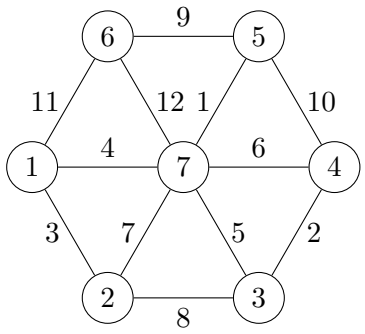
COMPSCI330 Design and Analysis of Algorithms
Assignment 6

Due Date: Thursday March 21, 2019

Guidelines

- **Describing Algorithms** If you are asked to provide an algorithm, you should clearly define each step of the procedure, establish its correctness, and then analyze its overall running time. There is no need to write pseudo-code; an unambiguous description of your algorithm in plain text will suffice. If the running time of your algorithm is worse than the suggested running time, you might receive partial credits.
- **Typesetting and Submission** Please submit the problems to GradeScope. You will be asked to **label your solution for individual problems**. Failing to label your solution can cost you 5% of the total points (3 points out of 60 for this homework).
- LaTeX is preferred, but answers typed with other software and converted to pdf is also accepted. Please make sure you submit to the correct problem, and your file can be opened by standard pdf reader. **Handwritten answers or pdf files that cannot be opened will not be graded.**
- **Timing** Please start early. The problems are difficult and they can take hours to solve. The time you spend on finding the proof can be much longer than the time to write. If you need more time for your homework please use this form and submit a STINF.
- **Collaboration Policy** Please check this page for the collaboration policy. You are **not** allowed to discuss homework problems in groups of more than 3 students. **Failure to adhere to these guidelines will be promptly reported to the relevant authority without exception.**

Problem 1 (Prim and Kruskal)**.** (20 points) We will run Prim's algorithm and Kruskal's algorithm for the following graph:



In this graph, the numbers in circles are indices for vertices, and the numbers next to edges are their lengths.
(a) (5 points) Suppose we run Prim's algorithm with starting vertex 7, list the edges added by Prim's algorithm in the order that they are added by the algorithm.
(b) (5 points) Now we will run Kruskal's algorithm, list the edges added by Kruskal's algorithm in the order that they are added by the algorithm.
(c) (5 points) Construct a graph with 3 vertices that has exactly two minimum spanning trees.
(d) (5 points) Construct a graph that has two edges of the same length, but has a unique minimum spanning tree. Your graph can have at most 4 vertices.

Problem 2 (Edge of a MST)**.** (20 points) In this problem, we are given a weighted undirected graph G (you can assume G is connected) and an edge (u, v). Your goals is to decide whether (u, v) is an edge of some Minimum Spanning Tree of the graph G.
(a) (8 points) Show that if an edge (u, v) is not the minimum cost edge for any cut $(S, \bar{S})$, then (u, v) cannot belong to any minimum spanning tree.
(b) (12 points) Design an algorithm that decides whether the edge (u, v) belongs to some minimum spanning tree of G. That is, the graph G may have multiple minimum spanning trees. If any of them contains the edge (u, v) the algorithm should return YES; otherwise the algorithm should return NO. Your algorithm should run in $O(m)$ time.
(Hint: 1. Notice that the required running time is even smaller than the time it takes to find a MST. 2. Based on part (a) and the Key Lemma, your algorithm should try to find a cut $(S, \bar{S})$ where edge (u, v) is the minimum cost edge of this cut.)

Problem 3 (Independent Set for Bipartite Graph)**.** (20 points) Remember an independent set of a graph is a subset of vertices that are not connected by any edges. When we were doing dynamic programming we looked at an algorithm for computing the maximum independent set of a tree. In this problem we are going to compute the maximum independent set of a bipartite graph.

Given a bipartite graph $G = (U, V, E)$, where both parts U and V have n vertices. Let M be a maximum matching of U and $k = |M|$ be the size of the maximum matching.
(a) (5 points) Prove that the maximum independent set of graph G cannot be larger than $2n - k$.
(b) (5 points) Consider the following algorithm

```
Input: Graph G = (U,V,E), maximum matching M
Output: Maximum Independent Set
```

```
Initially all the vertices are not colored
Color all vertices outside of the maximum matching RED
WHILE there is a vertex u that has not been colored, and is adjacent to a RED vertex
   Find such a vertex u (has not been colored, and is adjacent to a RED vertex)
   Color u BLUE
   Color the vertex matched to u RED
END WHILE
For the uncolored vertices, color the vertices in U RED, and the vertices in V BLUE.
RETURN set of RED vertices.
```

Show that this algorithm always outputs a set of size $2n - k$.

(c) (10 points) Show that the algorithm in part (b) always returns an independent set (and therefore combining (a,b,c) we know the algorithm always returns the maximum independent set).

Hint: Try to show that if the algorithm does not return an independent set (i.e., there is an edge between two RED vertices), then there must be an augmenting path with respect to M in graph G. This is impossible because M is a maximum matching of G.