CSC 220 Algorithms

Midterm Test B, March 31, 2003

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

(a) $h(f(n)) = O(h(g(n))$;

False, because of the following counterexample: $f(n) = n, g(n) = 2n, h(n) = \frac{1}{2^n}$. Then $f(n) < g(n)$ for every n, so f(n) = O(g(n)). On the other hand, there exists no constant C such that $h(f(n)) = \frac{1}{2^n} \leq Ch(g(n)) = C\frac{1}{2^{2n}}$ for all sufficiently large n, because then $\frac{2^{2n}}{2^n} \leq C \Rightarrow 2^n \leq C$ for all n, which is impossible.

(b) $f(2n) = O(g(3n))$?

False, because of the following counterexample: $f(n) = g(n) = \frac{1}{2^n}$. Then f(n) = O(g(n)), but there exists no constant C such that $f(2n) = \frac{1}{2^{2n}} \leq Cg(3n) = C\frac{1}{2^{3n}}$ for all n, because then we would have $\frac{2^{3n}}{2^{2n}} \leq C \Rightarrow 2^n \leq C$ for all n, which is impossible.

2. Using the method of *QUICKSORT*, find the right ('splitting') position of 67 in the unsorted sequence

$$67, 100, 38, 79, 58, 25, 41, 3, 66.$$

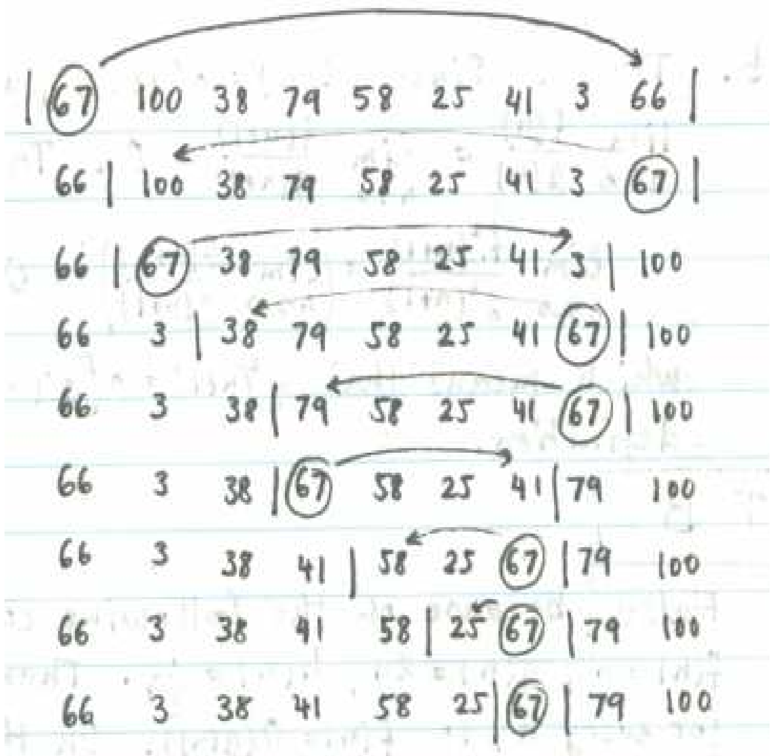

Figure 1: Quicksort's Parition Algorithm

3. *MERGESORT* the following sequence:

