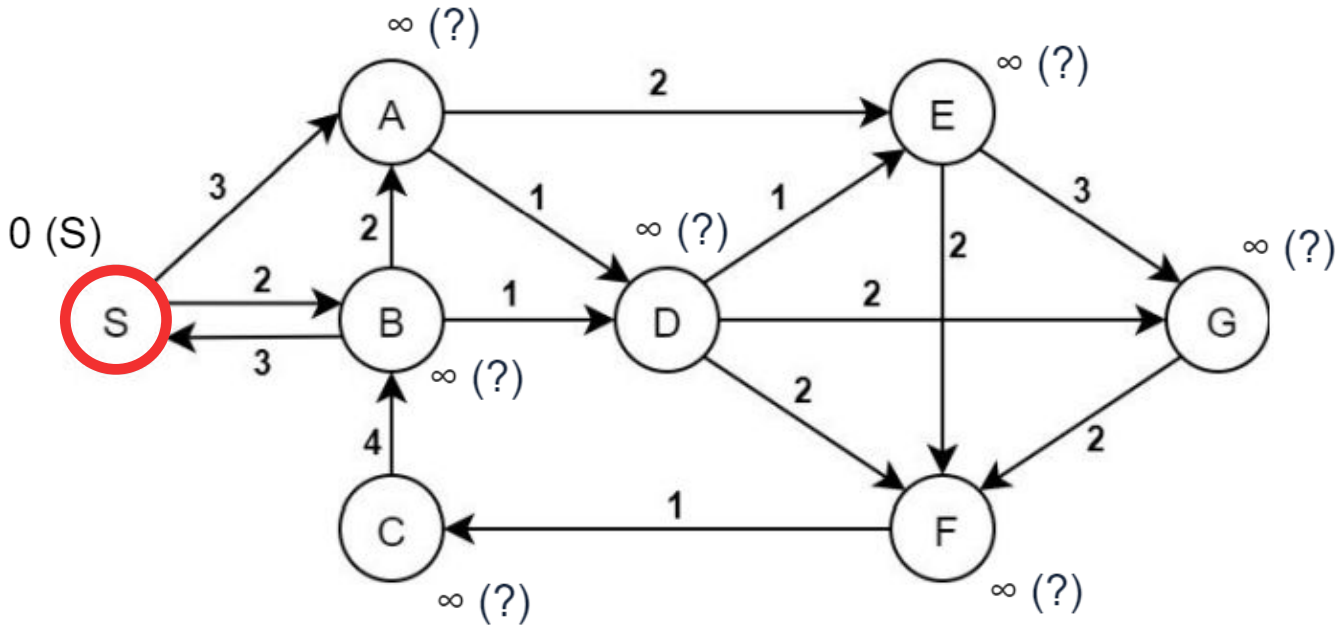
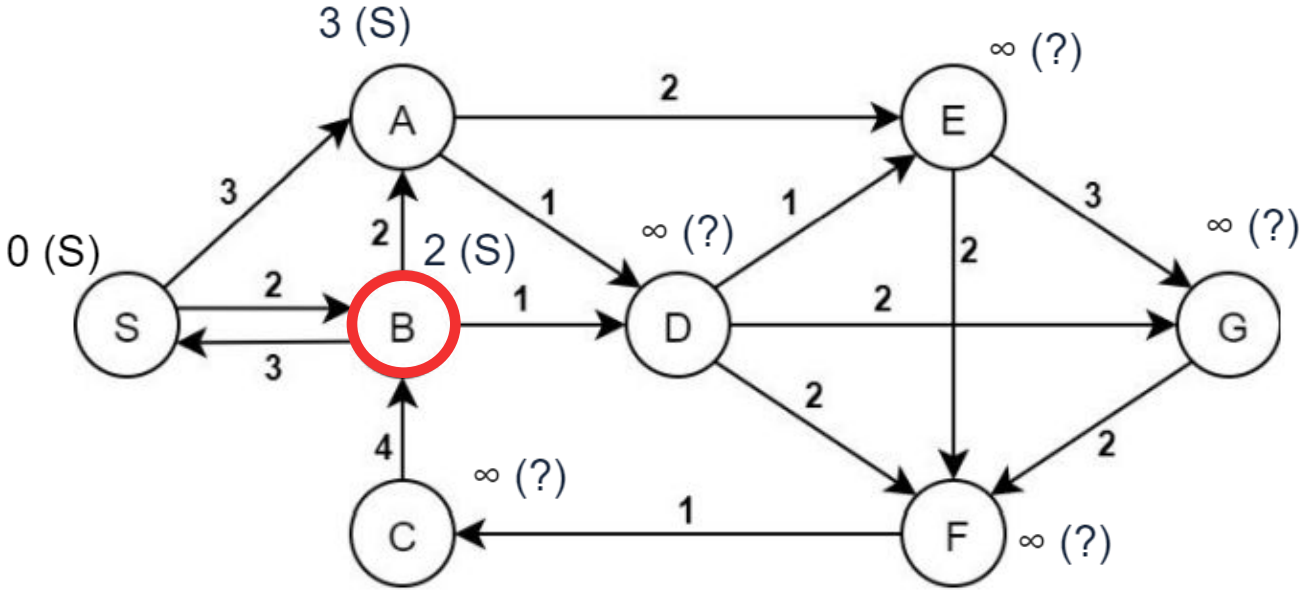
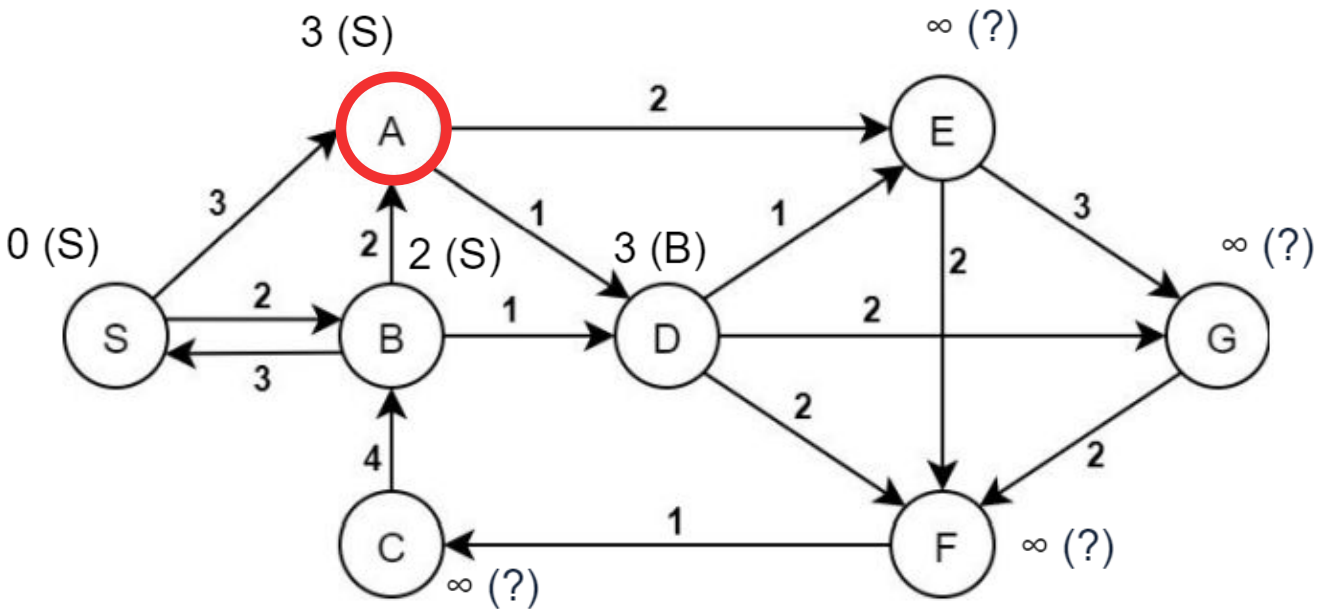
1.



