Homework 7

Due Wednesday, March 16th at 11:59 pm ET on Gradescope

Solution guidelines For problems that require you to provide an algorithm, you must give the following:

1. a precise description of the algorithm in English and pseudocode (*),
2. a proof of correctness,
3. an analysis of the asymptotic running time and space.

You may use algorithms from class as subroutines. You may also use any facts that we proved in class, e.g. correctness of subroutines, running time of subroutines.

You should be as clear and concise as possible in your write-up of solutions.

A simple, direct analysis is worth more points than a convoluted one, both because it is simpler and less prone to error and because it is easier to read and understand.

(*) It is fine if the English description concentrates on the high level ideas and doesn't include all the details. But the reader should not have to figure out your solution solely based on the pseudocode. You can also add comments to your pseudocode, in fact that is best practice. FYI we will share a document on good pseudocode style with you and it will also be discussed in labs and lecture.

Problem 1. *Heaps.*

Design an algorithm that, given a pointer to the root of a binary min-heap and a number t, returns a list of all items in the heap that have priority less than t.

Your algorithm should run in time $O(k_t)$, where k_t is the number of items with priority less than t (in particular, the running time should not depend directly on the total number of items in the min-heap).

Hint: Think of the min-heap as a graph. How many nodes in the min-heap can have edges to an item with priority less than t, as a function of k_t?

Problem 2. *Best Paths.*

Although we typically speak of "the" shortest path between two nodes, a single graph could contain several minimum-length paths with the same endpoints. Even for weighted graphs, it is often desirable to choose a minimum-weight path with the fewest edges; call this a *best path* from s to t. Suppose we are given a directed graph G with positive edge weights and a source vertex s in G. Describe and analyze an algorithm to compute best paths in G from s to every other vertex.

Hint: use the number of edges to break ties.

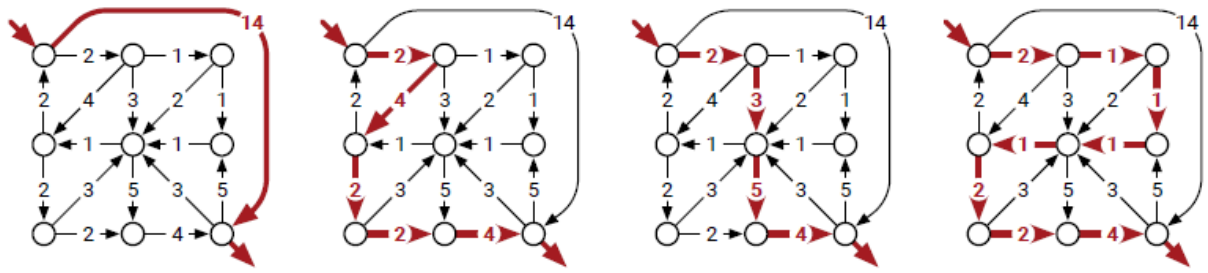

Figure 1: Four (of many) equal-length shortest paths. The first path is the "best" shortest path.