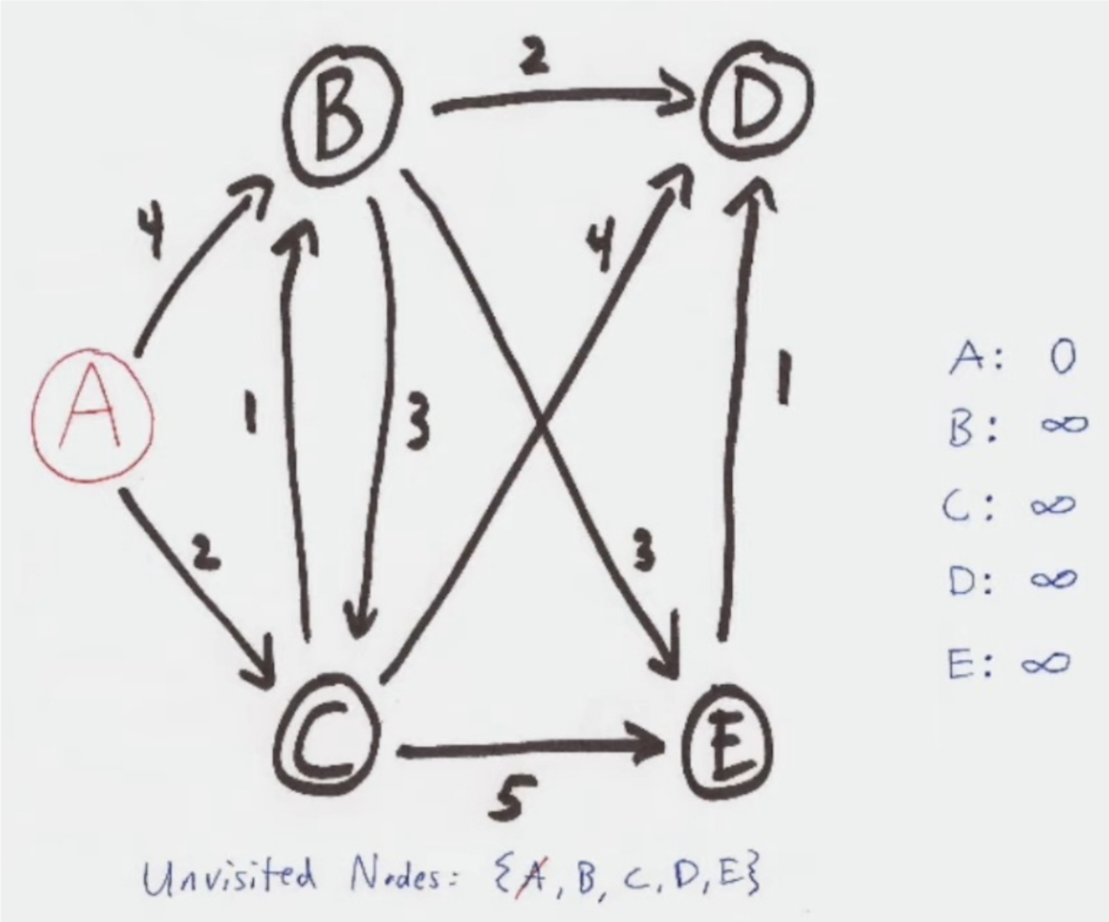
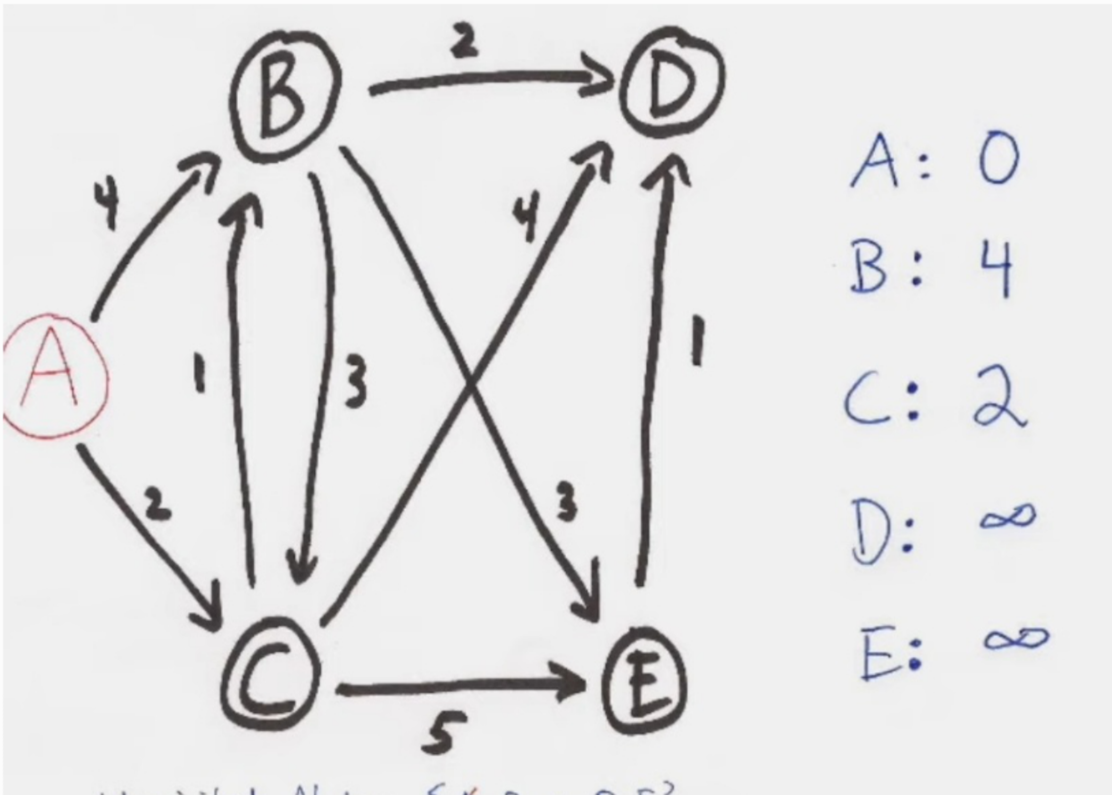
