UNIVERSITY OF EDINBURGH

FACULTY OF SCIENCE AND ENGINEERING

COMPUTER SCIENCE THREE

ALGORITHMS AND DATA STRUCTURES

Wednesday 6 June 2001

14:00 to 15:30

Examiners:

K. Turner (External)

G. Plotkin (Chairman)

INSTRUCTIONS TO CANDIDATES

Answer any TWO questions.

All questions carry equal weight.

1. Recall that the *rank* of an element x in a set S with a total order $\leq$ is the number $|\{y \in S : y \leq x\}|$ of elements less than or equal to x. This question concerns the task of identifying the element of given rank i within an (unordered) array A of n elements. For simplicity, assume those elements are all distinct.

   (a) The procedure PARTITION has the following specification. Given an array A, and indices p and r, with $p < r$, the procedure call

   $$q \leftarrow \text{PARTITION}(A, p, r)$$

   permutes the elements of the sub-array $A[p \mathinner{..} r]$ and returns an index q such that:

      i. the "pivot element" $x = A[q]$ is equally likely to be any of the $r - p + 1$ elements of $A[p \mathinner{..} r]$;
      ii. all array elements $A[p \mathinner{..} q - 1]$ are less than x;
      iii. all array elements $A[q + 1 \mathinner{..} r]$ are greater than x; and
      iv. the number of (pairwise) comparisons performed by PARTITION is $r - p$.

   Using PARTITION, present a randomised recursive procedure SELECT, based on QUICKSORT, that locates the element of given rank in an array. Specifically, the procedure call

   $$\text{SELECT}(A, p, r, i)$$

   should return the element of rank i in the array $A[p, r]$. [*8 marks*]

   (b) Demonstrate that, with probability at least $\frac{1}{2}$, both sub-arrays $A[p \mathinner{..} q - 1]$ and $A[q + 1 \mathinner{..} r]$ produced by PARTITION have size at most $\frac{3}{4}n$, where $n = r - p + 1$ is the size of the initial array. Hence show that, at each level of recursion in SELECT, the problem instance shinks to (at most) $\frac{3}{4}$ of its previous size, with probability at least $\frac{1}{2}$. [*6 marks*]

   (c) Let $T(n)$ denote the *expected* number of comparisons performed by SELECT when presented with an array of size n. Write down a recurrence that bounds $T(n)$ from above. Use the "Master Theorem" to show that $T(n) = O(n)$. (You may assume that $T(n)$ is monotonically increasing.) [*6 marks*]

   Hint. With probability at most $\frac{1}{2}$ the problem size does not necessarily shrink by as much as $\frac{3}{4}$. In that case, assume the worst, i.e., no shrinkage.

   (d) Let $W(n)$ denote the number of comparisons performed by SELECT, in the *worst case*, when presented with an array of size n. Write down a recurrence that bounds $W(n)$ from above and show that $W(n) = O(n^2)$. [*5 marks*]

2. The context for this question is the design and use of a data structure for representing disjoint sets.

   (a) Describe the operations MAKESET(x), UNION(x, y) and FINDSET(x) for disjoint sets, where x and y denote objects. *[3 marks]*

   (b) Describe a data structure for representing disjoint sets that supports an arbitrary sequence of m MAKESET, UNION and FINDSET operations, n of which are MAKESET, in total time $O(m \log n)$. Prove that your proposal meets the desired time bound. *[10 marks]*

   (c) Recall Kruskal's algorithm for finding a minimum-cost spanning tree in a weighted connected graph $G = (V, E)$:

       Sort the edge set E by increasing weight
       $F \leftarrow \emptyset$
       while $|F| < |V| - 1$ **do**
           let $e = \{u, v\}$ be the next edge in E
           if $F \cup \{e\}$ is acyclic **then** $F \leftarrow F \cup \{e\}$
       return F.

   Let $w_{\min}$ be the minimum weight of any edge in G. Prove that any minimum spanning tree in G must contain at least one edge of weight $w_{\min}$; i.e., prove that the first iteration of Kruskal's algorithm is "correct". *[6 marks]*

   (d) Demonstrate how the data structure from (b) can be used to implement Kruskal's algorithm in time $O(|E| \log |E|)$. To avoid trivialities, assume $|E| \geq 2$ so that $\log |E| > 0$. Again, prove that your implementation meets the desired time bound. *[6 marks]*

3. (a) An *interval* $I = [a, b]$ of the real line is specified by two real numbers $a \leq b$, and is defined to be the set of points x with $a \leq x \leq b$. Present a procedure to determine whether two intervals $I_1 = [a_1, b_1]$ and $I_2 = [a_2, b_2]$ intersect, i.e., whether $I_1 \cap I_2 \neq \emptyset$. [*3 marks*]

(b) A *rectangle* $R = [a, b] \times [c, d]$ in the plane is specified by real numbers $a \leq b$ and $c \leq d$, and is defined to be the set of points (x, y) with $a \leq x \leq b$ and $c \leq y \leq d$. Present a procedure to determine whether two rectangles $R_1 = [a_1, b_1] \times [c_1, d_1]$ and $R_2 = [a_2, b_2] \times [c_2, d_2]$ intersect. [*4 marks*]

(c) Consider the following problem: given a collection of rectangles $R_1, R_2, \ldots,$ R_n, generate a list of all pairs of rectangles that intersect. Present an algorithm that solves this problem in time $O(n \log n + m)$, where m is the number of intersecting pairs. Justify both the correctness and the claimed run-time of your algorithm. You may assume the existence of a data structure S holding up to n intervals that supports the following operations:

i. INSERT an interval into S;

ii. DELETE an interval from S;

iii. REPORT all intervals in S that intersect a given interval I.

The time taken for operations (i)–(iii) is, respectively, $O(\log n)$, $O(\log n)$ and $O(\log n + k)$, where k is the number of intersections reported. [*18 marks*]

Hint. Use the sweep-line strategy.

Note. Up to $\frac{2}{3}$ credit will be awarded for an algorithm that merely decides if there exists an intersecting pair, rather than reporting all of them.