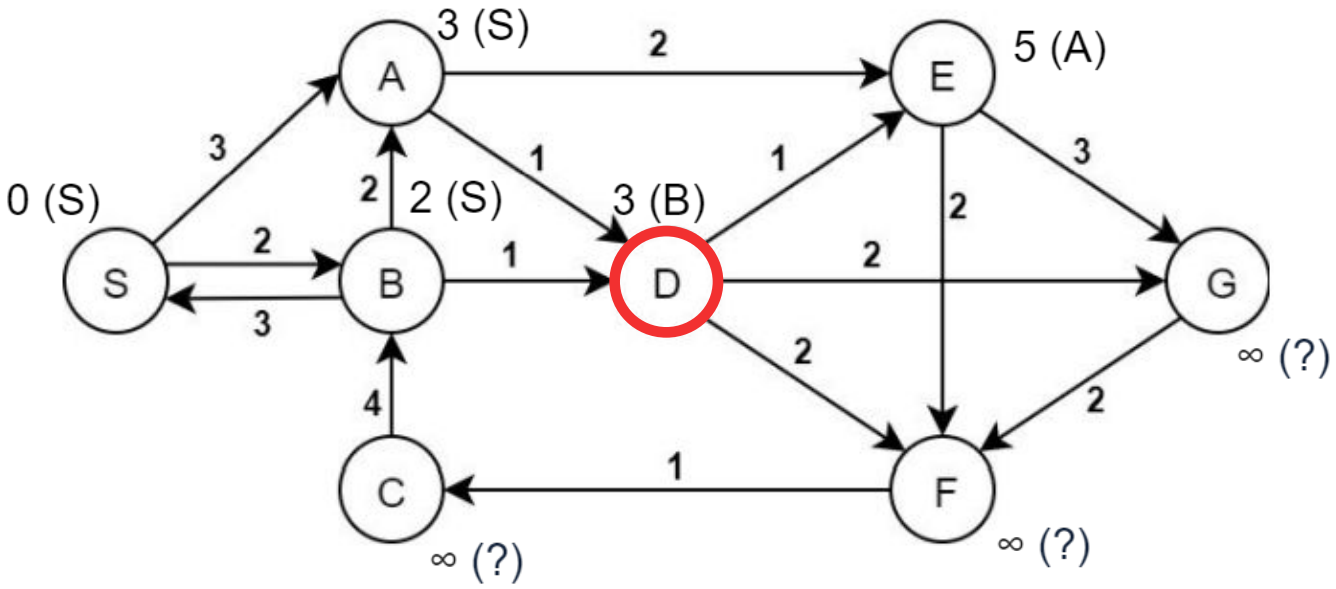
All nodes except S are unknown for now.
So S is now the chosen node to operate on.



