Department of Electronics Engineering
National Chiao Tung University
Algorithms, Fall 2017

October 19, 2017
Iris Hui-Ru Jiang

DEE3504:
Homework #2 (due 11:59AM, November 9, 2017)

Instructions: Submit your solution to TAs at ED413. No late submission allowed. Please list your collaborator and/or references (if any) for each problem.

Reading:

1. Chapters 3 and 4 including solved exercises.

Exercises:

1. (20 pt) Give an algorithm to detect whether a given undirected graph contains a cycle. If the graph contains a cycle, then your algorithm should output one. (It should not output all cycles in the graph, just one of them.) The running time of your algorithm should be $O(m+n)$ for a graph with n nodes and m edges.

2. (20 pt) We have a connected graph $G=(V, E)$, and a specific vertex $u \in V$. Suppose we compute a depth-first search tree rooted at u, and obtain a tree T that includes all nodes of G. Suppose we then compute a breadth-first search tree rooted at u, and obtain the same tree T. Prove that $G=T$. (In other words, if T is both a depth-first search tree and a breadth-first search tree at u, then G cannot obtain any edges that do not belong to T.)

3. (15 pt) Design an $O(n \lg n)$ algorithm to compute the depth d for the interval coloring. Given a set of requests $\{1, 2, \ldots, n\}$, i^{th} request corresponds an interval $[s(i), f(i))$, where start time $s(i)$ and finish time $f(i)$. The depth d of these given intervals is the maximum number of intervals that pass over any single point on the time-line.

4. (25 pt) We want to execute n jobs on a single machine. Job i has a weight w_i and an execution time t_i, for all $1 \leq i \leq n$. All of these $2n$ numbers are known in advance. Design a scheduling algorithm that minimizes the total weighted waiting time T, where
$$T = \sum_{i=1}^{n} w_i \times (\text{the overall waiting time for job } i).$$
The overall waiting time for job i means the interval from the starting time of the whole schedule until the time when job i finished. Justify the correctness of your algorithm.

5. (20 pt) One of the basic motivations behind the Minimum Spanning Tree Problem is the goal of designing a spanning network for a set of nodes with minimum *total* cost. Here we explore another type of objective: designing a spanning network for which the *most expensive* edge is as cheap as possible.
Specifically, let $G=(V, E)$ be a connected graph with n vertices, m edges, and positive edge costs that you may assume are all distinct. Let $T=(V, E')$ be a spanning tree of G; we define the *bottleneck edge* of T to be the edge of T with the greatest cost.
A spanning tree T of G is a *minimum-bottleneck spanning tree* if there is no spanning tree T' of G with a cheaper bottleneck edge.

(a) Is every minimum-bottleneck tree of G a minimum spanning tree of G? Prove or give a counterexample.
(b) Is every minimum spanning tree of G a minimum-bottleneck tree of G? Prove or give a counterexample.