Suggested Solutions to Midterm Problems

Problems

1. Given any binary tree T, let l_T denote the number of its leaves and m_T the number of its internal nodes.

   (a) Prove *by induction* that, if every internal node of T has two children, then $l_T - m_T = 1$. (8 points)

   Solution. The proof is by *strong induction* on the number n_T of nodes of an arbitrary binary tree T where every internal node has two children.

   Base case ($n_T = 1$): $l_T = 1$ and $m_T = 0$. Apparently, $l_T - m_T = 1$.

   Inductive step ($n_T > 1$): Let T_1 and T_2 denote respectively the left and the right subtrees of T's root (which is an internal node of T and hence has two children). It is clear that every internal node of T_1 and T_2 has two children. From the induction hypothesis, $l_{T_1} - m_{T_1} = 1$ and $l_{T_2} - m_{T_2} = 1$. The leaves of T_1 and T_2 are also leaves of T and the internal nodes of T_1 and T_2 are also internal nodes of T; therefore, $l_{T_1} + l_{T_2} = l_T$ and $m_{T_1} + m_{T_2} = m_T - 1$ (subtracting one for the root). It follows that $l_T - m_T = (l_{T_1} + l_{T_2}) - (m_{T_1} + m_{T_2} + 1) = (l_{T_1} - m_{T_1}) + (l_{T_2} - m_{T_2}) - 1 = 1$. $\square$

   (b) Use the preceding result to show that, if T is a complete binary tree, then either $l_T - m_T = 1$ or $l_T - m_T = 0$. (2 points)

   Solution. In a complete binary tree T, every internal node except the last one (counting from top to bottom and left to right) has two children. If the last internal node also has two children, then from Part (a) we have $l_T - m_T = 1$. Otherwise, add a second child to the last internal node of T to obtain another complete binary tree T' such that every internal node of T' has two children; it is clear that $l_{T'} = l_T + 1$ and $m_{T'} = m_T$. From Part (a), $l_{T'} - m_{T'} = 1$ and hence $l_T - m_T = (l_{T'} - 1) - m_{T'} = 0$. $\square$

2. Let $a_1, a_2, \cdots, a_n$ be positive real numbers such that $a_1 a_2 \cdots a_n = 1$. Prove *by induction* that $(1 + a_1)(1 + a_2) \cdots (1 + a_n) \geq 2^n$. (Hint: In the inductive step, try introducing a new variable that replaces two chosen numbers from the sequence.)

   Solution. The proof is by induction on n.

   Base case ($n = 1$): $a_1 = 1$. So, $(1 + a_1) = 2 \geq 2^1$.

   Inductive step ($n > 1$): In any sequence $a_1, a_2, \cdots, a_n$ ($n > 1$) of positive real numbers where $a_1 a_2 \cdots a_n = 1$, there must exist two numbers a_i and a_j such that $a_i \geq 1$ and $a_j \leq 1$. Without loss of generality, we assume that the two numbers are a_{n-1} and a_n (this can always be achieved by swapping numbers in the sequence). As $(1-a_{n-1})(1-a_n) \leq 0$, it follows that $a_{n-1}+a_n \geq 1+a_{n-1}a_n$. Let a'_{n-1} be the number equal to $a_{n-1}a_n$ (which is also a positive real number) so that $a_1a_2\cdots a_{n-2}a'_{n-1} = a_1a_2\cdots a_{n-2}a_{n-1}a_n = 1$.

$(1+a_1)(1+a_2)\cdots(1+a_{n-2})(1+a_{n-1})(1+a_n) = (1+a_1)(1+a_2)\cdots(1+a_{n-2})(1+a_{n-1}+a_n+a_{n-1}a_n) \geq (1+a_1)(1+a_2)\cdots(1+a_{n-2})((1+a_{n-1}a_n)+(1+a_{n-1}a_n)) = 2(1+a_1)(1+a_2)\cdots(1+a_{n-2})(1+a_{n-1}a_n) = 2(1+a_1)(1+a_2)\cdots(1+a_{n-2})(1+a'_{n-1})$, which from the induction hypothesis $\geq 2\times 2^{n-1} = 2^n$. □

3. For each pair f,g of functions, indicate whether $f(n) = O(g(n))$ and/or $f(n) = \Omega(g(n))$. (Stirling's approximation: $n! = \sqrt{2\pi n}\left(\frac{n}{e}\right)^n(1+O(1/n))$.)

$f(n)$	$g(n)$
$2n+\log n$	$n+(\log n)^2$
$(\log n)^{\log n}$	n
3^n	$3^{\frac{n}{2}}$
$\log(n!)$	$\log(n^n)$

Solution. (Tsai, Ming-Hsien)

(a)

$$\begin{aligned}\lim_{n\to\infty}\tfrac{2n+\log n}{n+(\log n)^2} &= \lim_{n\to\infty}\tfrac{2+\frac{1}{n}}{1+2\frac{1}{n}\log n}\\ &= \lim_{n\to\infty}\tfrac{2n+1}{n+2\log n}\\ &= \lim_{n\to\infty}\tfrac{2}{1+\frac{2}{n}}\\ &= 2\end{aligned}$$

Therefore, $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$.

