BU CLA CS 530: Analysis of Algorithms

16 March 1999

Mid-Term Test

Reminder: This is a closed-book test. You are allowed one 8.5 by 11 crib sheet which you have to hand in with your test book. Read carefully all questions before you start, organize your time, start with what you know best.

Problem 1 (10 points).

Consider a set S of $n \geq 2$ distinct numbers given in unsorted order. Each of the following 3 questions asks you to give an algorithm satisfying a stated condition. In as few words as possible, describe your algorithm and justifiy its running time. (We measure the running time by counting the number of comparisons performed.)

a. In $\mathcal{O}(n \log n)$ time, determine whether a given number $x \notin S$ is the sum of two numbers in S.

b. In $\mathcal{O}(n)$ time, find $v, w \in S$ such that $|v - w| \geq |x - y|$ for all $x, y \in S$.

c. In $\mathcal{O}(n \log n)$ time, find $v, w \in S$ such that $v \neq w$ and $|v - w| \leq |x - y|$ for all $x, y \in S$ such that $x \neq y$.

Problem 2 (10 points).

Consider the following divide-and-conquer algorithm to find the maximum element in an array $A[1..n]$ of $n \geqslant 1$ integers. The main body of the algorithm makes a single call to the recursive procedure $\textsc{Maximum}(A, 1, n)$. The parameters p and r range over $\{1, \ldots, n\}$.

$\textsc{Minimum}(A, p, r)$
if $r - p \leqslant 1$
then **return** $\max\{A[p], A[r]\}$
else $x \leftarrow \textsc{Maximum}(A, p, \lfloor (p + r)/2 \rfloor)$
$y \leftarrow \textsc{Maximum}(A, 1 + \lfloor (p + r)/2 \rfloor, r)$
return $\max\{x, y\}$

a. Prove that the algorithm is correct. You may assume that n is a power of 2. *Hint*: Use induction on $n = r - p + 1$ that $\textsc{Maximum}(A, p, r)$ returns the maximum value in $A[p..r]$.

b. Write down a recurrence for the worst-case number of comparisons (between elements in the array A) used by $\textsc{Maximum}(A, 1, n)$. Solve this recurrence. You can assume that n is a power of 2.

c. What is the running time of $\textsc{Maximum}(A, 1, n)$? Explain your answer carefully.

Problem 3 (10 points).

Consider the following Monte Carlo algorithm:

```
MAJOR(A[1 . . n])
    i ← RANDOM(1, n)
    x ← A[i]
    k ← 0
    for j ← 1 to n
        do if A[j] = x then k ← k + 1
    if k > n/2 then return yes
               else return no
```

Algorithm MAJOR decides whether the array $A[1 \,..\, n]$ contains a majority element, with some error probability p. RANDOM is a random-number generator; a call to RANDOM$(1, n)$ returns an integer between 1 and n, inclusive, with each such integer being equally likely.

a. Determine a tight upper bound on the error probability p.

b. Consider another algorithm MAJOR′ which calls MAJOR as a subroutine:

```
MAJOR′(A[1 . . n])
    if MAJOR(A[1 . . n]) = yes
        then return yes
        else return MAJOR(A[1 . . n])
```

Determine a tight upper bound on the resulting error probability p'.

Hint for both parts: Consider separately two cases, when $A[1 \,..\, n]$ contains a majority element and when $A[1 \,..\, n]$ does not contain one.

Problem 4 (10 points).

Consider the problem of finding the maximum value in an array $A[1 \,..\, n]$ of $n \geqslant 1$ integers. An efficient sequential algorithm for this problem makes $(n-1)$ comparisons. Design a *work-efficient* EREW algorithm for this problem that runs in $\mathcal{O}(\log n)$ time. If you write the pseudo-code, explain briefly how your algorithm works. Explain why it is work-efficient.

Created: 99.03.15