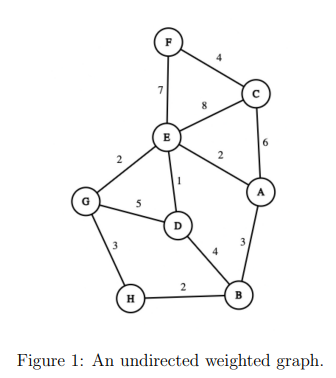
Doruk Benli

29182

ASSIGNMENT 4

Question 1) Trace the Dijkstra’s weighted shortest path algorithm on the graph given in Figure 1. Use vertex E as your start vertex.



Step 1) At first we mark All vertices distances as infinity then we look for E’s adjacent vertices. E’s neighbors are as follows: F(7), C(8), A(2), G(2),D(1), since vertex D is the nearest with distance 1, we mark D as visited and update the distances of F,C,A,G,D to the given numbers above instead of infinity.

Step 2) looking at unvisited vertices and looking for possible updates, the smallest distances are E🡪G and E🡪A with both distances equal to 2, choice is arbitrary so we can pick A and mark it as visited. There are no possible updates on the paths of vertices. H is now 2+3+2 = 7 instead of infinity.

Step 3) Then since E🡪G is the smallest distance that is not yet visited, we can mark G as visited, We can update H as 5 since E🡪G is 2 and G🡪H is 3.

Step 4) Unvisited known vertices are F(7), C(8), H(5), B(5) we can pick H since its smallest with B, so we pick H and mark it as visited. By addition of H we have no updates.

Step 5) We can pick B since it has the smallest distance to in unvisited known vertices. There are 2 paths with equal distances, Path1) E🡪A🡪B with distance d = 5, and path2) E🡪D🡪B with distance d =5, since both path has equal lengths, our choice can be made arbitrarily. So we mark B as visited. No updates can be done.

Step 6) We can pick F(7) since it has shorter path then, (E🡪F, Distance = 7) C(8), so we mark F as visited.

Step 7) C is the last unvisited vertex, we add it to our path E🡪C, Distance = 8. We mark C as visited.

List of Vertices with shortest paths from vertex E.

D: E🡪D, Distance = 1,

A: E🡪A, Distance = 2,

G: E🡪G, Distance = 2,

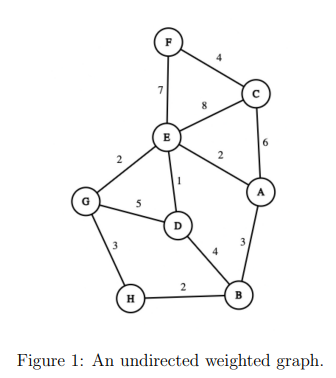
H: E🡪G🡪H, Distance = 5,

B: E🡪A🡪B, Distance = 5,

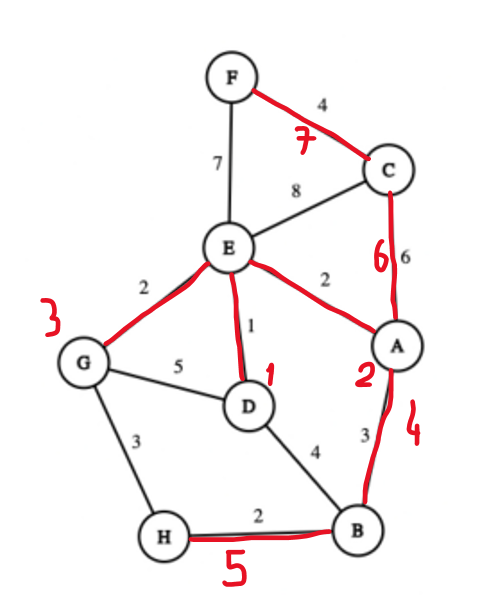
F: E🡪F, Distance = 7,

C: E🡪C, Distance = 8

Question 2) Trace the Prim’s minimum spanning tree algorithm on the graph in Figure 1. Use vertex E as your start vertex.



