CS584
HW#2

DUE: Wednesday, October 1

1. (10 points) Assume we are given two lists of numbers, $a_1,...,a_n$ and $b_1,...,b_n$, and you know that

- $\forall 1 \le i < j \le n \;\; a_i \ne a_j$
- $\forall 1 \le i < j \le n \;\; b_i \ne b_j$
- There is a bijection π between $a_1,...,a_n$ and $b_1,...,b_n$ such that $\forall 1 \le i \le n \; a_i = b_{\pi(i)}$

The benchmark operation is to test, for any pair of numbers a_i, b_j, whether $a_i < b_j$, $a_i = b_j$ or $a_i > b_j$.

a Prove that the complexity of computing π is $O\left(n^2\right)$.

b Prove that the complexity of computing π is $\Omega\left(n \lg n\right)$.

2. (8 points) An array *A*[1..*n*] is *unimodal* if there is a unique *mode j*, $1 \le j \le n$, such that $A[i] > A[i+1]$ for $n \ge i \ge j$ and $A[i] > A[i-1]$ for $1 \le i \le j$.

a Prove that the complexity of testing if *A* is unimodal is in $\Omega\left(n\right)$.

b Prove that the worst-case complexity of finding the mode of a unimodal array is in $O\left(\lg n\right)$.

3. (15 points) As an implementation of a MIN-PRIORITY-QUEUE, consider the *Young Tableau* introduced in ***Problem 6-3*** on pages 143→144 of our text. Solve ***Problem 6-3***.

C.S.584
Solutions for H.W. #2

1. ***a*** **for** $i \leftarrow 1$ **to** n **do**
` for $j \leftarrow 1$ to n do
if $a_i = b_j$ then $\pi(i) \leftarrow j$

b There are $n!$ possible pairings (matchings) of $a_1, ..., a_n$ with $b_1, ..., b_n$, so the decision tree computation must have at least $n!$ leaves. Each comparison of a_i with b_j yields three possible outcomes: $a_i < b_j$, $a_i = b_j$ or $a_i > b_j$, so the internal nodes of the decision tree have degree 3. A decision tree of internal degree three with $n!$ leaves has a path of length at least

$\log_3 n! \geq \log_3 \left(\frac{n}{e}\right)^n = n \log_3 n - n \log_3 e \in \Omega(n \lg n)$, where the inequality follows from Stirling's inequality, Equation 3.19 of our text.

2. ***a*** A necessary condition for an array to be unimodal with mode j is that it must be sorted in increasing order for i from $1 \leq i \leq j$ and in decreasing order for i from $n \geq i \geq j$. Assume there is an algorithm to test if unimodal array A is indeed unimodal in time in $o(n)$. Then some element, say $A[k]$, is not examined by the algorithm. Change A to A^* where

$$A^*[i] = \begin{cases} A[i], \text{if } i \neq k \\ -\infty, \text{if } i = k \leq j \\ \infty, \text{if } i = k > j \end{cases}$$

We note that A^* is not unimodal. But every question asked by the algorithm of A^* is exactly the same question asked of A by the algorithm, and it receives the same answer. Hence it must report that A^* is unimodal, which is incorrect, and contradicts that the algorithm works correctly. Hence the complexity of testing for unimodality is $\Omega(n)$.

b If we know that *A* **is** unimodal, then we seek the mode by binary search.

$lo \leftarrow 1$
$hi \leftarrow n$
repeat
$mid \leftarrow (hi + lo)/2$
if $(A[mid-1] < A[mid]) \wedge (A[mid+1] < A[mid])$
then return *mid*
else $A[mid-1] < A[mid]$ then $lo \leftarrow mid+1$
else $hi \leftarrow mid-1$

