CS525DA
HW#3

DUE: Tuesday, October 14

1. (6 points) Suppose you're trying to find the MAXIMUM and the MINIMUM of *n* numbers $a_1, \ldots, a_n$.

(a) How many possible answers are there?

(b) Noting that your answer to part (a) is a lower bound on the number of leaves of a decision tree to compute the MAXIMUM and the MINIMUM of *n* numbers, what lower bound does the decision tree model provide for the worst-case number of pairwise comparisons necessary to solve the problem? Do not use asymptotic notation.

2. (6 points) Do ***Problem 8.4 a*** and ***b*** on pages 179→180 of our text.

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***.

CS525DA
HW#3 SOLUTIONS

1. (a) The number of pairs of integers $(i, j), 1 \le i, j \le n$, is $\left|\{1,...,n\} \times \{1,...,n\}\right| = n^2$

(b) Since the height of a binary tree with m leaves is at least lg$m$, any algorithm to compute the MAXIMUM and the MINIMUM of n numbers must use, in the worst-case, at least $\lg n^2 = 2\lg n$.

2. (a) Call the jugs $r_1,...,r_n,b_1,...,b_n$.

for $i \leftarrow 1$ **to** n **do**
` **for** $j \leftarrow 1$ **to** n **do**
if EQUALVOLUME(r_i,b_j) **then** PAIR(r_i,b_j)

(b) There are n! possible pairings (matchings) of $r_1,...,r_n$ with $b_1,...,b_n$, so the decision tree computation must have at least n! leaves. Each comparison of r_i with b_j yields three possible otcomes: $r_i < b_j$, $r_i = b_j$ or $r_i > b_j$, so the internal nodes of the decision tree have degree three. A decision tree of internal degree three with n! leaves has a path of length at least

$$\log_3 n! \ge \log_3 \left(\frac{n}{e}\right)^n = n\log_3 n - n\log_3 e \in \Omega(n\lg n).$$

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 $Y[i_{lo}+1, j_{lo}] \leq Y[i_{lo}, j_{lo}+1]$

    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} increases by 1 or j_{lo} increases by 1. Since every invocation of EXTRACT-MIN-AUX takes time in *O*(1), and ultimately $i_{lo} + j_{lo} \leq m + n$, 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 $x > Y[i_{lo}, j_{lo}+1] \geq Y[i_{lo}+1, j_{lo}]$

    then INSERT-AUX(*x*,i_{lo}+1,j_{lo})

  else if $x > Y[i_{lo}+1, j_{lo}] \geq Y[i_{lo}, j_{lo}+1]$

    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 $O(n^3)$, and then EXTRACT-MIN each of the n^2 numbers, also in time in $O(n^3)$.

(f) The function MEMBER(*x*,i_{lo},i_{hi},j_{lo},j_{hi}), which tests whether or not *x* belongs to $Y[i_{lo}...i_{hi}, j_{lo}..j_{hi}]$, 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 $Y[i_{lo}...i_{hi}, j_{lo}..j_{hi}]$.s

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