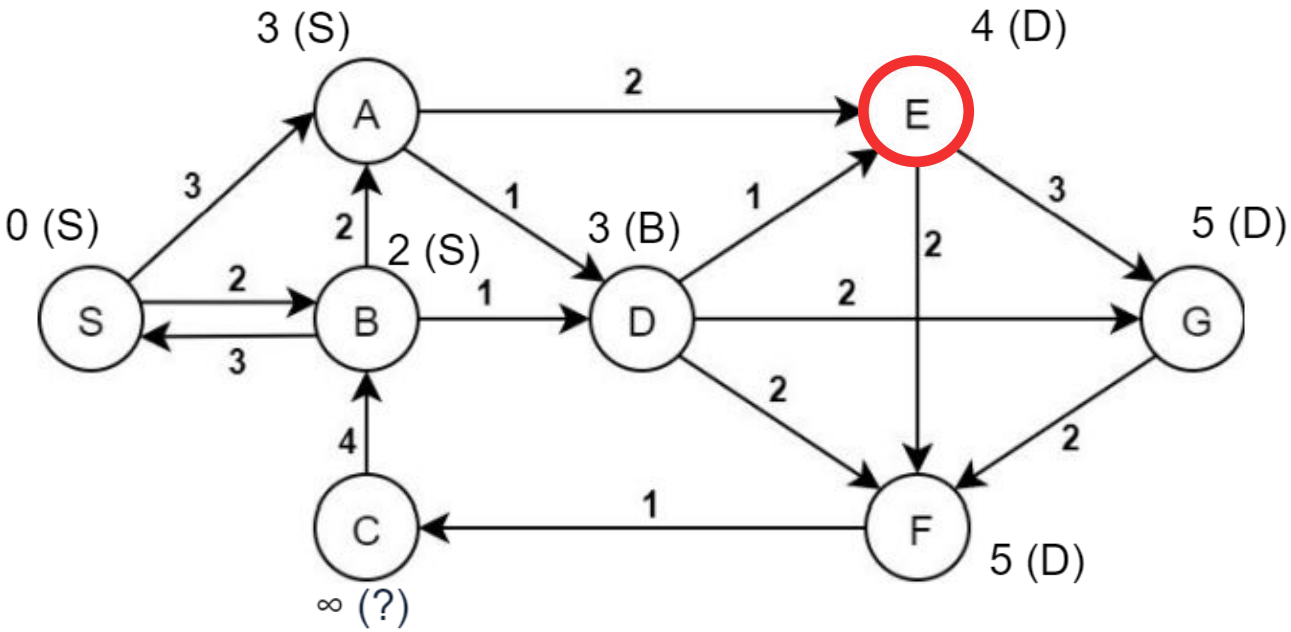
After the first iteration. 2 unkown nodes (A,B). Pick the least distance which is B. Then B is known now. Let's operate on B.