(b) $f(n) = (\log n)^{\log n} = 10^{\log n \log\log n}$; $g(n) = n = 10^{\log n}$.

Therefore, $f(n) \geq g(n)$ and hence $f(n) = \Omega(g(n))$.

(c) $f(n) = 3^n = 3^{\frac{n}{2}}\cdot 3^{\frac{n}{2}} \geq 3^{\frac{n}{2}} = g(n)$.

Therefore, $f(n) = \Omega(g(n))$.

(d)

$$\begin{aligned}\lim_{n\to\infty}\tfrac{\log n!}{\log n^n} &= \lim_{n\to\infty}\tfrac{\log(\sqrt{2\pi n})(\frac{n}{e})^n(1+O(1/n))}{n\log n}\\ &= \lim_{n\to\infty}\tfrac{\log(\sqrt{2\pi})+\frac{1}{2}\log n+n\log n-n\log e}{n\log n}\\ &= \lim_{n\to\infty}\tfrac{\frac{1}{2n}+\log n+1-1}{\log n+1}\\ &= \lim_{n\to\infty}\tfrac{1+2n\log n}{2n\log n+2n}\\ &= \lim_{n\to\infty}\tfrac{2\log n+2}{2\log n+2+2}\\ &= \lim_{n\to\infty}\tfrac{\frac{2}{n}}{\frac{2}{n}}\\ &= 1\end{aligned}$$

Therefore, $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$.

□

4. Modify the following code for determining the sum of the maximum consecutive subsequence so that it also records the start and end indices of the subsequence.

Algorithm Max_Consec_Subseq (X, n);
begin
  Global_Max := 0;
  Suffix_Max := 0;
  for i := 1 **to** n **do**
    if $x[i]$ + *Suffix_Max* > *Global_Max* **then**
      Suffix_Max := *Suffix_Max* + $x[i]$;
      Global_Max := *Suffix_Max*
    else if $x[i]$ + *Suffix_Max* > 0 **then**
        Suffix_Max := *Suffix_Max* + $x[i]$
    else *Suffix_Max* := 0
end

Solution. (Tsai, Ming-Hsien)

```
Algorithm Max_Conseq_Subseq(X, n);
begin
    Global_Max := 0;
    Suffix_Max := 0;
    Suffix_Start_Index := 0;
    Global_Start_Index := 0;
    Global_End_Index := 0;
    for i := 1 to n do
        if x[i] + Suffix_Max > Global_Max then
            Suffix_Max := Suffix_Max + x[i];
            Global_Max := Suffix_Max;
            Global_Start_Index := Suffix_Start_Index;
            Global_End_Index := i;
        else if x[i] + Suffix_Max > 0 then
            Suffix_Max := Suffix_Max + x[i];
        else
            Suffix_Max := 0
            Suffix_Start_Index := i + 1;
end
```

□

5. In a history exam problem, the students are asked to put several historical events into chronological order. Students who order all events correctly will receive full credit. Partial

credits are awarded according to the longest (not necessarily contiguous) subsequence of events that are in the correct order relative to each other. Your task is to design an algorithm that determines the length of such a subsequence for the answer given by a student. Assume you already have a procedure that can find the longest *increasing* subsequence of a given sequence of distinct integers. Utilize the assumed procedure to obtain the needed algorithm.

Solution. Assume that no two historical events happened "at the same time," i.e., every two events can be given a strict chronological order.

