Weekly Assignment 13

December 11, 2019

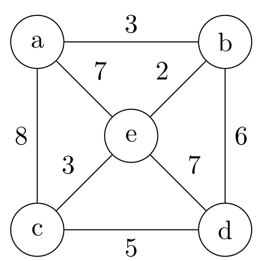
Exercise 1. *Weight: 20%*

Suppose that $G = (V, E)$ a connected graph and $T = (V, A)$ is a spanning tree of G. Prove the following statements or give a counterexample:

1. If T is a MST then at least one edge of minimum weight of G is in T.
2. If at least one edge of minimum weight of G is in T then T is a MST.

Exercise 2. *Weight: 20%*

1. Run Prim's algorithm on the following graph:



   For each iteration, indicate: the vertex added to the spanning tree, the priority queue for the remaining vertices and, for each vertex, predecessor in the spanning tree and key.

2. Run Kruskal's algorithm on the same graph. For each iteration, indicate the vertex and edge added to the spanning tree.
3. Show how one can use Prim's algorithm to sort an array of size n.

Exercise 3. *Weight: 20%*

Consider a situation in which there are n jobs with starting times $s_1, s_2, \ldots s_n$ and finish times $f_1, f_2, \ldots, f_n$, respectively. We say that two jobs are *compatible* if they don't overlap. Our goal is to find a maximum subset of mutually compatible jobs. In the first set of slides on Brightspace (04GreedyAlgorithmsI) a greedy algorithm is presented (not discussed during the lecture) that solves this problem by selecting repeatedly the job with the earliest finish time that is compatible with the previously selected jobs. Now consider an alternative greedy algoritm that repeatedly selects the job with the latest starting time that is compatible with the previously selected jobs. Prove that this alternative algorithm also solves our problem, that is, computes a maximum subset of mutually compatible jobs.

Exercise 4. *Weight: 20%*

Suppose we are given a set of $\{x_1, x_2, ..., x_n\}$ of n real numbers. Describe an efficient algorithm that determines a minimal set of closed intervals of length 1 that contain all of the given points. For instance, the set $\{\pi, 1.8, 0.4, 1.5\}$ can be covered by the intervals $[0, 1]$, $[1, 2]$ and $[\pi - 0.5, \pi + 0.5]$. This set is minimal because clearly it is impossible to cover all the numbers by just two unit length intervals. Show that your algorithm is correct and discuss its time complexity.

Exercise 5. *Weight: 20%*

We say that a sequence S' is a subsequence of S if there is a way to delete zero or more elements from S so that the remaining elements, in order, are equal to the sequence S'. Give a greedy algorithm that takes two sequences – S' of length m and S of length n, each possibly containing an element more than once – and decides in time $O(m + n)$ whether S' is a subsequence of S. Explain why your algorithm is correct.