CS300 Homework #7

Yeonghun Kim
neutrinoant@kaist.ac.kr

Total 100 points Due: 2020-06-04 23:59:00 KST

- Write in English or Korean.
- Make sure your writing is readable before upload it (especially for scanned image).

For problems 1, 2, and 3-(c), your answer should contain (1) the definition of subproblem, (2) recurrence relation and why it works, (3) base condition, (4) algorithm (pseudo code also allowed), and (5) analysis of worst case time complexity with O-notation.

1. Cutting Sticks (30 pts)

You have a thick wooden stick with length $L > 0$ and you wish to cut n different positions on it. However WCC(Wood Cutting Company), one of the famous company in this task, charges a cost for each cut of the stick according to its length, in other words, the cost is k for cutting once any position on a stick with length k.
Assuming that the left end and right end of the stick you have are at positions 0 and L respectively, let p_i the positions you wish to cut satisfying $0 < p_1 < \ldots < p_n < L$. Assume that you can cut only one position at a time, that is, you cannot cut multiple positions at the same time. Give an $O(n^3)$ algorithm using dynamic programming to find minimal total cost for cutting all of the positions p_i given.

2. Particle on Energy Field (30 pts)

A small particle in 3-dimensional space is located at origin. It can move only by $+1$ on one axis at a time, that is, its position will be changed from (x, y, z) to $(x+1, y, z)$ or $(x, y+1, z)$ or $(x, y, z+1)$ after one movement. Also there is an energy field in the space, which is given by a real-valued function $E = f(x, y, z)$, so whenever the particle moves it will get the energy corresponding to the location it arrived, including its starting point.
Given an energy function $f(x, y, z)$ having positive value for all $x, y, z \in \mathbb{R}$ and for positive integer n, if the particle moves from $(0, 0, 0)$ to (n, n, n), we want to find a maximal total energy the particle can get during its journey. Give an $O(n^3)$ algorithm for this task using dynamic programming.

3. Optimal Binary Search Tree (40 pts)

Given a set of words and their corresponding probabilities, optimal binary search tree (Optimal BST) is a binary search tree with each word as a key and which provides minimal expected search cost. In this case, the cost is defined as a sum of all (depth of key) $\times$ (its corresponding word probability). This definition is just a brief explanation of that in our class lecture slides.

(a) (10 pts) Optimal BST might not have highest-probability word at root. Give an example for this fact. Your answer should contain (1) a set of words and their probabilities, and (2) its Optimal BST (you don't need to prove optimality).

(b) (10 pts) Optimal BST might not have smallest height. Give an example for this fact. Your answer should contain (1) a set of words and their probabilities, and (2) its Optimal BST (you don't need to prove optimality).

(c) (20 pts) It is proved by Knuth that there are always roots of optimal subtrees such that $root[i, j-1] \leq root[i, j] \leq root[i+1, j]$ for all $1 \leq i < j \leq n$, where $root[i, j]$ is the root of an optimal binary search tree containing keys $k_i, \ldots, k_j$. Using this fact, give an $O(n^2)$ algorithm to find Optimal BST for a given set of word probabilities p_i. (hint: modify some of lines of algorithm in lecture slides)