3. ***a*** One of many Young Tableaux for these numbers is

2	3	8	15
4	8	∞	∞
5	14	∞	∞
12	∞	∞	∞

b Denote the Young Tableau $Y[1..m,1..n]$. Since the elements in the top row are sorted, $\infty = Y[1,i] \le Y[1,i+1], 1 \le i < n$, it must be the case that $Y[1,i] = \infty, 1 \le i \le n$. Since the elements in each column are sorted, $Y[i,j] \le Y[1,j] = \infty, 1 \le i \le m, 1 \le j \le n$, so *Y* only contains ∞s, and must be empty. Since the elements in the bottom row are sorted, $Y[m,i] \le Y[m,i+1] < \infty, 1 \le i \le i+1$, it must be the case that $Y[m,i] < \infty, 1 \le i \le n$. Since the elements in each column are sorted, $Y[i,j] \le Y[m,j] < \infty, 1 \le i \le m, 1 \le j \le n$, so *Y* doesn't contain any ∞s, and must contain *mn* finite elements.

c To simplify the algorithm, we assume that *Y* has a column *n*+1 and a row *m*+1, all of whose entries are ∞.

Extract-Min(*Y*)
return Extract-Min-Aux(1,1)
Extract-Min-Aux(i_{lo},j_{lo})
$Min \leftarrow Y[i_{lo}, j_{lo}]$
if $i_{lo} = m \wedge j_{lo} = n$ then $Y[i_{lo}, j_{lo}] \leftarrow \infty$
else if $Y[i_{lo}+1, j_{lo}] \le Y[i_{lo}, j_{lo}+1]$
then $Y[i_{lo}, j_{lo}] \leftarrow$ EXTRACT-MIN-AUX(i_{lo+1},j_{lo})
else $Y[i_{lo}, j_{lo}] \leftarrow$ EXTRACT-MIN-AUX(i_{lo},j_{lo+1})
return *Min*

For every recursive call of EXTRACT-MIN-AUX, either i_{lo} or j_{lo} increases by 1. Since every invocation of EXTRACT-MIN-AUX takes time in $O(1)$, and ultimately $i_{lo} + j_{lo} \le m+n$, it follows that the execution time of EXTRACT-MIN is in $O(m+n)$.

d To simplify the algorithm, we assume that Y has a column n+1 and a row m+1, all of whose entries are ∞.

INSERT(x,Y)
    Insert-Aux(x,1,1)
INSERT-AUX(x,i_{lo},j_{lo})
    if $x > Y\left[i_{lo}, j_{lo}+1\right] \geq Y\left[i_{lo}+1, j_{lo}\right]$
      then INSERT-AUX(x,i_{lo}+1,j_{lo})
    else if $x > Y\left[i_{lo}+1, j_{lo}\right] \geq Y\left[i_{lo}, j_{lo}+1\right]$
      then INSERT-AUX(x,i_{lo},j_{lo}+1)

e Since EXTRACT-MIN and INSERT each take time in $O(m+n)$, we start with an empty Young Tableau and INSERT each of the n^2 numbers, in time in $O\left(n^3\right)$, and then EXTRACT-MIN each of the n^2 numbers, also in time in $O\left(n^3\right)$.

f The function MEMBER(x,i_{lo},i_{hi},j_{lo},j_{hi}), which tests whether or not x belongs to $Y\left[i_{lo}...i_{hi}, j_{lo}..j_{hi}\right]$, is invoked by MEMBER(1,m,1,n), and it eliminates one column (the left one) or one row (the left one) with each recursive call. It does this by comparing x to the bottom left element of $Y\left[i_{lo}...i_{hi}, j_{lo}..j_{hi}\right]$.

MEMBER(x,i_{lo},i_{hi},j_{lo},j_{hi})
    if $i_{lo} = i_{hi} \wedge j_{lo} = j_{hi}$
      then return x==Y[i_{lo},j_{lo}]
    else if x=Y[i_{hi},j_{lo}]
      then return true
    else if x>Y[i_{hi},j_{lo}]
      then return MEMBER(i_{lo},i_{hi},j_{lo}+1,j_{hi}) ▸ remove left column
      else return MEMBER(i_{lo},i_{hi}-1,j_{lo},j_{hi}) ▸ remove bottom row