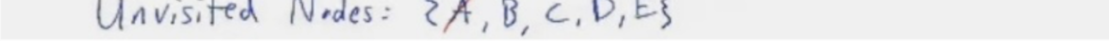
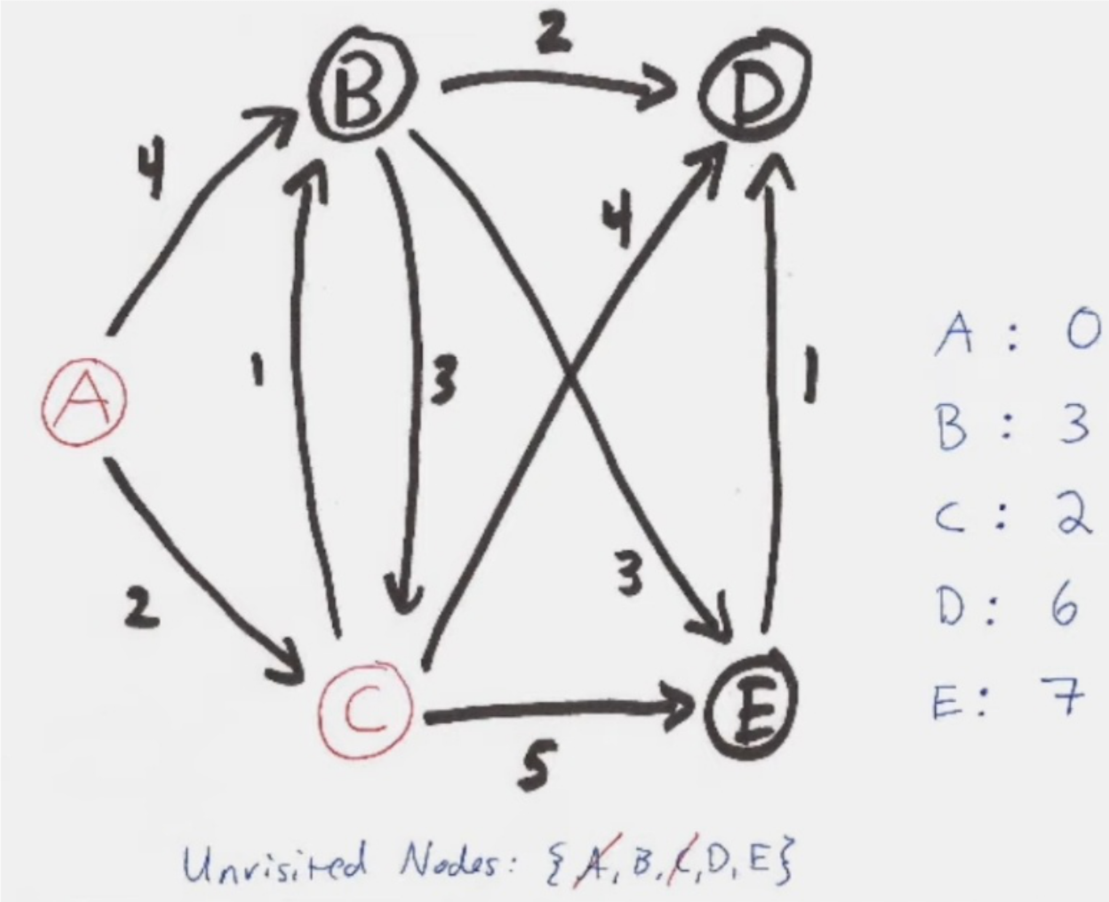
DIJKSTRA'S ALGORITHM