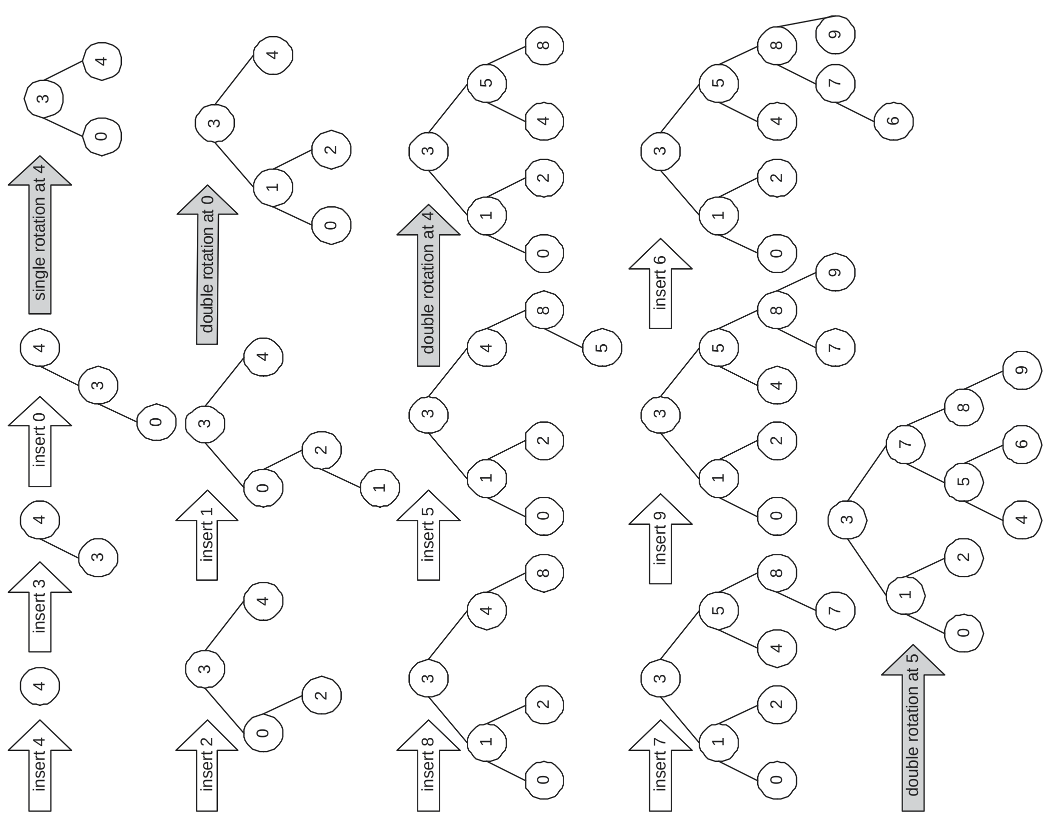
Step 1: Assign to each event e an integer n_e according to the correct chronological order of the historical events, i.e., $n_{e_1} < n_{e_2}$ if event e_1 happened before e_2.

Step 2: Feed the sequence of integers corresponding to a student's answer into the known procedure for determining the longest *increasing* subsequence.

Step 3: Compute and return the length of the subsequence obtained in Step 2. □

6. Show all intermediate and the final AVL trees formed by inserting the numbers 4, 3, 0, 2, 1, 8, 5, 7, 9, and 6 (in this order). Please use the following ordering convention: the key of an internal node is larger than that of its left child and smaller than that of its right child. If a rotation is performed during an insertion, please also show the tree before the rotation. (15 points)

Solution. (Chen, Po-An)



□

7. Apply the quicksort algorithm to the following array. Show the contents of the array after each partition operation. Please briefly describe your partition algorithm if it is different from the one we discussed in class.

7	1	5	11	2	10	9	3	6	12	4	8

Solution. (Chen, Po-An)

7	1	5	11	2	10	9	3	6	12	4	8
3	1	5	4	2	6	7	9	10	12	11	8
2	1	3	4	5	6	7	9	10	12	11	8
1	2	3	4	5	6	7	9	10	12	11	8
1	2	3	4	5	6	7	9	10	12	11	8
1	2	3	4	5	6	7	9	10	12	11	8
1	2	3	4	5	6	7	8	9	12	11	10
1	2	3	4	5	6	7	8	9	10	11	12
1	2	3	4	5	6	7	8	9	10	11	12

□

8. We have discussed in class how to rearrange an array into a (max) heap using a bottom-up approach. Please present the approach in pseudocode. (15 points)

Solution.

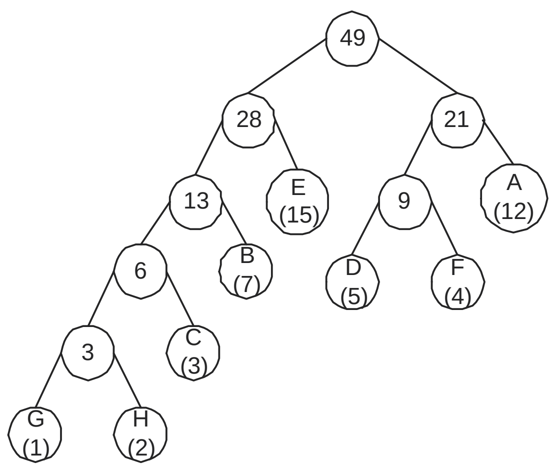
```
Algorithm Build_Heap(A,n);
begin
   for i := n DIV 2 downto 1 do
      parent := i;
      child1 := 2*parent;
      child2 := 2*parent + 1;
      if child2 > n then child2 := child1;
      if A[child1]>A[child2] then maxchild := child1
      else maxchild := child2;
      while maxchild<=n and A[parent]<A[maxchild] do
         swap(A[parent],A[maxchild]);
         parent := maxchild;
         child1 := 2*parent;
         child2 := 2*parent + 1;
```

```
            if child2 > n then child2 := child1;
            if A[child1]>A[child2] then maxchild := child1
            else maxchild := child2;
            end;
      end;
end;
```

□

9. Draw a Huffman tree for a text with the following frequency distribution: $A : 12$, $B : 7$, $C : 3$, $D : 5$, $E : 15$, $F : 4$, $G : 1$, and $H : 2$.

Solution. (Chen, Po-An)



□