Algorithms Core 15-750 Midterm Spring 1998

DUE: 12Noon Tuesday 17 March 1998, in room 4307 DH

This test is a 24 hour take-home test. Do not discuss this test with anyone but Gary Miller. You may use the three books from the course (CLR, AHU, Kozen), your personal notes, material from the class directory, and class handouts.

If you have any questions or clarifications either phone me or send me email. I will maintain a list of corrections and clarifications on the newsgroup.

Answer each question on separate sheets of paper for grading purposes. Make sure that each page has your name and email address.

Make sure that each algorithm you consider includes the following three bullets.

DESCRIPTION: A high level description of the algorithm.

CORRECTNESS: At least a high level outline of how the full proof would go. Possibly citing lemmas from class.

TIMING: A timing analysis.

The algorithms will be graded based on what you put in each bullet.

1 Union-Find using Trees of Bounded Depth

In this problem we will design Union operations so that each tree has bounded depth and thus the cost of each find will be $O(1)$.

1. Design a Union operation such that each tree has depth at most one, i.e. a child of a root is a leaf. The total cost of all the unions is $O(n \log n)$, where n is the total number of set elements.

2. Design a Union operation such that each tree has depth at most two. The total cost of all the unions is $O(n \log \log n)$.

Hint: Use your solution to the first part while the size of a tree is less than or equal to $\log n$.

2 Yet Another Convex Hull Algorithm (YACHA)

Consider the following 2D upper convex hull algorithm. The recursive invariant is that YACHA returns the upper convex hull not including the right most point of the point set $\{l\} \cup P \cup \{r\}$, where l is strictly to the left of the points in P and r is strictly to the right. For simplicity assume that no three points are co-linear.

YACHA(l, P, r)

0) If $l = r$ then return the empty list
1) Discard points in P on or below the line from l to r.
2) If P is empty return ________________________?
3) Pick a pairing of the points in P, say, $(p_1, q_1), \ldots, (p_k, q_k)$ and compute their respective slopes $s_1, \ldots, s_k$. Assume that p_i is to the left of q_i.
4) Pick the median slope s from $s_1, \ldots, s_k$ and find the support line with slope s. Let q be a point in P on this line.
5) Let P_1 be the set of points in P strictly left of q and P_2 be the set of point in P strictly right of q.
6) Discard q_i if $p_i, q_i \in P_1$ and $s_i \leq s$. Discard p_i if $p_i, q_i \in P_2$ and $s \leq s_i$.
7) Return YACHA(l, P_1, q) concatenate YACHA(q, P_2, r)

The function YACHA is the recursive helper function. Give the main function which calls YACHA. Explain how to handle the base case, step 2). Note that the code does not handle the case when a slope is infinite. Explain how it handles this case as well.

Show that this algorithm runs in time $O(n \log h)$ where n is the number of points and h is the number of points on the hull.

Here are few hints:

Use the recurrence

$$\begin{aligned} T(n, 1) &\leq cn \\ T(n, h) &\leq T(\alpha n, h_1) + T(\beta n, h_2) + cn \end{aligned}$$

where $\alpha + \beta \leq 1$, and $\alpha, \beta \leq \frac{3}{4}$, and $h_1 + h_2 \leq h$.

You may use the following fact: The function $f(x) = x^\alpha (h - x)^\beta$ achieves a maximum when $x = \alpha h$ and the value is **at most** $\alpha^\alpha \beta^\beta h$.

Your goal is to show that the size of each recursive call is at most $\frac{3}{4}n$. Consider two groups of points (1) those points in a pair with slope greater than the median s, P^+ and (2) those points in a pair with slope less than s, P^-. What proportion of each set can be in, say, the left recursive call?

3 Computing Maximum Augmenting Bottleneck Path

Let $G = (V, E, s, t, c)$ be a flow graph. Give an $O(m+n)$ time algorithm to find an augmenting path in G from s to t with maximum bottleneck capacity, where $n = |V|$ and $m = |E|$. Recall that the bottleneck capacity of a path is equal to the capacity of its minimum capacity edge.

Clarification: I can think of 3 solutions. Two of which use $O(m \log n)$ time. You can get partial credit for an $O(m \log n)$ time algorithm. You can get full credit for an $O(m+n)$ time algorithm for the undirected case, i.e., $c(u, v) = c(v, u)$. You will get extra credit for the directed case.

Hint: Suppose a little bird gave you the value of the bottleneck capacity. How could you find the path? Use binary search to find the bottleneck capacity.