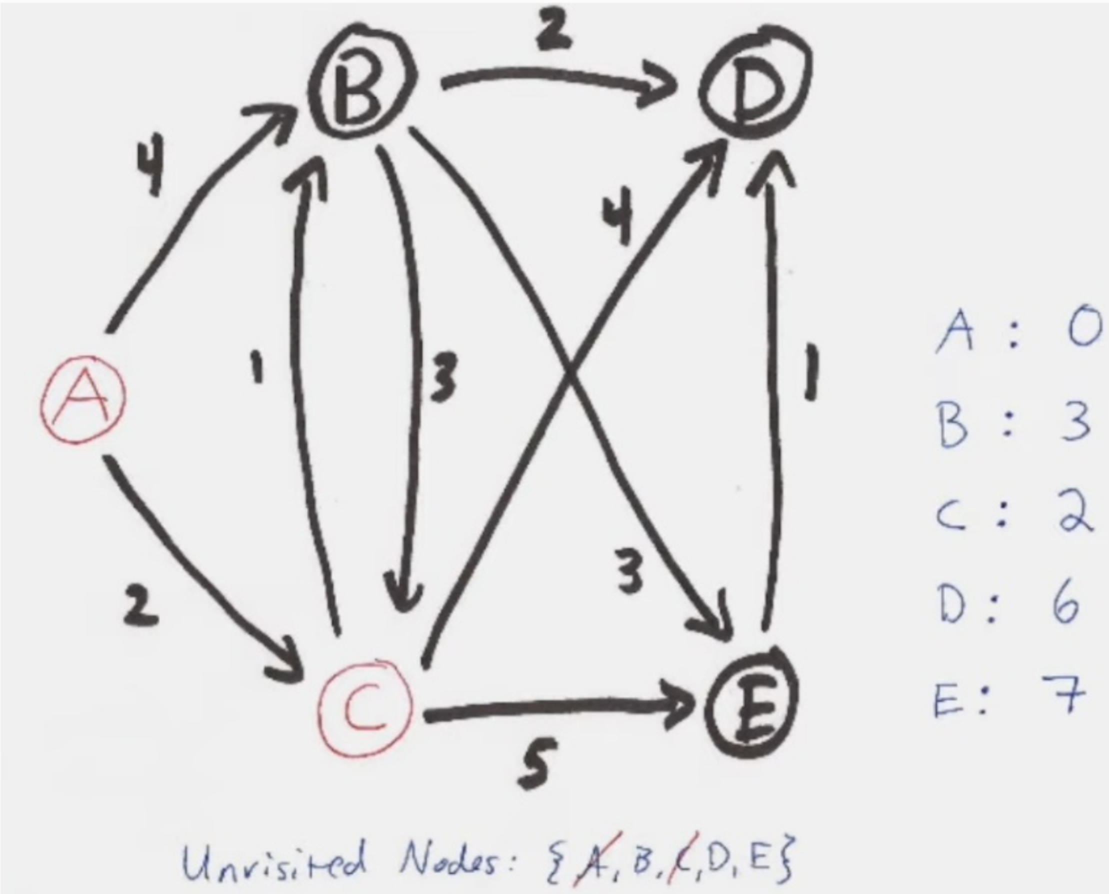
shortest path from <u>one</u> node to every other node

