CS 577: Introduction to Algorithms 10/22/2015

Homework 6

Instructor: Dieter van Melkebeek TAs: Kevin Kowalski, Andrew Morgan, Bryce Sandlund

This assignment covers dynamic programming. Good luck!

Review problems

1. You are given an array $A[1 \ldots n]$ of integers and want to find the maximum sum of the elements of (a) any subsequence, and (b) any subarray. Recall that a subsequence of A is obtained from A by deleting any number of positions; a subarray is a subsequence in which the positions that are not deleted form an interval.

   Give linear-time algorithms for both problems. What if the goal is to find out whether a given integer value v can be realized as a sum of the form (a) or (b)?

2. When you were little, every day on your way home from school you passed the house of your aunt. When you stopped by for a chat on day i, your aunt would give you a number ℓ_i of lollies but also tell you that she wouldn't give you any more lollies for the next k_i days.

   Give an efficient algorithm that takes as input the numbers (ℓ_i, k_i) for $i \in \{1, 2, \ldots, n\}$, and outputs the maximum number of lollies you can get during those n days.

Graded written problem

3. [10 points] The library has n books that must be stored in alphabetical order on adjustable height shelves. Each book has a height and a thickness. The width of the shelf is fixed at w, and the sum of the thicknesses of books on a single shelf cannot exceed w. The next shelf will be placed on top, at a height equal to the maximum height of a book on the shelf.

   Give an efficient algorithm that minimizes the total height of shelves used to store all the books. You are given the list of books in alphabetical order, $b_i = (h_i, t_i)$, where h_i is the height and t_i is the thickness, and the shelf width w.

Additional written problem

4. Gerrymandering is the practice of carving up electoral districts in very careful ways so as to lead to outcomes that favor a particular political party. Recent court challenges to the practice have argued that through this calculated redistricting, large numbers of voters are being effectively (and intentionally) disenfranchised.

   Computers, it turns out, have been implicated as some of the main "villains" in much of the news coverage on this topic: it is only thanks to powerful software that gerrymandering grew from an activity carried out by a bunch of people with maps, pencil, and paper into the industrial-strength process that it is today. Why is gerrymandering a computational problem? Partly it's the database issues involved in tracking voter demographics down to the level of individual streets and houses; and partly it's the algorithmic issues involved in grouping voters into districts. Let's think a bit about what these latter issues look like.

Suppose we have a set of n precincts P_1, P_2, ..., P_n, each containing m registered voters. We're supposed to divide these precincts into two districts, each consisting of $n/2$ of the precincts. Now, for each precinct, we have information on how many voters are registered to each of two political parties. (Suppose for simplicity that every voter is registered to one of these two.) We'll say that the set of precincts is susceptible to gerrymandering if it is possible to perform the division in such a way that the same party holds a majority in both districts.

Give an algorithm to determine whether a given set of precincts is susceptible to gerrymandering. The running time of your algorithm should be polynomial in n and m.

Example Suppose we have $n = 4$ precincts, and the following information on registered voters. Party A has 55, 43, 60, and 47 voters in districts P_1, P_2, P_3, and P_4, respectively, and party B has 45, 57, 40, and 53. This set of precincts is susceptible, since if we grouped precincts P_1 and P_4 into one district, and precincts P_2 and P_3 into the other, then party A would have a majority in both districts. (Presumably, the "we" who are doing the grouping here are members of party A.) This example is a quick illustration of the basic unfairness in gerrymandering: although party A holds only a slim majority in the overall population (205 to 195), it ends up with a majority in not one but both districts.

Optional programming problem

5. [2.5 points] Solve SPOJ problem Coins Game (problem code MCOINS).

Challenge problem

6. There is a famous joke-riddle for children:

> Three turtles are crawling along a road. One turtle says: "There are two turtles ahead of me." The other turtle says: "There are two turtles behind me." The third turtle says: "There are two turtles ahead of me and two turtles behind me." How could this have happened? The answer is – the third turtle is lying!

In this problem you have n turtles crawling along a road. Some of them are crawling in a group, so that they do not see members of their group neither ahead nor behind them. Each turtle makes a statement of the form: "There are a_i turtles crawling ahead of me and b_i turtles crawling behind me."

Your task is to find the minimal number of turtles that must be lying. More formally, let x_i denote the position along the road of turtle i, $1 \leq i \leq n$. Some turtles may be at the same position. Turtle i tells the truth if and only if a_i is the number of turtles j such that $x_j > x_i$ and b_i is the number of turtles j such that $x_j < x_i$. Otherwise, turtle i is lying.

Give an algorithm that solves the problem in $O(n \log n)$ time.