CS 577: Introduction to Algorithms	10/2/2013

Homework 2

Instructor: Dieter van Melkebeek	Due: 10/16/2013

This assignment covers greed (chapter 4 in the textbook).

Ungraded problems

1. You are given a knapsack with a weight limit of W and n items with nonnegative weights $w_1, w_2, \ldots, w_n$. You want to fill your knapsack as close to the weight limit W as possible but without exceeding it.

   Give an algorithm that solves this problem in the case where the weight of each item is at least as large as the weight of all previous items combined, i.e., $w_i \geq \sum_{j=1}^{i-1} w_j$ for each $1 < i \leq n$. If so, your algorithm should output a list of items to take in your knapsack; otherwise, the algorithm should output "invalid input."

   Your algorithm should take no more than $O(n)$ elementary operations. Arithmetic operations like the addition of two numbers and the comparison of two numbers are considered elementary for this problem.

2. Problem 4.6 in the textbook (p. 191).

3. Problem 4.26 in the textbook (p. 202).

Graded written problems

4. (10 points)

   (a) Consider the following edge exchange operation, which is implicit in the key observation underlying the minimum spanning tree algorithms from class: Let G be a connected weighted graph, T a spanning tree of G, and e^* an edge in G that is not in T. We obtain a new spanning tree T^* as follows: Add e^* to T. This creates a unique cycle in T. Next remove an edge $e \neq e^*$ from this cycle. We now have a new spanning tree T^* of G.

   Show that for any two spanning trees T_1, T_2 of G, there is a sequence of edge exchange operations that transforms T_1 into T_2.

   (b) Suppose you are given an integer k and a connected graph G in which each edge weight is either 0 or 1. Give a polynomial-time algorithm that determines whether G has a spanning tree of cost exactly k.

5. (10 points) The Department of Recreation has decided that it must be more profitable, and it wants to sell advertising space along a popular jogging path at a local park. They have built a number of billboards (special signs for advertisements) along the path and have decided to sell advertising space on these billboards. Billboards are situated evenly along the jogging path, and they are given consecutive integer numbers corresponding to their order along the path. At most one advertisement can be placed on each billboard.

A particular client wishes to purchase advertising space on these billboards but needs guarantees that every jogger will see its advertisement at least k times while running along the path. However, different joggers run along different parts of the path.

Interviews with joggers revealed that each of them has chosen a section of the path which he/she likes to run along every day. Since advertisers care only about billboards seen by joggers, each jogger's personal path can be identified by the sequence of billboards viewed during a run. Taking into account that billboards are numbered consecutively, it is sufficient to record the first and the last billboard numbers seen by each jogger.

Unfortunately, interviews with joggers also showed that some joggers don't run far enough to see k billboards. Some of them are in such bad shape that they get to see only one billboard (here, the first and last billboard numbers for their path will be identical). Since out-of-shape joggers won't get to see k billboards, the client requires that they see an advertisement on every billboard along their section of the path. Although this is not as good as them seeing k advertisements, this is the best that can be done and it's enough to satisfy the client.

In order to reduce advertising costs, the client hires you to figure out how to minimize the number of billboards they need to pay for and, at the same time, satisfy stated requirements. Design an algorithm that produces an optimal selection of billboards given k and the first and last billboard numbers of each of the n joggers.

For full credit, your algorithm should run in time $O(n \log n)$.

6. (10 points) Problem 4.12 in the textbook (p. 193-194).

Programming problem

7. (5 points) Solve UVa problem #1013, *Island Hopping*.

Extra-credit problem

8. In some courses you can choose a certain number k of the n assignments that will be dropped in the calculation of your grade. If all the assignments counted equally, the choice would be easy: simply drop the assignments with the lowest scores. However, each assignment may have a different maximum score. Your final homework grade will be the percentage ratio of your total score to the maximum possible score for the retained assignments. This leads to the following problem. You are given a value of k and a list of n assignment results (s_i, m_i), $1 \leq i \leq n$, where s_i denotes your score on the ith assignment and m_i denotes the maximum possible score on that assignment. Your goal is to find a set $I \subseteq \{1, 2, \ldots, n\}$ with $|I| = k$ such that $\sum_{i \notin I} s_i / \sum_{i \notin I} m_i$ is as large as possible.

   Give an algorithm that runs in time $O(n^2 \log n)$. For starters, aim for an algorithm that runs in time polynomial in the number of bits in the input.