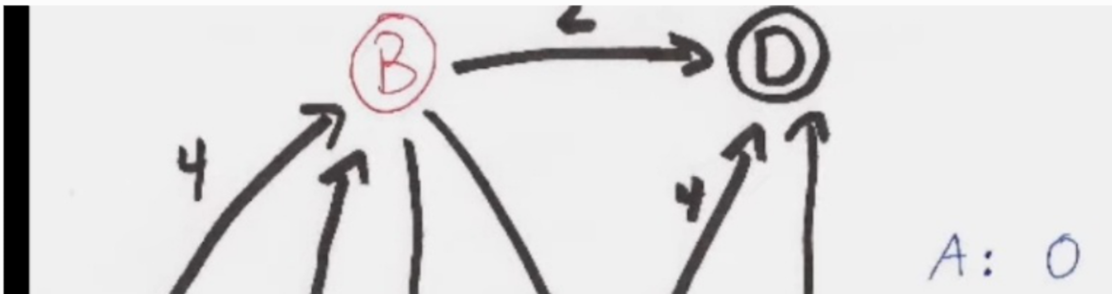


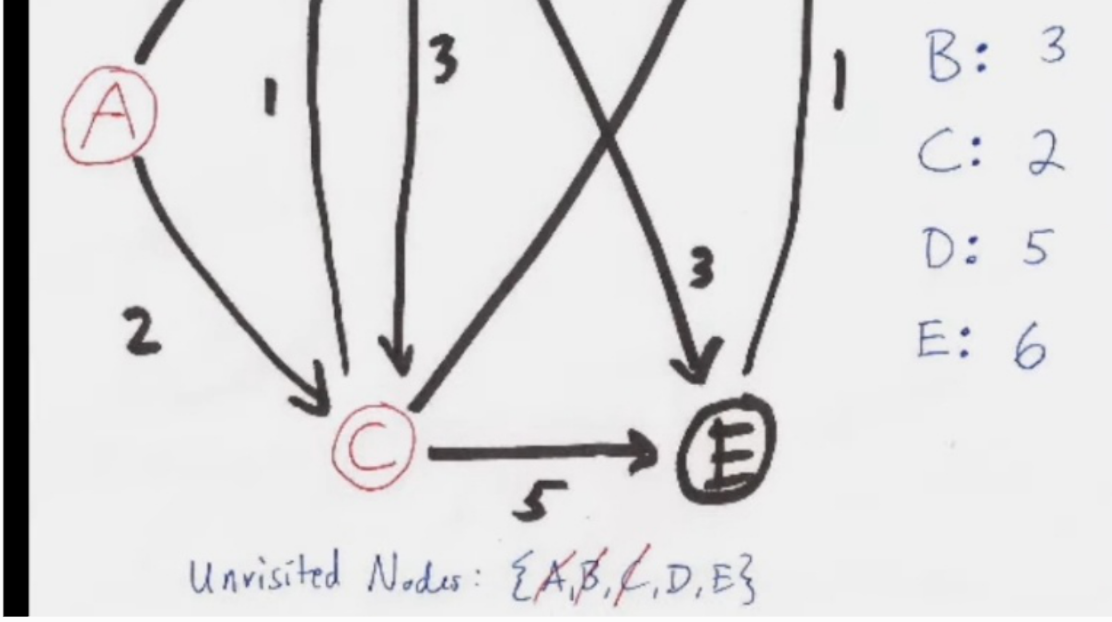


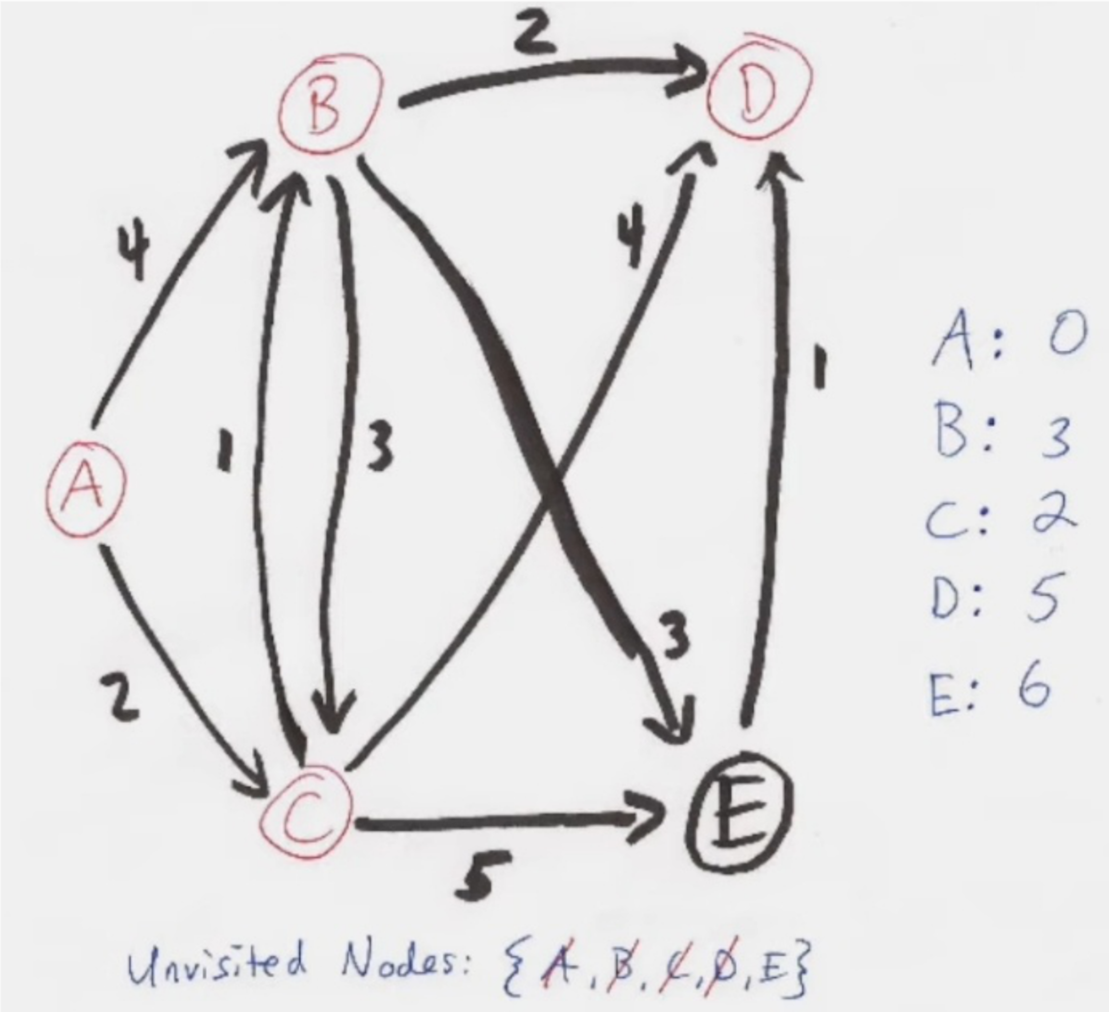

Unvisited Nodes: {~~A~~, B, C, D, E}