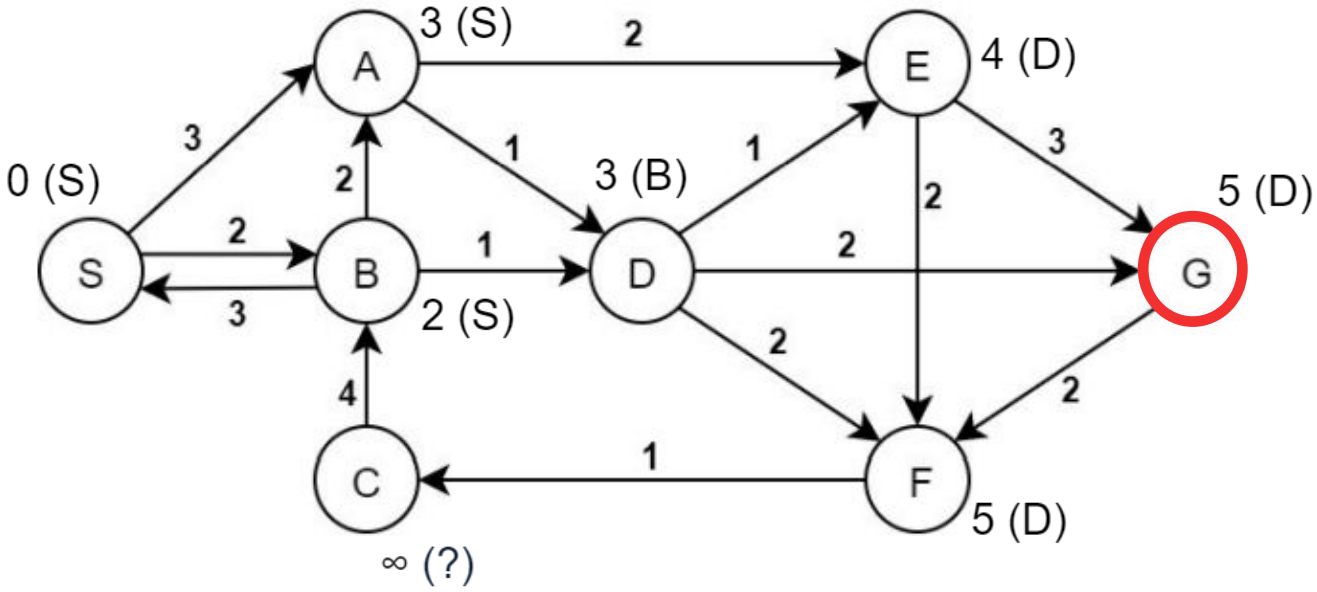
After second iteration D and A are reached. Now the decision is arbitrary since unknown vertices have the same distance. Let's pick A, mark it as known and operate on.



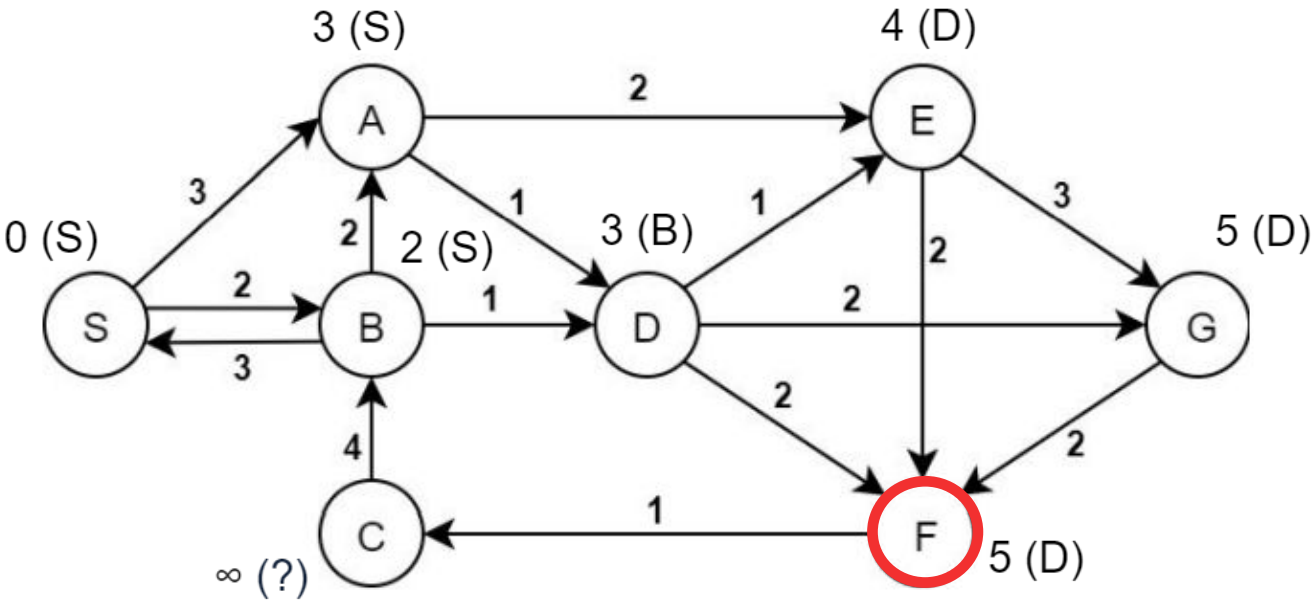
After the third iteration D and E are visited. For D no change. Move to D because least distance and operate on. D is known now.



