CSC 220 Algorithms

Midterm Test A, March 31, 2003

1. Design an algorithm for simultaneously finding the second smallest *and* the second largest elements among n distinct numbers using at most $3n/2 + o(n)$ comparisons.

We can find the smallest (resp. largest) element in n-*1* steps by using a binary tree structure of depth $\lceil log_2 n \rceil$. At the first level we arrange the elements in $\lfloor n/2 \rfloor$ pairs, compare each element with its pair and discard the larger (resp. the smaller) one. Proceeding like this, at the bottom of the tree we find the smallest (resp. largest) element.

Notice that to find the smallest and the largest elements simultaneously, we need only $(n-1)+(n-1)-\lfloor n/2 \rfloor \leq \frac{3}{2}(n-1)$ comparisons, because at the first level after comparing each element with its pair, if we know which one is smaller it follows that the other one is larger. So it is sufficient to do these $\lfloor n/2 \rfloor$ comparisons only once.

To find the second smallest (resp. second largest) element, it is sufficient to determine the smallest (resp. largest) element among those numbers that have been compared with the smallest (resp. largest) element during the above procedure. This can be done using $\lceil log_2 n \rceil - 1$ comparisons, because there is at most one such element at each level.

So altogether the number of comparisons needed is at most $\frac{3}{2}(n-1)+2(\lceil log_2 n \rceil - 1) = \frac{3}{2}n + o(n)$, because $\lim_{n\to\infty} \frac{2\lceil log_2 n \rceil - 3.5}{n} = 0$.

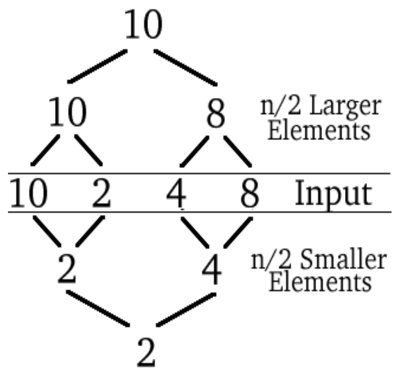

Figure 1: Min-Max Algorithm

2. Using the method of *QUICKSORT,* find the right ('splitting') position of 25 in the unsorted sequence

$$25, 20, 93, 19, 75, 82, 12, 41, 23.$$

Figure 2: Quicksort's Parition Algorithm

3. *MERGESORT* the following sequence:

$$20, 93, 19, 75, 82, 12, 41, 23.$$

Show all steps. What is the total number of comparisons?

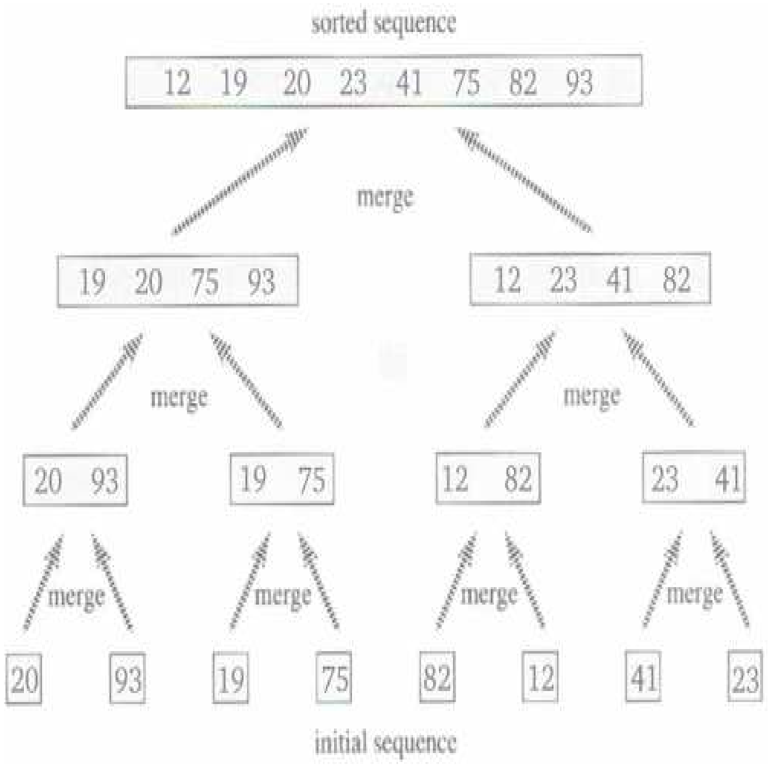

Figure 3: Mergesort Algorithm

Merge two sorted lists (of sizes p and q) by noticing that the smallest element in their union is the smaller of the smallest element on the first list and the smallest element of the second list. Whichever is smaller, erase it, and repeat the procedure for the truncated lists. This takes $\leq p+q-1$ comparisons.

