After Lecture 24 & 25 & 26 – Answer any questions on HW6 (due today)
Practice Problems (all taken from previous exams)

1. The edges used in a Depth First Search traversal of a graph will result in?
   a) Linked List
   b) Tree
   c) Graph with back edges
   d) Min Heap
2. In Depth First Search, how many times a node is checked to see if it has been visited yet?
   a) Once
   b) Twice
   c) Equivalent to number of indegree of the node
   d) Thrice
3. Regarding implementation of Breadth First Search using queues, what is the maximum difference in the distance from the source for two nodes present in the queue? (considering each edge length 1)
   a) Can be anything
   b) 0
   c) At most 1
   d) Insufficient information
4. Topological sort can be applied to which of the following graphs?
   a) Undirected Cyclic Graphs
   b) Directed Cyclic Graphs
   c) Undirected Acyclic Graphs
   d) Directed Acyclic Graphs
5. Which of the following is false?
   a) The spanning trees do not have any cycles
   b) MST have $n-1$ edges if the graph has n edges
   c) Edge e belonging to a cut of the graph (partitions the vertices of a graph into two disjoint subsets), if has the weight smaller than any other edge in the same cut, then the edge e is present in all the MSTs of the graph.
   d) Removing one edge from the spanning tree will not make the graph disconnected
6. Which of the following is false about the Kruskal's algorithm?
   a) It is a greedy algorithm

b) It constructs MST by selecting edges in increasing order of their weights

c) It can accept cycles in the MST

d) It uses disjoint set data structure

7. Kruskal's algorithm (pick minimum edge in the graph between two vertices that are not yet in the same connected component) is best suited for the dense graphs than the Prim's algorithm (pick minimum edge from visited to unvisited vertex).

   a) True

   b) False

8. Reword the statement below as a theorem about graphs and then prove it. Assume that is A is a friend of B, then B is a friend of A and that for all A, A is not a friend of A.

   - In any group of $n \geq 2$ people, there are two people with the same number of friends in the group.

9. Argue that in a breadth-first search, the value $d[u]$ assigned to a vertex u is independent of the order in which the vertices in each adjacency list are given. Using the graph shown as an example, show that the breadth-first tree computed by BFS can depend on the ordering within adjacency lists.

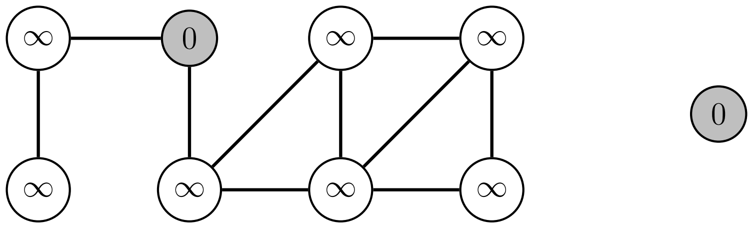

Figure 13.1:

10. 10a. Show how on a directed graph that depth first search starting at vertex u can result in vertex v not being reachable from u even though both u and v have incoming and outgoing edges.

    10b. If a directed graph contains a path from u to v, show that it is not necessary that the $\text{S}(v) < \text{F}(u)$. S( ) is the depth first search start time and F( ) is the depth first search finish time.

11. Bob loves foreign languages and wants to plan his course schedule to take the following nine language courses: LA15, LA16, LA22, LA31, LA32, LA126, LA127, LA141, and LA169. The course prerequisites are:

LA15: (none)

LA16: LA15 is prerequisite

LA22: (none)

LA31: LA15 is prerequisite

LA32: LA16 and LA31 is prerequisite

LA126: LA22 and LA32 is prerequisite

LA127: LA16 is prerequisite

LA141: LA16 and LA22 is prerequisite

LA169: LA32 is prerequisite

Find a sequence of courses that allows Bob to satisfy all the prerequisites.

12. Let e be a maximum-weight edge on some cycle of $G = (V, E)$. Prove that there is a minimum spanning tree of $G' = (V, E - \{e\})$ that is also a minimum spanning tree of G. That is, there is a minimum spanning tree of G that does not include e.

13. In this problem, we give pseudocode for different algorithms. Each one takes a graph as input and returns a set of edges T. For each algorithm, you must either prove that T is a minimum spanning tree or prove that T is not a minimum spanning tree. Also describe the most efficient implementation of each algorithm, whether or not it computers a minimum spanning tree.

Algorithm 13.1 Maybe Minimum Spanning Tree A

```
function MAYBE-MST-A($G$, $w$)
    sort the edges into decreasing order by weight
    $T \leftarrow E$
    for all edge $e$, taken in increasing order by weight do
        if $T - \{e\}$ is a connected graph then
            $T \leftarrow T - \{e\}$
        end if
    end for
    return $T$
end function
```

Algorithm 13.2 Maybe Minimum Spanning Tree B

```
function MAYBE-MST-B($G$, $w$)
    $T \leftarrow \emptyset$
    for all edge $e$, taken in arbitrary order do
        if $T + \{e\}$ has no cycles then
            $T \leftarrow T + \{e\}$
        end if
    end for
    return $T$
end function
```

Algorithm 13.3 Maybe Minimum Spanning Tree C

```
function MAYBE-MST-C($G$, $w$)
    $T \leftarrow \emptyset$
    for all edge $e$, taken in arbitrary order do
        $T \leftarrow T + \{e\}$
        if $T$ has a cycle $c$ then
            $e' \leftarrow$ the maximum weight edge on $c$
            $T \leftarrow T - \{e'\}$
        end if
    end for
    return $T$
end function
```