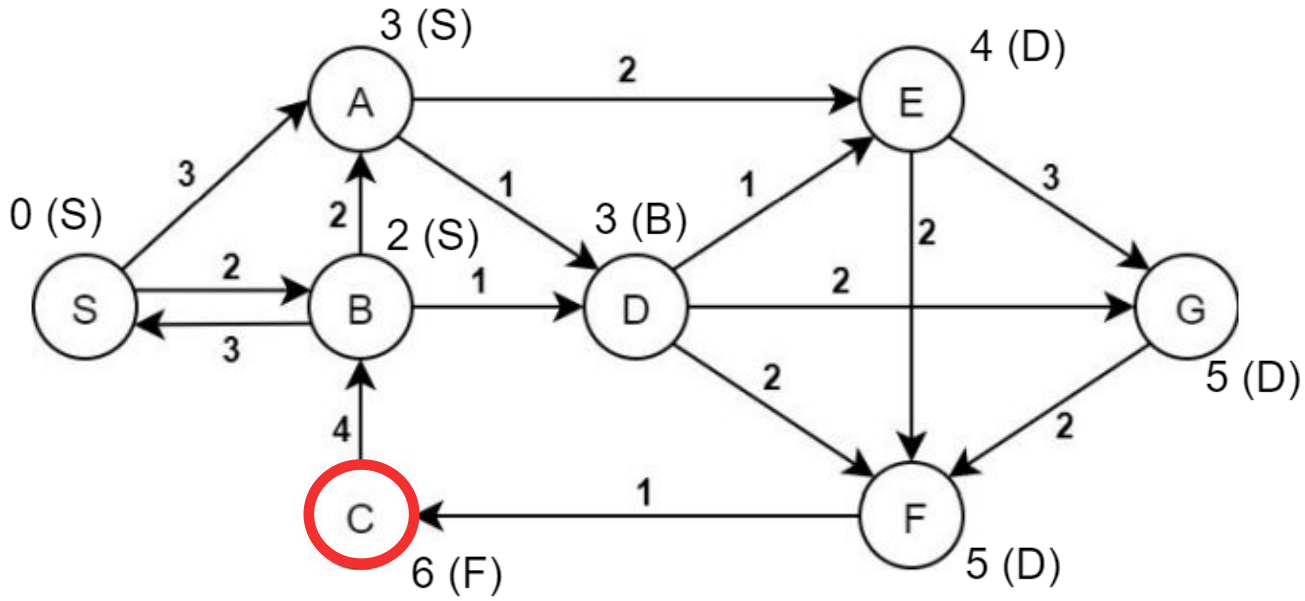
After the fourth iteration nodes E,G and F are visited from D. E is updated to 4. Let's pick E (now known) and move on since it has the least distance.



No updates for G and F. Decision is arbitrary. Pick G, mark as known and operate on it.

