HackerRank For Work

Spring_2018_INFO6205_Se... 30 minutes

Question - 1

SCORE: **5 points**

What is the maximum number of edges in an acyclic undirected graph with n vertices?

- 2 * n -1
- n * (n - 1) / 2
- n - 1
- n * (n - 1)

Question - 2

SCORE: **5 points**

Which of the following statements is/are TRUE for an undirected graph?
A: Number of odd degree vertices is even
B: Sum of degrees of all vertices is even

- A
- B
- Both A and B
- Neither A nor B

Question - 3

SCORE: **5 points**

Classroom question

Select the numbers on the whiteboard

- 3
- 4
- 19
- 29
- 47
- 74
- 81

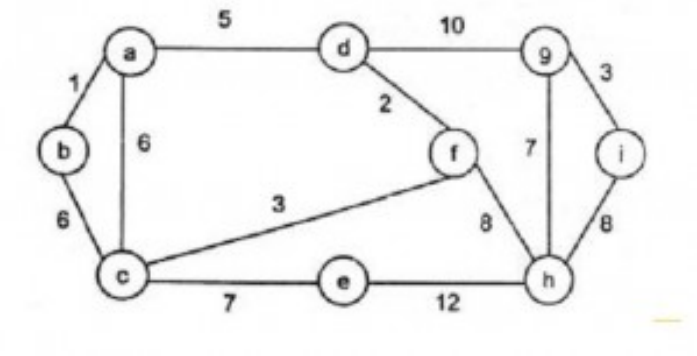
Question - 4

SCORE: **10 points**

MST

Problem Statement

For the undirected, weighted graph given below, which of the following sequences of edges represents a correct execution of Kruskal's algorithm to construct a Minimum Spanning Tree?



A. (c, e), (c, f), (f, d), (d, a), (a, b), (g, h), (h, f), (g, i)
B. (d, f), (f, c), (d, a), (a, b), (c, e), (f, h), (g, h), (g, i)
C. (h, g), (g, i), (h, f), (f, c), (f, d), (d, a), (a, b), (c, e)
D. (a, b), (d, f), (f, c), (g, i), (d, a), (g, h), (c, e), (f, h)

Answer is: <blank 1>

Answers

<blank 1> : [D, d, (a, b), (d, f), (f, c), (g, i), (d, a), (g, h), (c, e), (f, h)]

Question - 5

SCORE: **30 points**

Kruskal MST

Given an Edge Weighted Undirected Graph G, Return a Queue with all edges of its MST
Please finish the codes below using given method:

```
// Egde class has method:
// public double weight()
// public Vertice getOneVertice()
// public Vertice getAnotherVertice()
// EdgeWeightedGraph class has method:
// public int getVerticeNum() (return the number of vertice of this graph)
public Queue<Edge> KruskalMST(EdgeWeightedGraph G) {
    // mst used to store all edges of MST;
    // Queue<Edge> has method:
    // public void enqueue(Edge e)
    // public int getSize()
    Queue<Edge> mst = new Queue<Edge>();
    // PriorityQueue<Edge> has methods:
    // public Edge getMin() (return the edge with min weight)
    // public void remove(Edge e) (delete the edge within PQ)
    // public boolean isEmpty() (return true if empty, false if not)
    PriorityQueue<Edge> pq = new PriorityQueue<Edge>
(G.edges());
    // UnionFind has methods:
    // public boolean connected(Vertice a, Vertice b) (return true if
connected, false if not)
    // public void union(Vertice a, Vertice b)
    UnionFind uf = new UnionFind(G.V());

    // TODO
```

```
    return mst;
}
```