| Introduction to Algorithms | Problem Set 2 |
|---|---|
| CS 4820, Spring 2018 | Due 11:59pm Thursday, February 8 |

**Hand in your solutions electronically using CMS. Each solution should be submitted as a separate file. Collaboration is encouraged while solving the problems, but:**

1. **list the names of those with whom you collaborated;**
2. **you must write up the solutions in your own words;**
3. **you must write your own code.**

**Remember that when a problem asks you to design an algorithm, you must also prove the algorithm's correctness and analyze its running time. The running time must be bounded by a polynomial function of the input size.**

**(1)** In class, we have seen several examples of optimization problems. These problems come with a notion of what constitutes a solution, and they come with an objective function that measures the quality of a solution. The goal is to find a solution that minimizes (or maximizes) the objective function. For example, in the minimum spanning tree problem, solutions are spanning trees of the given graph and the objective function is the total edge weight of a spanning tree. In the scheduling problem to minimize lateness, solutions are valid schedules for the given requests and the objective function is the maximum lateness of a request.

This way of framing the task of algorithm design — that problems have predefined, mathematically precise objectives, and that the the algorithm designer's job is to solve those predefined problems — is convenient from the standpoint of teaching lectures on algorithms or assigning exercises, but it isn't always an accurate reflection of how algorithm design takes place in real life. In many cases, there is more than one objective function that may be deemed appropriate for the problem at hand. For example for spanning trees, an alternative objective function could be the maximum weight of an edge of the tree (instead of the sum of the weights), whereas for valid schedules, we could consider the sum of the latenesses of requests (instead of the maximum lateness).

*What is the right objective function?* The choice of the objective function is often guided (and sometimes misguided) by applications and heuristic reasoning. However, it is important to take algorithmic considerations into account. For some objective functions the problem may be hard to solve, whereas for other objective functions, there may be efficient algorithms to solve the problem. In such cases, it often happens that the precise formulation of the objective is dictated by the choice of algorithm and not the other way around.

The following problems study the interaction between algorithms and objective functions.

**(1.a)** *(5 points)*
Consider the scheduling problem in Section 4.2 of the textbook. Suppose the goal is to minimize the sum of the latenesses of requests. Show that for this objective function, the earliest-deadline-first algorithm does not always find an optimal schedule.

**(1.b)** *(10 points)*
Again consider the scheduling problem in Section 4.2 of the textbook. Suppose every request has a positive weight $w_i$ and the goal is to minimize the weighted sum of latenesses, $\sum_i w_i \ell_i$.

Give an efficient algorithm for the special case that all deadlines are equal to the time the resource becomes available (i.e., $d_i = s$ for all $i \in \{1, \ldots, n\}$).

**(2)** *(10 points)* Suppose we are given a tree, $T$, and a set of paths in $T$. Two paths are called *compatible* if they have no vertices in common. Design an algorithm to select a maximum cardinality subset of the paths, such that every two paths in the subset are compatible.