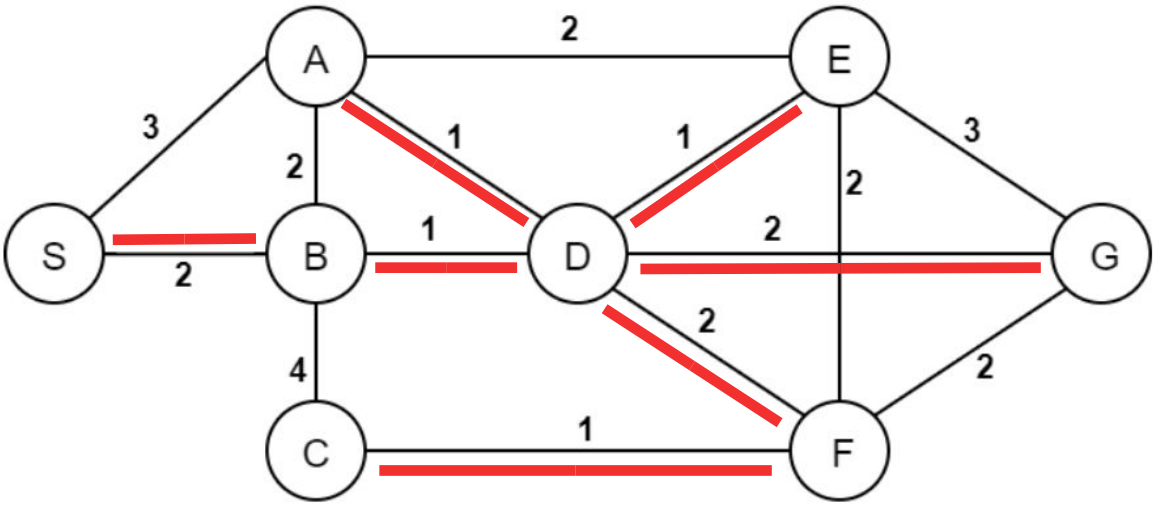


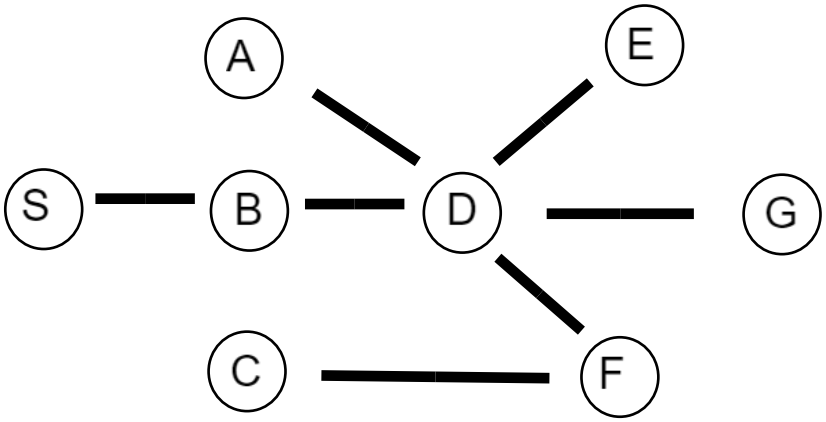
1 edge. Only F is visited, no update. Now known. Move on from F.



Now the Dijkstra's Algorithm is complete. All nodes visited. Shortest paths are founded.

2. Visited = {S} | Least distance B.
Visited = {S,B} | A,C,D are reachable. D with the least distance.
Visited = {S,B,D} | All nodes are reachable. Two edges with one, decision is arbitrary.
Visited = {S,B,D,A} | Now pick E because of the distance 1.
Visited = {S,B,D,A,E} | F and G have the same distance. Decision is arbitrary.
Visited = {S,B,D,A,E,G} | Then F.
Visited = {S,B,D,A,E,G,F} | Finally C.
Visited = {S,B,D,A,E,G,F,C} | Total cost 2+1+1+1+2+2+1 = 10





3.

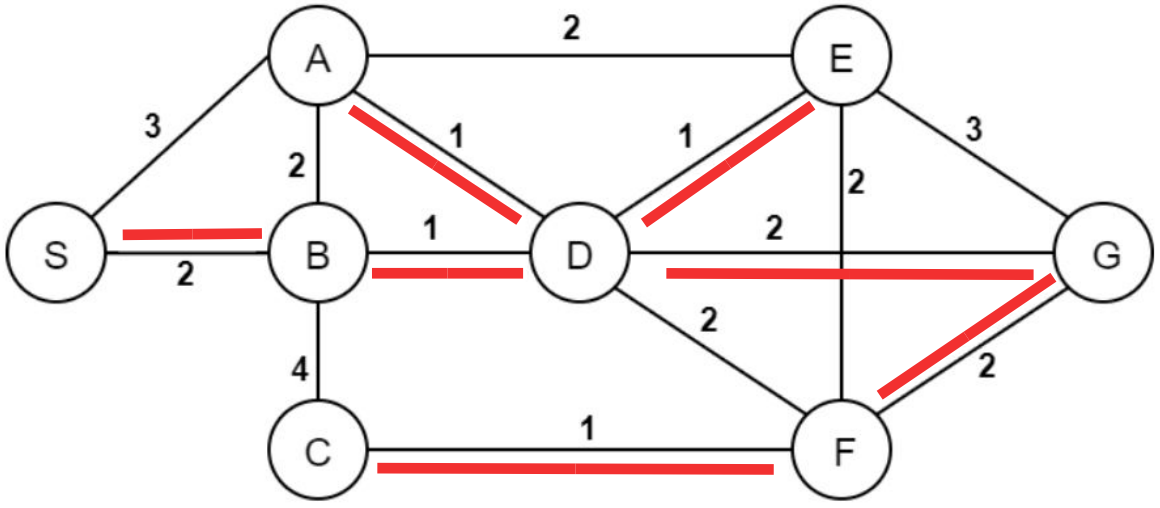
Start from smallest distance. Choose next smallest without making a cycle.There are four 1s so take any of them.

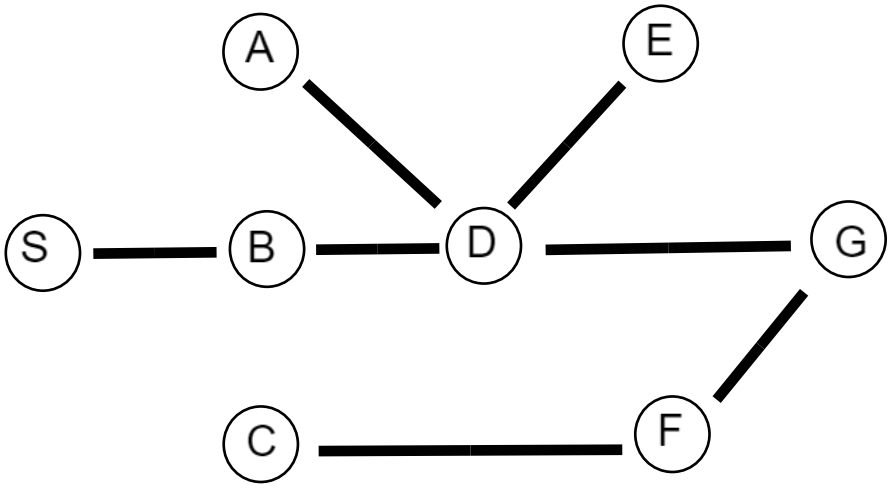
B-D, A-D, D-E, C-F

Then three 2s. Take any of them.

