Algorithm 2016 fall

Homework 4

1. Use GREEDY-ACTIVITY-SELECTOR to solve activity-selection problem. Suppose that instead of always selecting the first activity to finish, we instead select the last activity to start that is compatible with all previously selected activities. Describe how this approach is a greedy algorithm, and prove that it yields an optimal solution

2. List two problems which can use greedy algorithm and two problems which can't use it. Please explain the difference between them?

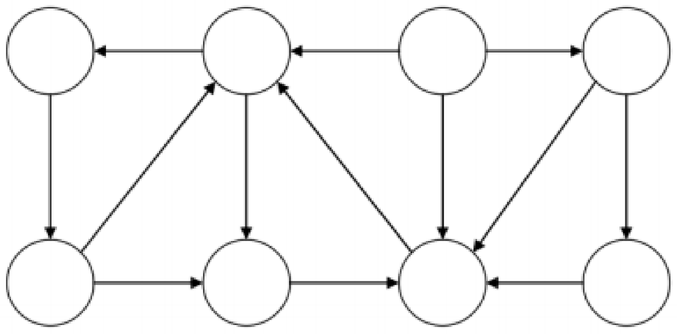


Figure 1: A directed graph.

3. Is a directed graph in Figure 1 acyclic? (Hint: Use the depth-first-search algorithm DFS.)

4. Find the strongly connected components on a directed graph in Figure 1.

5. Find the Topological Sort on a directed graph in Figure 2(Use the depth-first-search

algorithm DFS.)

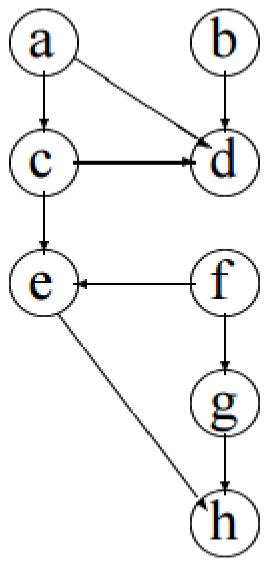
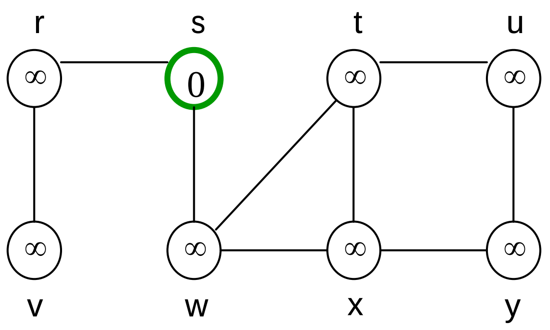


Figure 2

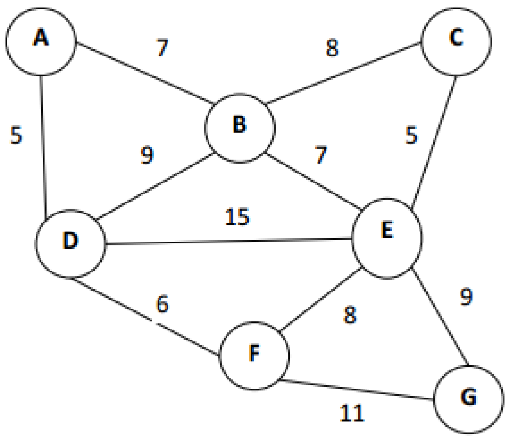
6. Illustrate the progress of breadth-first search (BFS) on the following sample graph.



7. The Depth-First-Search algorithm doesn't work for directed graphs. (True or False and explain briefly)

8. The topological sort of an arbitrary directed graph G = (V, E) can be computed in linear time. (True or False)

9. Given the following graph G=(V, E)



(a) Please use Kruskal's algorithm step by step.

(b) Please choose the most correct statement for Kruskal's algorithm.

(I) this is a greedy algorithm

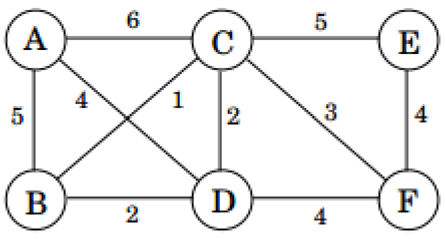
(II) its greedy rules are starting with any root node and adding the frontier edge with the smallest weight

(III) its greedy rules are adding edges in decreasing weight and skipping those whose addition would create a cycle

(IV) by using a disjoint set data structure, its running can achieve O($ElgV$)

(V) Kruskal's algorithm can have better performance than Prim's algorithm if edges are already sorted

10. Given the following graph G=(V, E)



(a) Please use Prim's algorithm step by step(**Assume that the source is E**).

(b) Please choose the most correct statement for Prim's algorithm.

(I) this is not a greedy algorithm

(II) its greedy rules are starting with any root node and it always forms a single tree T and by adding the edge connected to tree T with the smallest weight

(III) its greedy rules are adding edges in increasing weight and skipping those whose addition would create a cycle

(IV) If adjacency matrix is used, its running time is O(V^2)

(V) If adjacency list and binary heap is used, its running time could be O($(V+E)lgV$)