CS 473: Fundamental Algorithms, Fall 2011

Homework 6 (due Monday, 23:55:00, October 17, 2011)

Collaboration Policy & submission guidelines: See homework 1.
Each student individually have to also do **quiz 6** online.

Version: **1.11**

1. (30 PTS.) Encoding sparse binary strings.

   (This question is long but it is easy.)

   One of the most fascinating issues when dealing with storing data using computers is how to store it efficiently. In many cases the information is very sparse, and compressing it is beneficial. Here, we are going to concentrate on a simple compression scheme for binary strings that would also prove some combinatorial identity that would turn out to be interesting later on.

   (A) (5 PTS.) Describe an encoding algorithm that reads a non-negative integer x, and outputs a string of bits of length at most $x + 1$, such that one can compute the value of x from this string. Specifically, describe how to compute the value of x given this string. (Think simple.)

   (B) (5 PTS.) Given k integer non-negative numbers $n_1, \ldots, n_k$, describe an encoding and decoding algorithms for a sequence of k such numbers. Specifically, the encoding algorithm (given such k numbers) should output a binary string of length $\leq k + n_1 + n_2 + \ldots + n_k$ (i.e., this is a bound on the number of bits in the output string), such that the decoding algorithm reading this string can recover $n_1, \ldots, n_k$ (in this order) and output them. Note that k is not part of the input and it is known in advance.

   (C) (5 PTS.) The above scheme is of course not very efficient if the numbers n_i are large. So let Δ be some positive integer. Describe how to encode an integer number i between 0 and Δ (inclusive) using $\leq 1 + \log_2(\Delta + 1)$ bits (you can assume Δ is known in advance).

   (D) (5 PTS.) An integer number x can be encoded as two integers $x' = \lfloor x/\Delta \rfloor$ and $x'' = x - \lfloor x/\Delta \rfloor * \Delta$. Indeed, we have that $x = x' * \Delta + x''$. Given k integer non-negative numbers $n_1, \ldots, n_k$, show how to encode them such that the output binary string is of length $\leq 2k + k \log_2(\Delta + 1) + \sum_{i=1}^{k} \lfloor n_i/\Delta \rfloor$ bits (hint: start with $k = 1$). Describe also the decoding algorithm that outputs the original numbers given this string.

   (E) (5 PTS.) Describe an algorithm that reads in a binary string of length n that has exactly k bits in it that are one (the rest are zero), and encode it as a binary string of length $\leq 4k + k \log_2(n/k)$ using your algorithms from the previous parts. (Hint: Think about the runs in this string; that is, the sequences of consecutive zeros and how to encode them as numbers.) Describe also the decoding algorithm.
   Conclude that any binary string of length n containing exactly k ones, can be written (i.e., encoded) as a binary string of length $\leq 4k + k \log_2(n/k)$.
   (Note, that this bound is way better than the naive bound of $k + k \log_2 n$.)

   (F) (5 PTS.) Prove (using the previous part) that $\binom{n}{k} \leq 2(16n/k)^k$. (This is in no way the best bound one can prove, but this is a surprisingly good estimate.)
   Hint: Think about the connection of $\binom{n}{k}$ to binary strings of length n.

2. (40 PTS.) My Kingdom for a Tree.

let G be a graph with m edges and n vertices with weights on the edges.

(A) (10 PTS.) You are given a minimum spanning tree T of G. The weight of one edge e of the graph had changed (the edge might be an edge of T). Describe an $O(n+m)$ time algorithm that computes the MST of G with the updated weights.

(B) (10 PTS.) You are given a minimum spanning tree T of G. For social reasons that are still not well understood, Vogon children just broke into your house and stole k edges of T. Describe an algorithm to compute an MST for the graph G without these k edges. The running time of your algorithm should be $O(k \log k + m)$.

(C) (10 PTS.) You are given a minimum spanning tree T of G. The weight of k edges of G (that are not in T) had been suddenly decreased. Describe an $O((n+k)\log n)$ time algorithm that computes the MST of the new graph.

(D) (10 PTS.) You are given a graph G and a minimum spanning tree T. The ***max-price*** of a path π is the price of the most expensive edge on π. Describe an algorithm, as efficient as possible, for computing the minimum max-price path between two given vertices x and y of G. (For full credit, your algorithm should work in $O(n)$ time.)
Prove the correctness of your algorithm.

(And no, you can not use hashing in the solution for this question.)

3. (30 PTS.) Increasing subsequence.

You are given two sequences of numbers $x_1, \ldots, x_n$ and $y_1, \ldots, y_m$. Describe an algorithm (as fast as possible) for computing the longest increasing subsequence of numbers that is an increasing subsequence of both sequences. For example, for the sequences

$$30, 50, 10, 70, 40, 20, 60$$
$$10, 20, 30, 40, 50, 60, 70$$

The longest increasing subsequence common to both sequences is $10, 40, 60$. (There are other possible solutions in this specific example.)