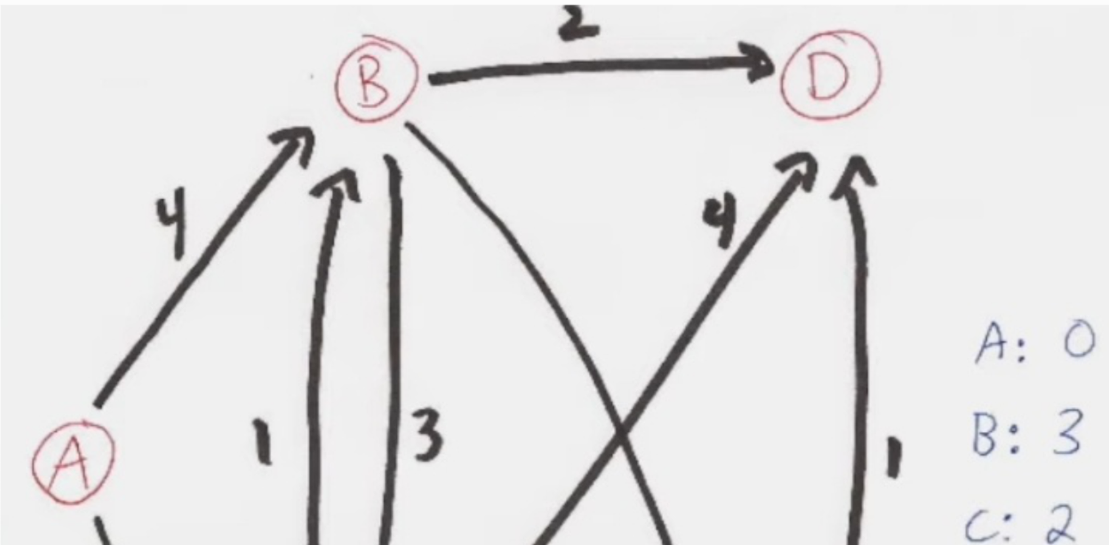

A
B
C
D
E
4
2
1
3
4
3
1
2
5
A : 0
B : 3
C : 2
D : 6
E : 7
Unvisited Nodes: {~~A~~, B, ~~C~~, D, E}

