Midterm Exam #2 - Data Structures and Algorithms
Computer Science 308-251 A - November 7, 2000
Each question has equal weight.

1. Huffman Coding for Text Compression

(a) A special text on a 10-symbol alphabet is to be stored in binary form by using *Huffman coding*. The symbols have probabilities A=0.02, B=0.03, C=0.04, D=0.06, E=0.07, F=0.08, G=0.09, H=0.11, I=0.20, and J=0.30 of occurring in the text. Construct, by hand, a *Huffman tree* for this alphabet. List the *binary codes* for each of the ten symbols.

(b) How many bits would you expect to save on the average on a text consisting of 1000 random symbols by using Huffman coding instead of a *block* code? What is the resulting *compression ratio*?

2. Heap-Building

Recall that the O(n) time *bottom-up* heap-building algorithm for n keys, has 2 main steps: in step (1) it creates a *balanced binary* tree with the keys placed in the nodes arbitrarily and in step (2) it makes promotions in this tree in a *bottom-up* fashion. Consider *heap-building* in general for an input of 8 distinct positive integers (keys) of your choice. We are interested in analyzing the performance of building a *max-heap* under 2 modifications of the bottom-up heap building algorithm: (1) ***not*** using a balanced binary tree but keeping everything else the same and (2) ***not*** promoting bottom-up but keeping everything else the same.

(a) Devise a tree (not necessarily balanced or binary) for your 8 integers which causes the bottom-up heap-building procedure to perform the *maximum* number of promotions.

(b) If the heap-building algorithm uses a *balanced binary* tree and is modified so that promotions are made *blindly (at random)*, what is the greatest number of promotions that might be made?

(c) For each of the two variants above, give formulas (in “Big Oh” notation) for the number of promotions as a function of n where n is the number of keys used and is arbitrarily large.

3. Planar Point Location in a Polygon

Consider that you are given a simple (non-self crossing) polygon P consisting of n vertices, such that it is *monotonic* in the x-direction. You are also told that no two vertices of P lie on a vertical line. In other words, every vertical line that intersects the polygon intersects its boundary at no more than two points and at no more than one vertex of P. It is desired to preprocess the polygon in time $P(n)$ incorporating a data structure that uses space $S(n)$ such that all future point-inclusion queries (of the form: is point p inside, outside, or on the boundary of P?) can be answered in $Q(n) = O(\log n)$ worst-case time. Invent an algorithm for carrying out the preprocessing in $O(n)$ time if you can (for full marks). $S(n)$ should be $O(n)$. Prove the correctness of the algorithm. Prove the time complexities $P(n)$ and $Q(n)$ and the space complexity $S(n)$.

4. Proving Lower Bounds via Reduction

You are **given** the information that the problem of computing the *maximal vectors* of a set S of n points in the plane (given by their x and y coordinates as real numbers) has complexity $\Omega(n \log n)$ in the worst case in the *real RAM* model of computation. {Recall that a point p in the plane is said to *dominate* another point q if **both** the x and y coordinates of p are greater than or equal to those of q. A point p is a *maximal vector* of S (in the first quadrant sense) if **no** other point of S dominates it.}

However, the problem you **are** interested in is in fact the following *interval-inclusion problem*. Given n intervals $I_i(L_i,R_i)$, i=1,2,...,n, on the *positive* real line (x-axis), report each and every interval that is contained in some other interval. Here L_i and R_i are the x-coordinates of the *left* and *right* end-points of segment I_i respectively. Assume that all end-points are distinct.

Show in detail, by **reduction**, using your knowledge of the complexity of the maximal vector problem, that the *interval-inclusion problem* has complexity $\Omega(n \log n)$ in the worst case in the *real RAM* model of computation.

***Hint*:** First consider mapping each interval $I_i(L_i,R_i)$ to the 2-dimensional point $p_i(x_i,y_i)$ where the x,y coordinates of p_i are: $x_i = -L_i$ and $y_i = +R_i$. Then consider the relationship between an interval that is contained in another and whether its corresponding mapped point is or is not a maximal vector. Use the "black-box" approach. Include a proof of correctness of whatever relationship you discover.