Exercise VI, Algorithms 2024-2025

These exercises are for your own benefit. Feel free to collaborate and share your answers with other students. There are many problems on this set, solve as many as you can and ask for help if you get stuck for too long. Problems marked * are more difficult but also more fun :).

For more exercises on dynamic programming and video explanations of solutions, see *http://people.csail.mit.edu/bdean/6.046/dp/*

Solve New Problems Using Dynamic Programming

1 *(25 pts)* **Restaurant placement.** Justin Bieber has surprisingly decided to open a series of restaurants along the highway between Geneva and Bern. The n possible locations are along a straight line, and the distances of these locations from the start of the highway in Geneva are, in kilometers and in arbitrary order, $m_1, m_2, \ldots, m_n$. The constraints are as follows:

- At each location, Justin may open at most one restaurant. The expected profit from opening a restaurant at location i is p_i, where $p_i > 0$ and $i = 1, 2, \ldots, n$.
- Any two restaurants should be at least k kilometers apart, where k is a positive integer.

As Justin is not famous for his algorithmic skills, he needs your help to find an optimal solution, i.e., **design and analyze** an *efficient* algorithm to compute the maximum expected total profit subject to the given constraints.

2 (half *, Problem 15-1) **Longest simple path in a directed graph.**

Suppose that we are given a directed acyclic graph $G = (V, E)$ with real valued edge weights and two distinguished vertices s and t. Describe a dynamic programming approach for finding a longest weighted simple path from s to t. What does the subproblem graph look like? What is the efficiency of your algorithm?

3 (*) **Knapsack Problem** Suppose that you are going on a beautiful hike in the Swiss Alps. As always, you are faced with the following problem: your knapsack is too small to fit all the items that you wish to bring with you. As items are of different importances and have different sizes, you would like to maximize the total value of the items that you can bring with you. Formally, we can define the "packing" problem as follows:

INPUT: A knapsack of capacity C and n items where item $i = 1, 2, \ldots, n$ has value $v_i \geq 0$ and size $s_i > 0$.

OUTPUT: A subset of items S that maximizes $\sum_{i \in S} v_i$ (the total value) subject to $\sum_{i \in S} s_i \leq C$ (the total size of the packed items is at most the capacity).

3a (The Fractional Knapsack Problem) Suppose that your items are divisible, i.e., you can pack a fraction $f \in [0, 1]$ of an item i and in that case it will give you a profit of $f \cdot v_i$ and occupy a space of $f \cdot s_i$ in the knapsack. Give a greedy algorithm for this case that runs in time $O(n \log n)$.

3b Show that the greedy algorithm fails when the items are indivisible. How bad can the profit of the solution returned by the greedy algorithm be compared to an optimal solution?

3c Design a dynamic programming algorithm to solve the Knapsack Problem. Your algorithm should run in time $O(nC)$.