Homework assignment 7:

Due date: Friday, December 8 2017 at 03:30pm

1. Draw the below graphs and then write the size, order, deg(v), adj(v), of each graph. (and $\deg^+(v)$, $\deg^-(v)$ for the directed graphs)

   E={{a,b},{b,e},{a,c},{a,d},{c,e},{e,f},{c,f}}
   E={{a,f,3},{b,c,4},{a,c,1},{a,d,2},{c,e,3},{c,f,1}}
   E= {(j, a),(j, g),(a, b),(a, e),(b, c),(c, k),(d, e),(e, c),(e, f),(e, i),(f, k), (g, d),(g, e),(g, h),(h, e),(h, i),(i, f),(i, k)}.
   E={(a,b,2),(a,c,1),(b,f,3),(c,b,1),(c,g,4),(d,c,2),(d,e,3),(e,f,7),(g,f,5)(g,h,4)}

2. The graph Q_n, $n \geq 1$, has vertex set equal to the set of all binary strings of length n. Moreover, two vertices are adjacent iff they differ in at most one bit place. For example, in Q_3, 000 is adjacent to 010, but not to 011. Draw Q_1, Q_2, and Q_3. Show that Q_3 has a Hamilton cycle, (a cycle that visits all the **vertices exactly once**).
3. Provide formulas for both the order and size of Q_n. Explain.
4. Consider the directed graph where vertices are reachable tic-tac-toe board positions and edges represent valid moves. What are the in-degree and out-degree of the following vertex? (It is O's turn.)

X	O	X
	O	
	X	

5. Starting at vertex 000, perform a BFS of Q3. Assume all adjacency lists are in numerical order. For example, (000, 001) occurs before (000, 010). Show the resulting spanning trees.
6. Draw the directed graphs and perform
   a. Breadth-First Search (BFS) algorithm:
      - ✓ To determine the shortest paths starting at vertex **a** to every other node.
      - ✓ Show the resulting spanning tree.

   b. Depth-First Search (DFS) to explore the whole graph:
      - ✓ Record the start/end time for all the vertices.
      - ✓ Show the resulting spanning forest
      - ✓ Label the name of the edges.
      - ✓ Write the topological order of the vertices (if no cycle = no back edge) (Show the state of the linked list each time you insert a new vertex)

   E={(a,b),(a,c),(b,f),(c,b),(c,g),(d,c),(d,e),(e,f),(g,f)(g,h)}

   E= {(j, a),(j, g),(a, b),(a, e),(b, c),(c, k),(d, e),(e, c),(e, f),(e, i),(f, k), (g, d),(g, e),(g, h),(h, e),(h, i),(i, f),(i, k)}.

7. What is the running time of
    a. Breadth-first search
    b. Depth-first search,

    as a function of |V | and |E|, if the input graph is represented by an adjacency matrix instead of an adjacency list?
8. Draw the weighted directed graphs whose edges-weights are given by

    E={(a,b,2),(a,c,1),(b,f,3),(c,b,1),(c,g,4),(d,c,2),(d,e,3),(e,f,7),(g,f,5)(g,h,4)}

    E={(a, b, 2),(b, g, 1),(g, e, 1),(b, e, 3),(b, c, 2),(a, c, 5),(c, e, 2),(c, d, 7),(e, d, 3), (e, f, 8),(d, f, 1),(d, a, 2)}.

    Perform Dijkstra's algorithm to determine the distances and shortest paths from a to every other node.

9. Draw the weighted undirected graphs whose edges-weights are given by

    E={{a,b,2}, {a,c,1}, {b,f,3}, {c,b,1}, {c,g,4}, {d,c,2}, {d,e,3}, {e,f,7}, {g,f,5}, {g,h,4}}

    E={{a, b, 3}, {a, d, 4}, {a, e, 4} , {b, c, 10}, {b, e, 2}, {b, f, 3}, {c, f, 6}, {c, g, 1}, {d, e, 5}, {d, h, 6}, {e, f, 11}, {e, h, 2}, {e, i, 1}, {f, g, 2}, {f, i, 1}, {f, j, 11}, {h, i, 4}, {i, j, 7}}.

    a. Perform Kruskal's algorithm
    b. Perform Prim's algorithm

    to obtain a minimum spanning tree (MST) for G.

10. Does Prim's and Kruskal's algorithm work if negative weights are allowed? Explain.
11. Explain how Prim's and/or Kruskal's algorithm can be used to find a maximum spanning tree. (Hint: neither algorithm needs modification, if the input graph is suitably modified.)
12. Give an algorithm that determines whether or not a given **undirected** graph G = (V,E) contains a cycle. Your algorithm should run in O(V) time.
13. Give a linear-time algorithm that determines if a simple graph has any odd cycles. Hint: perform a breadth-first traversal and mark the visited nodes with one of two colors (either red or blue). Whenever a (parent) node that is removed from the queue reaches an unvisited node, mark that node and give it the opposite color of its parent. What happens when the child node is already visited/marked with the same color? with the opposite color?

    ColorGraph(G):

    ```
     start at any node S, and color it blue
     Q.push(s)
     while Q not empty
        x = pop(Queue)
        for v=adj[x]
                 if v isn't colored
    ```

```
                color it the opposite color of x
                Q.push(v)
        if v is colored and is the same color as x
                return…..?                  //Break and return…?
return…?                                    //what should you return?
```