Algorithms (2IL15): Homework Set A.1

Exercises for Lecture 1

A.1-1 (1 point) Consider the following recurrence (which is similar to the recurrence for *TSP_BruteForce1* in the Handout for Lecture 1) for integers $n \geqslant 1$ and $m \geqslant 0$:

$$T(n,m) = \begin{cases} n & \text{if } m = 0 \\ m + m \cdot T(n+1, m-1) & \text{if } m > 0 \end{cases}$$

Prove using induction that $T(n,m) = O((n+m)m!)$, where we define $m! = 1$ for $m = 0$.

A.1-2 (1 point) Suppose that the distances for a given TSP instance satisfy the triangle inequality, meaning that for any three cities i, j, k we have $\mathit{distance}[i,k] \leqslant \mathit{distance}[i,j] + \mathit{distance}[j,k]$. Show how to make the pruning test in *TSP_Backtrack* stronger (so that it will prune more) by taking into account that the tour must eventually go back to city 1. Explain why you need the triangle inequality.

A.1-3 (1 point) Let S be a set of n cities, and let L be a number. As in the section on TSP in the Handout, we have a distance table $\mathit{distance}[1..n, 1..n]$. We want to determine if there is a subset $S' \subseteq S$ such that a tour exists on the cities in S' whose length is exactly L. (Note that the tour does not have to be the shortest tour on S'.) Give pseudocode for an algorithm that solves this problem.
Hint: Adapt algorithm *TSP_BruteForce2* from the Handout.

A.1-4 ($1 + \frac{1}{2} + \frac{1}{2}$ point) Let $S = \{x_1, \ldots, x_n\}$ be a set of n integers, and let B be another integer. The numbers from S are stored in an array $S[1..n]$, so $S[i] = x_i$ for all $1 \leqslant i \leqslant n$.

(i) Give pseudocode for a recursive backtracking algorithm that decides if there is a subset $S' \subseteq S$ such that $\sum_{x \in S'} x = B$. Carefully explain the meaning of the various parameters your algorithm has, and state the exact problem (in terms of these parameters) that the algorithm solves. Your algorithm should not use pruning.

(ii) State (and briefly explain) a recurrence for the running time of your algorithm. (Do not forget the base case.) You do not need to solve the recurrence.

(iii) Now suppose that all numbers in S are positive. Explain how to use pruning to speed up the algorithm, and whether or not the pruning would also work when there are negative numbers in the set S. (It is not sufficient to only prune when a partial solution already has the required value B.)

Exercises for Lecture 2

A.1-5 ($1\frac{1}{2}$ points) Suppose you have to drive from Eindhoven to the south of France. Your start and destination are fixed and the route is fixed as well. You start with a full petrol tank, but since the distance is quite long you will need to fill up your tank several times. There are n petrol stations along the way, and your goal is to stop at as few petrol stations as possible.

A natural greedy approach would be to make your first stop at the furthest petrol station, s^*, that is reachable from Eindhoven, then fill up your tank, drive again to the furthest reachable petrol station, and so on, until you can reach your destination. Prove or disprove that this is a good greedy choice. In other words, prove or disprove that there is an optimal solution for this problem that includes a stop at petrol station s^*.

NB: If you give a proof, it must follow the general proof structure for the greedy-choice property as given on the slides for Lecture 2. If you disprove the statement, you must give (and explain) an explicit counter-example.

A.1-6 (1 point) Suppose you are living in a country with coins of 1 cent, 4 cent, 9 cent, and 15 cent. Given an integer value $V \geqslant 1$, we want to select a minimum number of coins whose combined value is V cents. A possible greedy algorithm for this problem is as follows: Select the coin with the largest value v such that $v \leqslant V$ and recursively select coins whose combined value is $V - v$. The correctness of this algorithm would be based on the following lemma.

Lemma 1 *Let $v \in \{1, 4, 9, 15\}$ be the largest value such that $v \leqslant V$. Then there is an optimal solution to the Coin-Selection problem that uses a coin of value v.*

Prove or disprove the lemma.

NB: If you give a proof, it must follow the general proof structure for the greedy-choice property as given on the slides for Lecture 2. If you disprove the statement, you must give (and explain) an explicit counter-example.

A.1-7 ($1 + 1 + \frac{1}{2}$ point) Let $E_1, \ldots, E_n$ be a set of n exercises, each taking 1 day to solve, and suppose we have n days available to solve the exercises. Each exercise E_i has a deadline $D[i]$ associated to it, which is an integer in the range $1..n$. Each exercise E_i must either be solved on a day d with $d \leqslant D[i]$ (so that it meets its deadline) or it should not be solved at all. When we solve exercise E_i, we receive $P[i]$ points. Our goal is to compute a schedule for solving exercises such that the total number of points we score is maximized.

(i) Give pseudocode for a greedy algorithm that computes an optimal schedule for solving the exercises. Your algorithm should have the arrays $D[1..n]$ and $P[1..n]$ as input; its output should be an array $S[1..n]$ such that $S[d] = i$ if exercise E_i is scheduled on day d and $S[d] = 0$ if no exercise is scheduled for day d.

(ii) Prove that the greedy choice you have used in your algorithm is correct. Your proof must follow the general proof structure for the greedy-choice property as given on the slides for Lecture 2.

(iii) Analyze the running time of your algorithm.