Algorithms M2–Homework 2

November 2, 2021

1 Recurrence Relations

For the following recurrence relations give an estimate (using O-notation) of the asymptotic complexity of each function. Give a short justification for your answers (proof by induction or using Master Theorem).

1. $T_1(n) = 27T_1(n/3) + n^2$
2. $T_2(n) = T_2(n/2) + n$
3. $T_3(n) = T_3(n/2) + \log n$
4. $T_4(n) = 2^n T_4(n-1)$

2 Tournament Rankings

We set up a round-robin tennis tournament among n players: each player plays each other once, and one of the two players wins. We would like to **rank** the players according to the results of the tournament.

A **perfect ranking** is an ordering of the players $p_1, \ldots, p_n$ such that for all i, j, if $i < j$ then p_i won the game against p_j. Not all tournaments have perfect rankings (this depends on the individual results), so we also define the notion of **reasonable ranking**. A reasonable ranking is an ordering of the players $p_1, \ldots, p_n$ such that for all i, p_i won the game against p_{i+1}. (Note that p_i may have lost the game against, for example p_{i+2}, but the ranking is still considered reasonable).

1. Give an example of a tournament that does not have a perfect ranking and an example of a tournament that does.
2. Prove by induction that for any set of tournament results, a reasonable ranking always exists.
3. Give a polynomial-time algorithm to find a reasonable ranking.

3 Making Change

([DPV] Exercise 6.17) We are given an unlimited supply of coins of values $x_1, x_2, \ldots, x_n$ and a target value v. We are asked if it is possible to produce value v using these coins. For example, if we have coins with values 5 and 10, we can produce the value 15 but not the value 12.

1. Give an example where the greedy algorithm fails for this problem. That is, give some coin values and target v, such that it is possible to produce v using these coin values, but if keep using the largest coin that is smaller than the remaining change, we cannot produce the value v.

2. Given a dynamic programming algorithm that decides if it is possible to produce value v, running in time polynomial in $n + v$.

4 A card game

([DPV] Exercise 6.13) Consider the following game: we are given a sequence of n cards $s_1, \ldots, s_n$, where each card has an integer value. Two players take turns playing. In each turn the current player may pick up either the left-most or the right-most available card. This card is then removed from the game and the player gains the value of the card he selected. In the end the winner is the player whose selected cards have the higher sum.

1. Give an example that shows that it is not optimal to be greedy. That is, an example where if the first player selects the card with higher value (between the two available), he loses.

2. Give a dynamic programming algorithm to determine the optimal strategy for the first player.

5 Two-Dimensional Knapsack

You go to the supermarket. You have with you a bag that can take items of total weight W. You also have with you B euros. The supermarket sells n items and the i-th item has weight w_i and price p_i.

Give a DP algorithm that decides if it is possible to fill your bag completely (that is, with items of total weight W) without spending more than your budget B. You can assume that the supermarket has many copies of the same item, so that if a melon has weight $w_i = 1$ and $p_i = 3$ you are allowed to take three melons for a total weight of 3 and price of 9. Your algorithm should run in time polynomial in n, W, B.

6 Estimate a probability

([DPV] Exercise 6.15) Two teams A,B are repeatedly playing a best-of-n contest, that is, the two teams play repeatedly and the first team to win n teams is the champion. The two teams are of equal strength, so each team wins each match independently with probability 50%. Suppose that team A has so far won i matches and team B has won j, where $i, j < n$. Give an algorithm which, with input i, j, n calculates the probability that A wins overall. For example, if $i = n - 1, j = n - 3$ your algorithm should output 7/8.

Generalize your algorithm so that it outputs the probability that A wins overall assuming that the probability that A wins a specific match is $p_A \neq 1/2$.

7 Bin Packing

We are given n items with weights $w_1, \ldots, w_n$. We have three bags, each of which can take items of maximum total weight W. Is it possible to fit all the items into the three bags? Give an algorithm that decides this in time $O(nW^3)$.