Homework 1. Due May 7

CS180: Algorithms and Complexity
Spring 2018

Guidelines:

- Upload your assignments to Gradescope by 5:59 PM.
- Follow the instructions mentioned on the course webpage for uploading to Gradescope very carefully (including starting each problem on a new page and matching the pages with the assignments); this makes it easy and smooth for everyone. As the guidelines are simple enough, bad uploads will not be graded.
- You may use results proved in class without proofs as long as you state them clearly.
- Most importantly, make sure you adhere to the policies for academic honesty set out on the course webpage. The policies will be enforced strictly. Homework is a stepping stone for exams; keep in mind that reasonable partial credit will be awarded and trying the problems will help you a lot for the exams.

1. Give an algorithm based on BFS that given a graph $G = (V, E)$ (in adjacency list representation) checks whether or not G has a cycle. For full-credit, your algorithm should run in time $O(|V| + |E|)$. Prove that your algorithm works (you can use properties of BFS that we stated in class without further proving them). [.75 points]

2. Problem 4.1 from here (Chapter 4 of [DPV]). For part (a), you don't have to show how the d-values are computed, just show the values after each iteration in a table with columns indicating the vertices and rows denoting the iterations. For part (b), it suffices to draw the final shortest path tree. [.75 points]

3. Consider the interval scheduling problem we studied in class: Given a sequence of requests with start and finish times $(s(i), t(i))$, $i = 1, \ldots, n$, find a set of non-conflicting jobs of maximum possible size. Show that the following algorithm solves the problem correctly (i.e., returns a set of non-conflicting jobs of maximum size).

   Latest Start Time (LST):

   (a) Set $R \leftarrow \{1, \ldots, n\}$, and $A \leftarrow \emptyset$.

   (b) While $R \neq \emptyset$:

      i. Pick request $i \in R$ with the latest start time.

      ii. Add i to A.

iii. Remove all requests that conflict with i (including i) from R.

(c) RETURN A.

The goal of the problem is to give you some practice in analyzing a greedy algorithm and to better understand the analysis of EFF that we did in class. So for full-credit, you cannot assume the analysis of 'Earliest Finish First' (you can use the ideas though) and your answer must be comparable in detail to our analysis of EFF in class. [.75 points]

(Hint: You can follow the same approach we used for analyzing EARLIEST FINISH FIRST in class.)

4* Given an undirected graph $G = (V, E)$, a subset of vertices $I \subseteq V$ is an independent set in G if no two vertices in I are adjacent to each other. Let $\alpha(G) = \max\{|I| : I \text{ an independent set in } G\}$. The goal of the following questions is to give an efficient algorithm for computing an independent set of maximum size in a tree. Recall that a *leaf* in a graph is a vertex of degree at most 1 and that every acyclic graph (graph without any cycles) has at least one leaf.

Let $T = (V, E)$ be an acyclic graph on n vertices.

(a) Prove that if u is a leaf in T, then there is a maximum-size independent set in T which contains u. That is, for every leaf u, there is an independent set I such that $u \in I$ and $|I| = \alpha(T)$. [.3 points]

(b) Give the graph T as input (in adjacency-list representation), give an algorithm to compute an independent-set of maximum size, $\alpha(T)$, in T. To get full credit your algorithm should run in time $O(|V| \cdot |E|)$ (or better) and you must prove correctness of your algorithm. You don't need to analyze the time-complexity of your algorithm and it is sufficient to solve this part assuming part (1) (if you want) even if you don't solve it. [.45 points]

(Hint: You can try a greedy approach where you add vertices one after the other based on property (1).)

ADDITIONAL PROBLEMS. DO NOT turn in answers for the following problems - they are meant for your curiosity and understanding.

1. Given an undirected graph $G = (V, E)$ as input, use DFS to check if G has a cycle and if does output the cycle. This can be solved in time $O(|V| + |E|)$. (Note that unlike detecting a cycle, finding a cycle when one exists is much harder using BFS.)

2. Problems 4.8, 4.9, 4.18 from [DPV].

3. Problem 4.4, 4.7, 4.13, 4.17 from textbook [KT].