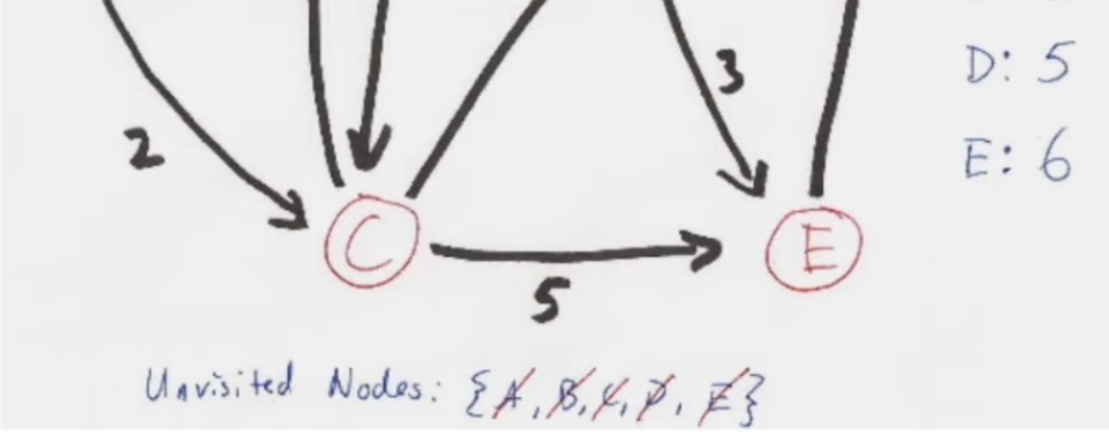

A
B
C
D
E
4
2
1
3
4
3
1
2
5
A : 0
B : 3
C : 2
D : 6
E : 7
Unvisited Nodes: {~~A~~, B, ~~C~~, D, E}