Altogether the number of comparisons used by MERGESORT on n elements is $n\lceil log_2 n\rceil - 2^{\lceil log_2 n\rceil} + 1$ which is 17 for n=8.

4. Let $f(0) = 0$ and $f(n) \leq 2f(\lfloor n/3 \rfloor) + n$ for every $n \geq 1$. Prove that $f(n) \leq 5n$ for every n.

We prove $f(n) \leq 5n$ by induction on n. For n=0 it is true. Assume that $n > 0$ and that we have already proved that $f(k) \leq 5k$ for every $k < n$. Then we have $f(n) \leq 2f(\lfloor n/3 \rfloor) + n \leq 2*5\lfloor n/3 \rfloor + n \leq 10n/3 + n \leq 5n$, as required. (At the second inequality, we applied the induction hypothesis $f(k) \leq 5k$ with $k = \lfloor n/3 \rfloor < n$).

5. Suppose that there exists an algorithm for finding the second smallest element among n distinct numbers, using at most $f(n)$ comparisons in the worst case.

Show that then one can also identify simultaneously the smallest *and* the second smallest elements, using at most $f(n)$ comparisons in the worst case.

Suppose we have an algorithm A for finding the second smallest among n distinct numbers using at most f(n) comparisons. Denote the ouput of A by x. Clearly, A must have compared x to atleast one other element, otherwise it would be impossible to come to any conclusion about its position.

Suppose that A compared x to the elements $y_1, y_2, y_3, .., y_k$. We claim that x must turn out to be larger than precisely one of these elements. Indeed, if x was smaller than all y's, then we could not have ruled out the possibility that x is the smallest element. If x was larger than two y's, then it could not be the second smallest.

Consequently, x turned out to be larger than one element, say, y_j, and then this element must be the smallest element, since it is smaller than the second smallest. Moreover, we established this fact without any extra comparison.

6. Let $f(n)$ and $g(n)$ be positive functions defined on the set of positive integers, and assume that $f(n) = o(g(n))$. Is it true that

(a) $f(n^2) + 1 = o(g(n^2) + 1)$; False, because of the following counterexample: $f(n) = \frac{1}{n}, g(n) = 1$. Then we have $f(n) = o(g(n))$ because $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty}(1/n) = 0$. On the other hand, $\lim_{n\to\infty} \frac{f(n^2)+1}{g(n^2)+1} = \lim_{n\to\infty} \frac{1/n^2+1}{1+1} = \frac{1}{2} \neq 0$, so $f(n^2) + 1 \neq o(g(n^2) + 1)$.

(b) $f^2(n+1) = o(g^2(n+1))$? True. Since $f(n) = o(g(n))$, we have $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{f(n+1)}{g(n+1)} = 0$. Therefore, $\lim_{n\to\infty} \frac{f^2(n+1)}{g^2(n+1)} = \lim_{n\to\infty}(\frac{f(n+1)}{g(n+1)})^2 = 0$, which means that $f^2(n+1) = o(g^2(n+1))$, by definition.

Addendum

```
QUICKSORT(A, p, r)
    if p < r
        then q ← PARTITION(A,p,r)
             QUICKSORT(A,p,q)
             QUICKSORT(A,q+1,r)
```

```
PARTITION(A, p, r)
    pivot ← A[p]
    i ← p − 1
    j ← r + 1
    while TRUE
        do   repeat j ← j − 1
                 until A[j] ≤ pivot
             repeat i ← i + 1
                 until A[i] ≥ pivot
             if i < j
                 then exchange A[i] ↔ A[j]
                 else return j
```

```
MERGE-SORT(A,p,r)
if p < r
      then q ← ⌊(p + r)/2⌋
      MERGE( MERGE-SORT(A,p,q) , MERGE-SORT(A,q+1,r) )
```