COMP3804 Winter 2006 — Assignment 2 — Due Thursday Mar. 9

This assignment is due in class on Thursday March 9th. If you are not able to come to class then please put your assignment in the COMP3804 assignment box in the CCSS lounge. The assignments will be picked up from this box immediately after class.

1 Weighted Rank Trees

Describe how to augment red-black trees so that they store a collection of pairs (k_i, w_i) under the following operations:

1. INSERT(k, w): insert the pair (k, w) into the tree.
2. DELETE(k): Delete the pair whose first element is k from the tree.
3. SEARCH(k): Find the pair whose first item is k.
4. SEARCH-WEIGHT(w): Find the largest key k such that $\sum_{i:k_i<k} w_i \leq w$.

All operations should run in $O(\log n)$ time, where n is the number of elements in the tree at the time of the operation.

2 Bentley's Problem Finished

Recall Bentley's problem:
Input: An array $A_1, \ldots, A_n$ of real numbers.
Output: two indices i and j such that $\sum_{k=i}^{j} A_k$ is maximum over all $1 \leq i \leq j \leq k$.

We have already given $O(n^3)$ (brute-force), $O(n^2)$ (finesse), and $O(n \log n)$ time (divide-and-conquer) algorithms for Bentley's problem. In this question we will find an $O(n)$ time dynamic programming algorithm. Let $m_j = \max\{\sum_{k=i}^{j} A_k : 1 \leq i \leq j\}$

1. Give a formula for m_{j+1} that uses only $m_1, \ldots, m_j$ and A_{j+1}.
2. Give an $O(n)$ time algorithm to solve Bentley's problem.

3 Longest Increasing Subsequence

Given a sequence $S = \langle x_1, \ldots, x_n \rangle$ of numbers, an *increasing subsequence* is a subsequence $\langle x_{i_1}, \ldots, x_{i_k} \rangle$ such that $i_j < i_{j+1}$ and $x_{i_j} < x_{i_{j+1}}$ for all $1 \leq i < k$. A *longest increasing subsequence* is an increasing subsequence of maximum length (k is as large as possible).

1. Define a directed acyclic graph G with one source s and one sink t and such that there is a bijection between the paths in G from s to t and increasing subsequences in S.

2. Give upper bounds on the number of vertices and edges in G and determine how long it takes to construct G from S.

3. Describe an algorithm for finding the longest-increasing subsequence and analyze its running time.

4 Dynamic Programming for Subset-Sum

Let $S = \{x_1, \ldots, x_n\}$ be a set of non-negative integers and let t be another non-negative integer. The SUBSET-SUM problem is to find a subset $S' \subseteq S$ such that $\sum_{x \in S'} x = t$, or determine that no such subset exists.

Let

$$B_{i,j} = \begin{cases} \text{true} & \text{if there exists a subset of } \{x_1 \ldots, x_i\} \text{ whose sum is } j \\ \text{false} & \text{otherwise} \end{cases}$$

1. Show how to compute $B_{i+1,j}$ given only x_{i+1} and $B_{i',j'}$ for all $1 \leq i' \leq i$ and $1 \leq j' \leq t$.

2. Give the fastest algorithm you can find for solving the SUBSET-SUM problem.