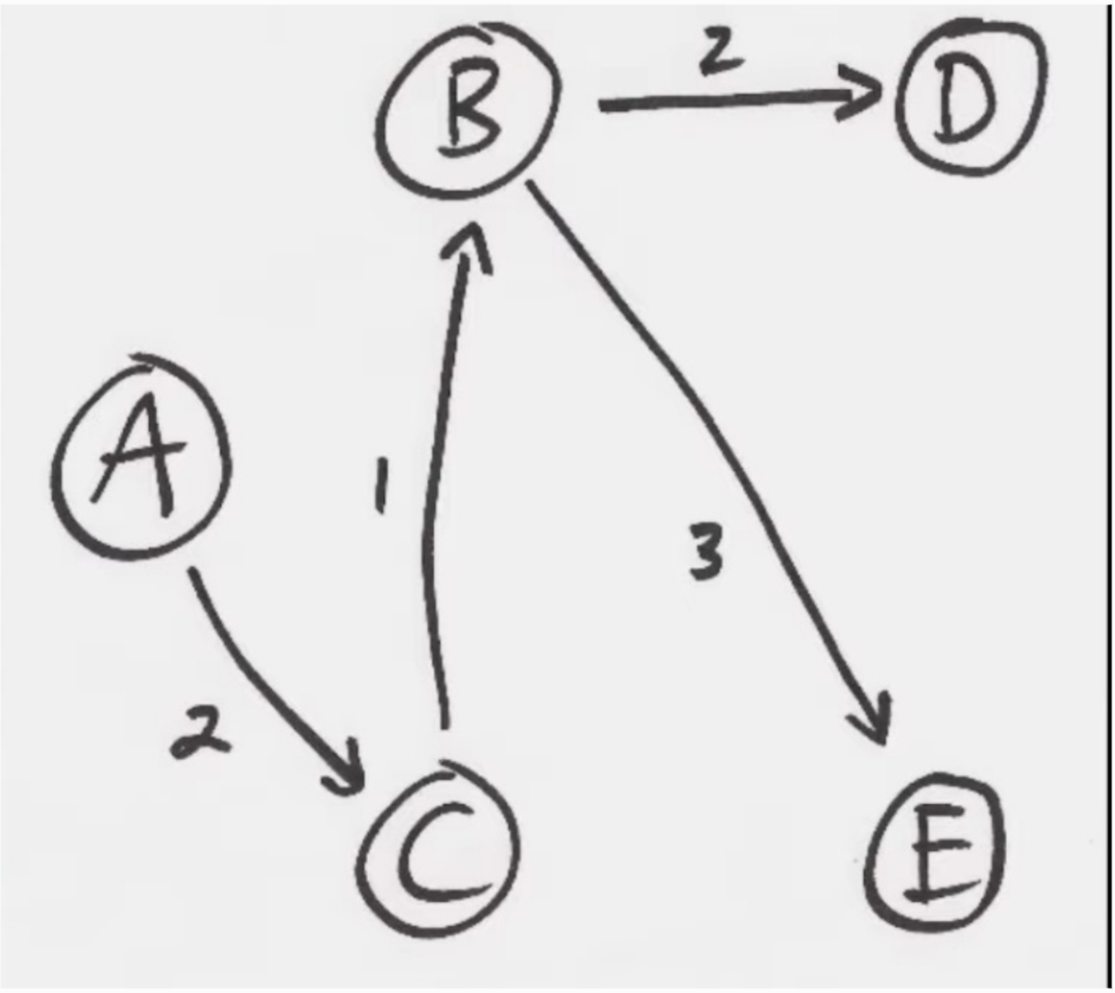

B
D
4
4
A: 0


A
C
E
1
3
3
1
2
5
B: 3
C: 2
D: 5
E: 6
Unvisited Nodes: {~~A~~, ~~B~~, ~~C~~, D, E}


B
D
A
C
E
2
4
4
1
3
1
3
2
5
A: 0
B: 3
C: 2
D: 5
E: 6
Unvisited Nodes: {~~A~~, ~~B~~, ~~C~~, ~~D~~, E}


B
D
A
2
4
4
1
3
1
A: 0
B: 3
C: 2





Time Complexity

$$O(|E| + |V| \log |V|)$$

```
procedure dijkstra(G,l,s)
Input:    Graph G = (V,E), directed or undirected;
          positive edge lengths {l_e : e ∈ E}; vertex s ∈ V
Output:   For all vertices u reachable from s, dist(u) is set
          to the distance from s to u.
```

```
for all u ∈ V:
   dist(u) = ∞
   prev(u) = nil
dist(s) = 0

H = makequeue(V)  (using dist-values as keys)
while H is not empty:
   u = deletemin(H)
   for all edges (u,v) ∈ E:
      if dist(v) > dist(u) + l(u,v):
         dist(v) = dist(u) + l(u,v)
         prev(v) = u
         decreasekey(H,v)
```