CS524
HW#2

DUE: Wednesday, October 5

1. (10 points) There are *m* nuts of distinct sizes $(n_1,...,n_m)$ and *m* bolts of distinct sizes $(b_1,...,b_m)$. Each bolt fits **exactly** one nut. You may not compare two nuts directly to each other, nor may you compare two bolts directly to each other. The only (benchmark) operation you may do is to compare a nut, say n_i, to a bolt, say b_j. The three possible outcomes of a comparison are that

- $n_i < b_j$, the nut is too small for the bolt,
- $n_i = b_j$, the nut fits the bolt,
- $n_i > b_j$, the nut is too large for the bolt.

Your goal is to find an algorithm for finding the correct nut for each bolt, and your algorithm must execute in expected time in $\Theta(m \lg m)$. You may describe your algorithm in precise pseudocode. (Hint: Think of a variation of QUICKSORT.)

2. (6 points) Assume that you are given an array *A*[1..*n*] where $A[i] \in \{0,1\}$ for $1 \le i \le n$.

a Show that an upper bound on the complexity of sorting is in *O*(*n*).

b Show that a lower bound on the complexity of sorting is in $\Omega(n)$.

C.S.524

SOLUTION FOR H.W. #2

1. NUTBOLTMATCH(*nuts*, *bolts*)

```
if (|nuts|>1 ∨ |bolts|>1) then
        LargeNuts, SmallNuts, LargeBolts, SmallBolts ← ∅
        Select and remove a random nut n* from nuts
        for each bolt b
                if b= n* then call the bolt b*
                else if b> n* then LargeBolts ← LargeBolts ∪ {b}
                                else SmallBolts ← SmallBolts ∪ {b}
        for each nut n
                if b*>n then SmallNuts ← SmallNuts ∪ {n}
                        else LargeNuts ← LargeNuts ∪ {n}
NUTBOLTMATCH(SmallNuts,SmallBolts)
NUTBOLTMATCH(LargeNuts,LargeBolts)
return(n*,b*)
```

Since the algorithm is essentially QUICKSORTing *nuts* **and** *bolts*, the expected time of the two QUICKSORTs is $\Theta(m \lg m)$.

2. ***a*** The following algorithm works in $O(n)$ time.

```
NUM0S ← 0
for i ← 1 to n do
    if A[i] = 0 then NUM0S ← NUM0S+1
for i ← 1 to NUM0S do
    if A[i] ← 0
for i ← NUM0S + 1 to n do
    if A[i] ← 1
```

b Assume that $A[i] \in \{0,1\}$ is sorted by an algorithm in less than linear time. At least one element of A, say $A[i]$, wasn't examined. Modify A by setting $A[i] \leftarrow |A[i]-1|$ and keeping all other values unchanged. The algorithm must give the same output as before (because all elements it examines are unchanged), but the answer is incorrect.