B-S, D-G, F-G

All nodes are visited. No cycle. Total cost 1+1+1+1+2+2+2 = 10

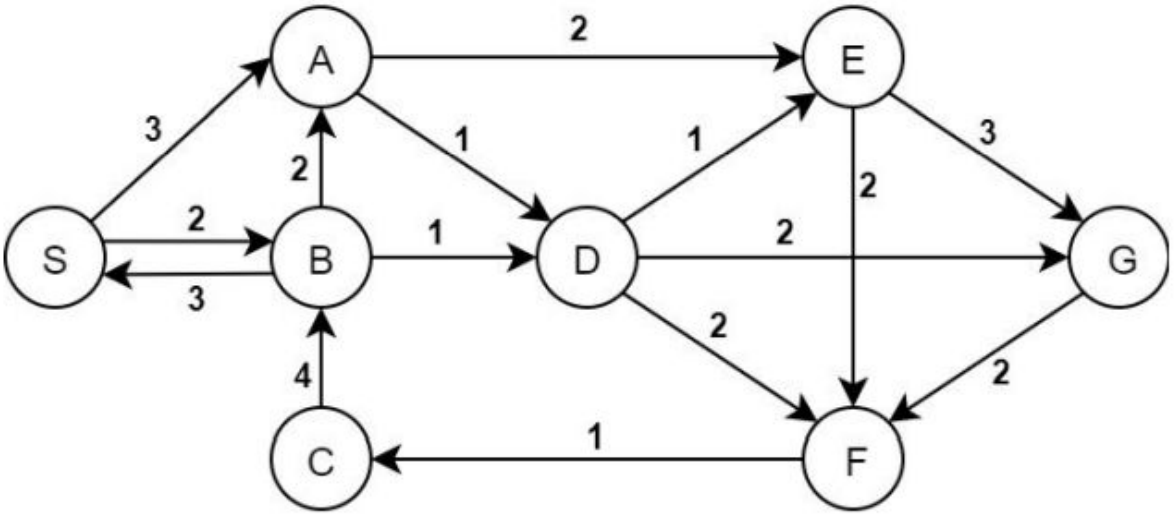


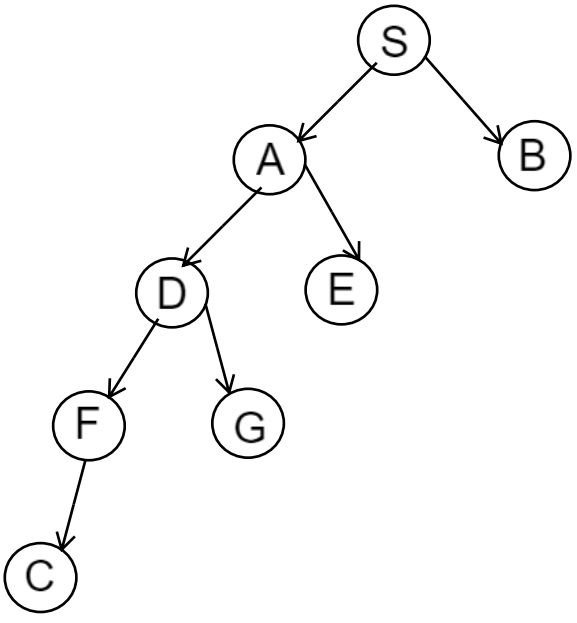


4.

Actually BFS can't be used in a weighted graph but let's assume all edges with cost 1 and continue.
Start from S. We can go to nodes A and B from S.
Then choose A. Go from A to D and E.
We already visited A and D so B is pointless now.Then move on with D. From D g to F and G since we already visited E.
Lastly go to C from F

