Introduction to approximation and randomized algorithms

4th exercise session 26th November 2019

Exercise 1. Consider the standard knapsack problem. That is we have n items with weights w_i and costs c_i and knapsack of size B into which we want to put items maximizing the cost.

a) Consider a greedy algorithm that adds items from the highest cost until they fit. Why isn't this algorithm good?
b) Try to consider a better algorithm. We first sort the items by their density (that is the ratio c_i/w_i) and greedily choose items from the highest density until they fit. Show that this algorithm also isn't good.
c) Try to make a minor modification to the previous algorithm such that we get a 2-approximation.

Exercise 2. We'll look at the k-center problem now. In this problem, we are given a metric space (V,d) with n points and we want to choose k centers (that is k points from V), such that the maximal distance between a point and its closest center would be minimized. So we are minimizing a function $\max_{v \in V} d(p,S)$, where S is the set of chosen centers and $d(v,S)$ is a distance between v and its closest point in S, that is $d(v,S) = \min_{s \in S} d(v,s)$. Find a greedy 2-approximation algorithm for this problem. It could be useful at the analysis of this algorithm to consider that we have an optimal solution of this problem and we are comparing it with the solution found by our algorithm.

Exercise 3. During the lecture you've seen a problem of scheduling on identical machines. Now we'll look at a similar problem and that is scheduling on identical machines with dependencies. On input we get n tasks with lengths p_i, m machines and a graph of dependencies, where there is an edge from task i to task j if and only if task i has to be completed before starting the task j. The goal is to find a schedule of minimum length which respects the

given dependencies. The graph of dependencies has to be acyclic, otherwise no such schedule would exist.

1. Show that we can lower bound the optimum by length of any chain. By chain we mean an arbitrary directed path in the graph of dependencies, where the length of a chain is a sum of task lengths which are in the chain.

2. Construct a greedy 2-approximation algorithm.