CS584
HW#2

DUE: Monday, October 5

1. (10 points) We want to test if every element of an array $A[1..n]$ is distinct. The benchmark operation is for any $1 \le i, j \le n$ to test $A[i] \stackrel{?}{=} A[j]$ which returns either $A[i] = A[j]$ or $A[i] \ne A[j]$. Provide an algorithm to solve this problem which is optimal in the worst-case, and prove that your algorithm is optimal. Do not use asymptotic notation; you must count exactly the number of comparisons $A[i] \stackrel{?}{=} A[j]$ that your algorithm does.

2. (12 points) We want to test if every element of an array $A[1..n]$ is distinct. The benchmark operation is for any $1 \le i, j \le n$ to test $A[i] \stackrel{?}{=} A[j]$ which returns one of three answers: $A[i] = A[j]$, $A[i] < A[j]$ or $A[i] > A[j]$. Provide an algorithm to solve this problem which is optimal in the worst-case, and prove that your algorithm is asymptotically optimal. Note that the answers to tests in this problem yield more information that the answers to tests in the previous problem, so the complexities may be different. (Hint: What set of answers to questions could permit your algorithm to infer that $A[i] \ne A[j]$ for every $1 \le i, j \le n$?)

3. (8 points) *Halving* is the process of dividing a set *S* into essentially equal sized subsets, $S_1, S_2, |S_1| = |S_2| \pm 1$ such that $(\forall x \in S_1)(\forall y \in S_2)\, x \le y$. Assume the median of a set of *n* elements can be found using at most $t_m(n)$ pairwise comparisons, and a set of *n* elements can be halved using at most $t_h(n)$ pairwise comparisons.

a Show that $t_h(n) \le t_m(n) + n - 1$.

b Show that $t_m(n) \le t_h(n) + \lceil n/2 \rceil - 1$.

4. Assume you want to INSERT a new element into a heap on $n \ge 1$ elements. For the following, show exact answers (with floors $\lfloor\ \rfloor$ and ceilings $\lceil\ \rceil$ as needed).

a (1 point) How many elements, $f(n)$, are on the path from where the new element in INSERTed to the root, counting the root but not counting the new element. For example, 1 comparison is needed for *n*=1.

b (3 points) Since the $f(n)$ elements on the path from where the new element in INSERTed to the root are sorted, propose and analyze a faster technique that the one from class and the text to locate **where** to INSERT the new element.

c (3 points) With your faster location technique from part ***b***, analyze the worst-case time to actually INSERT the new element.

HW#2
SOLUTIONS

1. An optimal algorithm is:

for $i \leftarrow 1$ **to** n-1 **do**
  for $j \leftarrow i+1$ **to** n **do**
    if $A[i] = A[j]$ **then return** "All elements are not distinct."
return "All elements are distinct."

The algorithm does $\binom{n}{2} = \frac{n(n-1)}{2}$ comparisons. To show (by contradiction) that $\binom{n}{2}$ comparisons are necessary in the worst-case, we assume there is an algorithm A^* to solve this problem which always uses $< \binom{n}{2}$ comparisons. On input $(1, 2, 3, ..., n)$ A^* returns "All elements are distinct.". But since there are $\binom{n}{2}$ pairs $(i, j), 1 \le i < j \le n$, there must be some pair, say (i^*, j^*) which A^* never examined. We change A to B by changing $A[i]$ to be $A[j]$, everything else remaining the same. A^* asks the same questions on B as on A, and it gives the same answer. But the answer is now wrong, and by contradiction A^* can not exist.

2.

MERGESORT(A)	$\Theta(n \lg n)$
for $i \leftarrow 1$ **to** n-1 **do**	$O(n)$
if $A[i] = A[i+1]$ **then return** "All elements are not distinct."	
return "All elements are distinct."	$O(1)$

To show that $\Omega(n \lg n)$ comparisons are necessary in the worst-case, we consider an input in which every element is distinct. The only way any algorithm can infer $A[i] \ne A[j]$ for some pair (i, j) is to have performed some nonempty sequence of comparisons yielding the results $A[i] < A[k_1], A[k_1] < A[k_2], ..., A[k_l] < A[j]$. Hence the algorithm must "know" (must have access to information permitting the inference) the relative order between every pair of elements. Since this is enough information to permit sorting A, the $\Omega(n \lg n)$ bound on the number of pairwise comparisons needed to sort n numbers must apply.

3. Without loss of generality, assume that $|S_1| \geq |S_2|$.

a compute the median μ of S $t_m(n)$

$S \leftarrow S - \{\mu\}$

$S_1, S_2 \leftarrow \varnothing$

for each $s \in S$ **do**

if $s \leq \mu$ **then** $S_1 \leftarrow S_1 \cup \{s\}$ **else** $S_2 \leftarrow S_2 \cup \{s\}$ n-1

$S_1 \leftarrow S_1 \cup \{\mu\}$

if $|S_1| > |S_2| + 1$ **then** move enough μ s from S_1 to S_2 to balance them

b halve S $t_h(n)$

return $\max(S_1)$ $\lceil n/2 \rceil - 1$

4. ***a*** $\lfloor \lg(n+1) \rfloor$

b $\lceil \lg(\lfloor \lg(n+1) \rfloor) + 1 \rceil = \lceil \lg \lceil \lg(n+2) \rceil \rceil = \lceil \lg \lg(n+2) \rceil$

c $\lceil \lg(n+2) \rceil$