Prims Algorithm: At each stage, add an edge(U,V) to the tree such that U is in tree and V is not and in addition to that (U,V) has smallest cost



The red numbers are the order which vertex pair(U,V) is added to the Minimum spanning Tree.

Step 1) Add(E,D) Step 4) Add(A,B) Step 7) Add(C,F)

Step 2) Add(E,A) Step 5) Add (B,H)

Step 3) Add(E,G) Step 6) Add (A,C)

Question 3) Trace the Kruskal’s minimum spanning tree algorithm on the graph in Figure 1.

Kruskal’s Algorithm: Always select the min cost Edge that wont form a cycle!

Step1) select (E,D) Step4) select (H,B)

Step2) select (E,A) Step 5) select (G,H)

Step 3) select (E,G) Step6) select (F,C)

A picture containing chart

Description automatically generatedStep7 ) select (A,C)

The above graph shows the edges that needed to be added and the red numbers indicate the order they are added to the Kruskal’s Minimum spanning Tree.

Question 4) Trace the breadth-first search traversal algorithm on the graph in Figure 1 starting from vertex E.

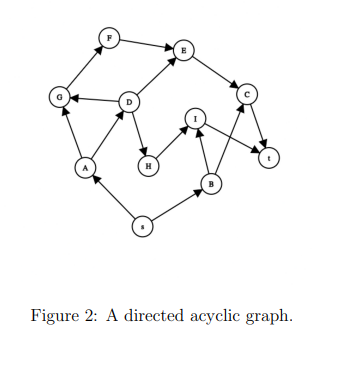
Solution:

We can easily use a queue to trace the BFS algorithm, Since we start from vertex E, we can directly enqueue D,A,G,F,C since these vertices are all the neighbors of E, once i make them known and enqueue each vertices, I can dequeue E and add it to my BFS path. Then I can explore D and look for unknown vertices that is not in queue, we have B so I enqueue B and deque D, then add D to my BFS path. Then next element In queue is A, we can look A’s adjacent vertices that is not known but both E, C and B are known, so We do not add any element to queue and directly dequeue A and add it to our BFS path. The next element In queue G, since it has an unknown adjacent vertex which is H, we add H to our queue and dequeue, G and add G to our BFS path. Then we next element in queue is F, check unknown neighbors of F but there is none, at this point we know all the vertices, so i directly dequeue F and add it to path. The next element in queue is C, since all vertices are known we do not add any element to queue and dequeue C and add it to BFS path, then we finish vertices with level: 1 the next element in queue is B since all vertices are known we do not add any element to queue and dequeue B and add it to our BFS path, the next and last element is H we dequeue H and add it to our BFS path.

BFS: E,D,A,G,F,C,B,H(this is the order of which we will visit, BFS is level order traversal)

Since BFS looks for a level order, and our first level vertices are D,A,G,F,C, so there can be many different solutions to this problem since I can change the order of exploration of the same level vertices.

Question 5) Find a topological ordering of the graph in Figure 2.



Topological sort algorithm: select vertex with indegree 0 and print it out then remove it and repeat until no vertex is left or every vertex is visited.

Step 1) we remove S, since its indegree is 0 then print it.

Step 2) we can remove A and B, since they both have an indegree zero after the removal of S so choice is arbitrary so remove A print it.

Step 3) we can remove D and B since they both have in-degree of 0, choice is arbitrary so remove D and print it.

Step 4) we can remove G and B since they both have in-degree of 0, choice is arbitrary so remove G print it.

Step 5) we can remove H and B since they both have in-degree of 0, choice is arbitrary so remove H print it.

Step 6) we can remove F and B since they both have in-degree of 0, choice is arbitrary so remove F print it.

Step 7) we can remove I remove I and print it.

Step 8) we can remove E and B since they both have in-degree of 0, choice is arbitrary so remove E print it.

Step 9) we can remove C remove C and print it.

Step 10) we can remove T remove T and print it.

Step 11) we can remove B so remove B and print it.