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

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

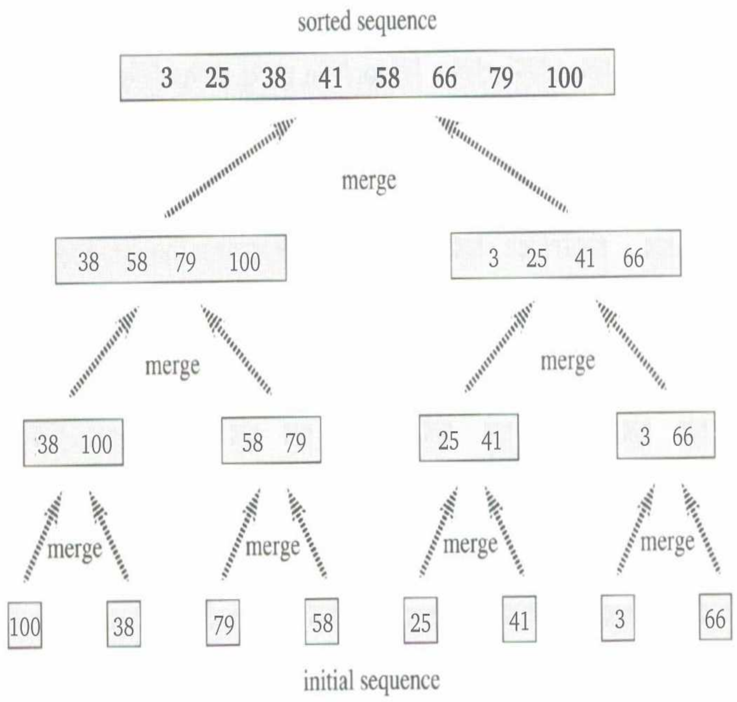

Figure 2: Mergesort Algorithm

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 $n\lceil log_2 n\rceil - 2^{log_2 n} + 1$ which is 17 for n=8.

4. Let $f(0) = 0$ and

$$f(n) \le n - 1 + \sum_{i=0}^{n-1} f(i)$$

for every $n \ge 1$. Show that $f(n) \le 2^{n-1}$ for all $n \ge 1$.

f(n) attains its largest value when all inequalities become equations, so it is sufficient to solve the recurrence relation.

(*) f(0) 0
f(n) $= (n-1) + \sum_{i=0}^{n-1} f(i)$ for every $n \ge 1$
For instance, for n=1, $f(1) = 1 - 1 + \sum_{i=0}^{0} f(i) = 0$
Substituting n-1 for n, we obtain
(**) f(n-1) $= (n-1) + \sum_{0}^{n-1} f(i)$ for every $n \ge 2$
Subtracting **(**)** from **(*)**, we get
f(n) $-f(n-1) = 1 + f(n-1)$
f(n) $= 2f(n-1) + 1$
Yielding the recurrence
f(1) $= 0$
(***) f(n) $= 1 + 2f(n-1)$
f(n) $= 2f(n-1) + 1$
f(n) $= 2(2f(n-2) + 1) + 1$
f(n) $= 2(2(2f(n-3) + 1) + 1) + 1$
:
:
f(n) $= 2^{n-1}f(1) + 2^{n-2} + 2^{n-3} + ... + 2^1 + 2^0$
f(n) $= 2^{n-2} + 2^{n-3} + ... + 2^1 + 2^0$ $\qquad$ $f(1) = 0$
f(n) $= 2^{n-1} - 1$ $\qquad$ $\sum_{i=0}^{n} 2^i = 2^{n+1} - 1$
f(n) $\le 2^{n-1}$ for every $n \ge 2$

One can also prove by an easy induction that **(***)** implies $f(n) = 2^{n-1} - 1$ for all $n \ge 1$. This is true for n=1. Let $n \ge 2$, and assume that we have already shown that $f(k) = 2^{k-1} - 1$ for all $k < n$. Then $f(n) = 2f(n-1) + 1 = 2(2^{n-2} - 1) + 1 = 2^{n-1} - 1$ as required, where the second equation follows from the induction hypothesis with k = n-1.

5. We have 9 coins that look the same. 3 of them have weight 1, 3 others have weight 0.9, and the remaining 3 have weight 0.95. Prove that any algorithm for determining the weight of each coin by a two-pan balance uses at least 7 measurements in the *worst case*.

According to the **Information Theory Bound**, any algorithm for determining the weight of each coin by a two-pan balance requires at least $\lceil log_k N \rceil$ measurements in the worst case, where the number of outcomes of a single measurement k=3 (the left side can be lighter than, heavier than, or equal to the right side), and the total number of possible outputs of the algorithm N = C(9,3) * C(6,3) = $\frac{9*8*7}{3*2*1} * \frac{6*5*4}{3*2*1} = 1680$. Thus, we have $\lceil log_k N \rceil = \lceil log_3 1680 \rceil = 7$.

6. Design an algorithm for simultaneously finding the smallest element, the largest element, *and* the median among n distinct numbers using fewer than $24n$ comparisons in the *worst case*, provided that n is sufficiently large.

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.

We have learned in class that the median of n elements can be determined using at most 22n comparisons. So, altogether one can simultaneously find the smallest, the largest elements using at most $3/2(n-1)+22n < 23.5n < 24n$ comparisons in the worst case.

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) )
```