S|A,B|D,E|F,G|C





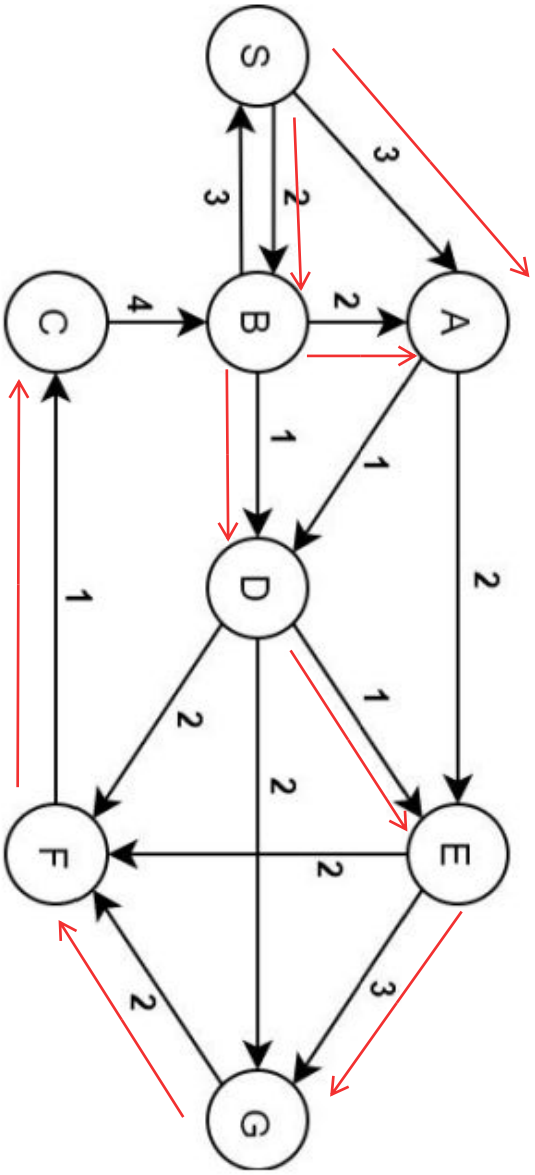
5.

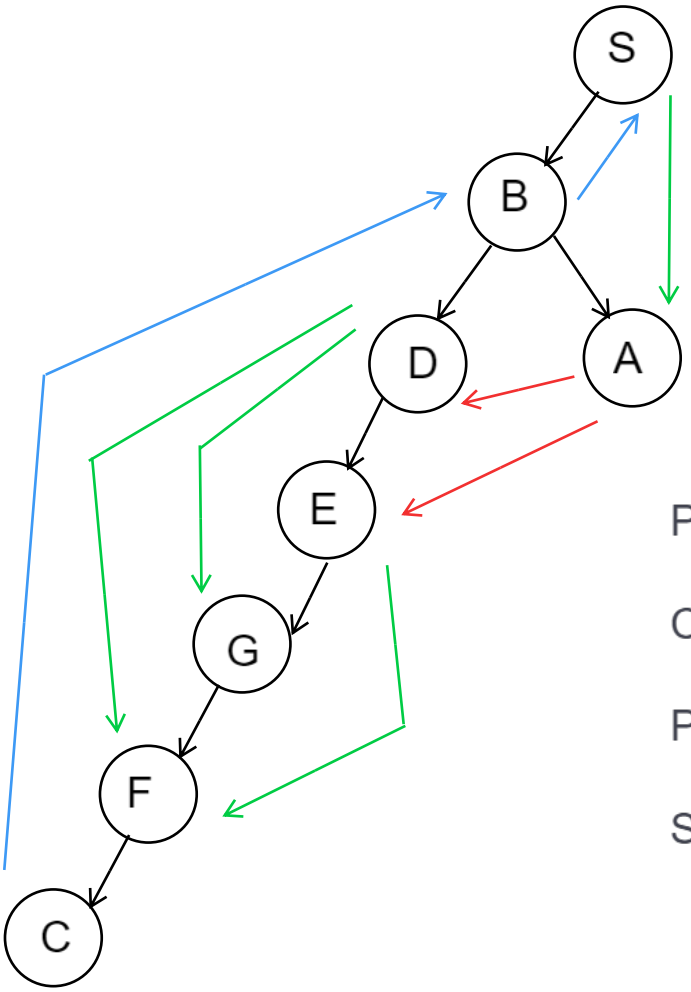
Starting from S we can go to A and B. I'll go with the B. Then traverse recursively.

S,B

Then go deep into D,E,G,F,C. Notice these nodes branch out from node B.
After that go back to S where A and B branch out. Finally visit A.

S,B,D,E,G,F,C,A





Postorder (Left, Right, Root) :

C F G E D A B S

Preorder (Root, Left, Right) :

S B D E G F C A

Black : Tree arcs

Green : Forward arcs

Blue : Backward arcs

Red : Cross arcs

6.

Notice that only node with in-degree 0 is D so start from there.
D,
Then move on with B since after removing D it has in-degree 0.
D,B
After removing B, nodes with in-degree 0 are A and E. Decision is arbitrary.
D,B,A
Now move on with E.
D,B,A,E
Only node with in-degree 0 is F. Remove it.
D,B,A,E,F
Two nodes left with out-degree 0. You can take whatever you want.
D,B,A,E,F,C,G