Albert-Ludwigs-Universität, Inst. für Informatik
Prof. Dr. Fabian Kuhn
H. Ghodselahi, Y. Maus

November 10, 2016

Algorithm Theory, Winter Term 2016/17
Problem Set 3

hand in (hard copy or electronically) by 09:55, Thursday November 24, 2016, tutorial session will be on November 28, 2016

Exercise 1: Fractional Knapsack problem (16 points)

In the fractional Knapsack problem a thief robbing a store finds n items. The i-th item is worth v_i euro and weighs w_i kilograms, where v_i and w_i are integers. The thief wants to take a load which is as valuable as possible, but he can carry at most W kilograms in his knapsack, for some integer W. However, the thief can take fractions of items, rather than having to make a binary (0-1) choice for each item.

Consider the following greedy algorithm to solve the fractional problem: we first compute the value per weight v_i/w_i for each item. The thief begins by taking as much as possible of the item with the greatest value per weight. If the supply of that item is exhausted and he can still carry more, he takes as much as possible of the item with the next greatest value per weight, and so forth, until he reaches his weight limit W. Hence, by sorting the items by value per weight, the greedy algorithm runs in $O(n \log n)$.

a) **(6 points)** Show that the aforementioned greedy algorithm computes an optimal solution for the fractional Knapsack problem.

b) **(10 points)** Now, the goal is to improve the runtime to solve the fractional Knapsack problem from $\mathcal{O}(n \log n)$ to $\mathcal{O}(n)$.

   The *median of medians* algorithm[1] returns the median of a given array in time $\mathcal{O}(n)$. Use the *median of medians* algorithm as a black box to devise an algorithm that returns an optimal solution for the fractional Knapsack problem with linear running time (in n).

Exercise 2: Matroids (14 points)

We have defined matroids in the lecture. For a matroid (E, I), a maximal independent set $S \in I$ is an independent set that cannot be extended. Thus, for every element $e \in E \setminus S$, the set $S \cup \{e\} \notin I$.

a) **(4 points)** Show that all maximal independent sets of a matroid (E, I) have the same size. (This size is called the rank of a matroid.)

b) **(5 points)** Consider the following greedy algorithm: the algorithm starts with an empty independent set $S = \emptyset$. Then, in each step the algorithm extends S by the minimum weight element $e \in E \setminus S$ such that $S \cup e \in I$, until S is a maximal independent set. Show that the algorithm computes a maximal independent set of minimum weight.

c) **(5 points)** Let E be any finite subset of the natural numbers $\mathbb{N}$ and $k \in \mathbb{N}$ be any natural number. Define a collection of sets $I := \{X \subseteq E : \forall x \neq y \in X,\ x \not\equiv y \pmod{k}\}$. Show that (E, I) is a matroid.

[1] If you are interested in the *median of the medians* algorithm you can find more information about it online. However, as mentioned, you can use it as a blackbox in this exercise.

Exercise 3: Road Trip Planning (10 points)

You want to plan your next road trip with your girlfriend/boyfriend. Because your girlfriend/boyfriend is very picky you can only stay at very nice hotels. Along the way, there are n very nice hotels, at kilometer posts $a_1 < a_2 < \ldots < a_n$, where each a_i is measured from the starting point. However, you do not need to stop at all hotels but you need to stop at a_n, which is your final destination.

To actually be able to do some sightseeing on the way and to reach your destination on time you would like to travel approximately 200 kilometers a day. Due to the distances between the hotels this is not always possible and if you drive x kilometers in one day the potential for being unsatisfied is $(200 - x)^2$.

You want to plan your trip such that the total potential for being unsatisfied is minimized, i.e., the sum over all travel days, of the daily potentials.

Devise an efficient algorithm to solve the problem. Besides a description of your algorithm provide pseudocode for your algorithm. Do not forget to state and explain its runtime.