CSE361 Computational Complexity of Algorithms. Spring 2008

Home work 1. Due on February 25, 2008.

1. There are $\sqrt{n}$ copies of an element in the array c. Every other element of c occurs exactly once. If the algorithm RepeatedElement is used to identify the repeated element of c, will the run time still be $\tilde{O}(\log n)$? If so, why? If not, what is the new run time?

2. Let $\mathcal{A}$ be a Monte Carlo algorithm that solves a decision problem π in time T. The output of $\mathcal{A}$ is correct with probability c, c being a constant greater than 1/2. Show how you can modify $\mathcal{A}$ so that its answer is correct with high probability. The modified version can take $O(T \log n)$ time.

3. In an infinite array, the first n cells contain integers in sorted order and the rest of the cells are filled with ∞. Present an algorithm that takes x as input and finds the position of x in the array in $\Theta(\log n)$ time. *You are not given the value of n.*

4. Find an efficient data structure for representing a subset S of the integers from 1 to n. Operations we wish to perform on the set are

   - **INSERT**(i) to insert the integer i to the set S. If i is already in the set, this instruction must be ignored.
   - **DELETE** to delete an arbitrary member from the set.
   - **MEMBER**(i) to check whether i is a member of the set.

   Your data structure should enable each one of the above operations in constant time (irrespective of the cardinality of S).

5. Input is a sequence X of n keys with many duplications such that the number of distinct keys is d ($< n$). Present an $O(n \log d)$-time sorting algorithm for this input. (For example, if $X = 5, 6, 1, 18,\ 6, 4, 4, 1,\ 5, 17$, the number of distinct keys in X is six.)

6. (a) Present an algorithm `HeightUnion` that uses the *height rule* for union operations instead of the weighting rule. This rule is defined below:

      [Height rule]: If the height of tree i is less than that of tree j, then make j the parent of i; otherwise make i the parent of j.

      Your program must run in $O(1)$ time and should maintain the height of each tree as a negative number in the `p` field of the root.

   (b) Show that the height bound of Lemma 2.3 (in the text) applies to trees constructed using the height rule.

   (c) Give an example of a sequence of unions that start with n singleton sets and create trees whose heights equal the upper bounds given in Lemma 2.3. Assume that each union is performed using the height rule.

7. Input is an array of n arbitrary real numbers (where n is odd). The array has $(n+1)/2$ distinct numbers such that each number has exactly two copies excepting for one number. Present an $O(n)$ time algorithm to identify the unique number.

8. Input is a (not necessarily sorted) sequence $S = k_1, k_2, \ldots, k_n$ of n arbitrary numbers. Consider the collection C of n^2 numbers of the form $\min\{k_i, k_j\}$, for $1 \le i, j \le n$. Present an $O(n)$-time and $O(n)$-space algorithm to find the median of C.

9. Two sets A and B have n elements each. Assume that each element is an integer in the range $[0, n^{100}]$. These sets are not necessarily sorted. Show how to check whether these two sets are disjoint in $O(n)$ time. Your algorithm should use $O(n)$ space.