Data Structure and Algorithm I

Midterm Examination

120 points

Time: 9:10am-12:10pm (180 minutes), Friday, November 12, 2010

Problem 1. In each of the following question, please specify if the statement is **true** or **false**. If the statement is true, explain why it is true. If it is false, explain what the correct answer is and why. (40 points. For each question, 2 points for the true/false answer and 3 points for the explanations.)

1. $n^{1.5} = O(n \log n)$.

   Ans:Flase

   Assume that $n^{1.5} = O(n \log n)$ i.e. there exist positive constants n_0 and c such that $n^{1.5} \leq c(n \log n)$ for all $n \geq n_0$, then we can get $\frac{n^{1.5}}{n \log n} \leq c$. But there is no constant greater than $\frac{n^{1.5}}{n \log n}$. So the assumption leads to contradiction.

2. A complete binary tree with a height of h can have more nodes than a full binary tree with a height of h.

   Ans:Flase

   A complete binary tree with a height of h have 2^{h-1} to $2^h - 1$ nodes.

   A complete binary tree with a height of h have $2^h - 1$ nodes.

3. A stack follows a FIFO (first-in-first-out) rule.

   Ans:False A stack follows a FILO (first-in-last-out) rule.

4. When we use a max heap to implement a priority queue, the time complexity of both the add and delete operations are $O(n)$.

   Ans:True

   The operation *add()* and *delete()* spend O(*the height of min heap*).

   Because the min heap is a complete tree, *the height of min heap* $\leq \log n$.

   $\Rightarrow$ the time complexity of $add()$ and $delete() = O(\log n) = O(n)$

5. $T(n) = T(n-1) + n, T(1) = 1$. Then $T(n) = O(n^3)$.

   Ans:True

```
  T(n)  =T(n-1)+n
  T(n-1)=T(n-2)+n-1
  ...
+ T(2)  =T(1)  +2
---------------------------
  T(n)=1+2+...+n=n(n+1)/2
```

   Then we get $T(n) = O(n^2) = O(n^3)$.

6. In a circular doubly linked list with 10 nodes, we will need to change 4 links if we want to delete a node other than the head node.

   Ans:False

   We can change the link fields of the nodes that precede and follow the node we want to delete.

   So we just need to change 2 links if we want to delete a node other than the head node.

7. If $f_1(n) = \Omega(g_1(n))$ and $f_2(n) = \Omega(g_2(n))$, then $f_1(n) \times f_2(n) = \Omega(g_1(n) \times g_2(n))$.

   Ans:True

   $f_1(n) = \Omega(g_1(n))$

   $\Rightarrow$ there exist positive constants n_1 and c_1 such that $f_1(n) \geq c_1 g_1(n)$ for all $n \geq n_1$

   $f_2(n) = \Omega(g_2(n))$

   $\Rightarrow$ there exist positive constants n_2 and c_2 such that $f_2(n) \geq c_2 g_2(n)$ for all $n \geq n_2$

   So we can find positive constants $c = c_1 c_2$ and $n_0 = max\{n_1, n_2\}$ such that

$f_1(n) \times f_2(n) \geq cg_1(n) \times g_2(n)$ for all $n \geq n_0$.
Then $f_1(n) \times f_2(n) = \Omega(g_1(n) \times g_2(n))$.

8. When using linked list to perform insertion sort, each time we remove an element from the input list and insert it to the correct position in the linked list. Assume that we have n numbers to be sorted, the time complexity for sorting these numbers using the insertion sort algorithm is $O(n^2)$.

   Ans:True

   The worst case is that we have to search all numbers in the list before we insert a new number.

   $\Rightarrow 1 + 2 + .. + n = n(n-1)/2$

   $\Rightarrow$ The time complexity for sorting these numbers using the insertion sort algorithm is $O(n^2)$.

Problem 2. "Fill the blank". (20 points. 5 points for each question.)

1. Please compute The failure function (in Knuth, Morris, and Pratt's pattern matching algorithm) for the pattern string "abaabaab" in Table 1.

2. Please draw all the threads of the threaded tree shown in Figure 7.

3. Please complete Table 2 to show the progress of converting the infix expression "2+1-(4-3*1)*3" to its postfix expression using a stack.

4. Please complete the following function to concatenate two circularly singly linked list into one circularly singly linked list.

```
typedef struct listNode * listPointer;
typedef struct listNode {
   int data;
   listPointer link;
};
```

```
listPointer concatenate(listPointer list1, listPointer list2) {

/* Produce a new list which contains list1 followed by list2.
   Return the pointer which points to the new list.
   The arguments list1 and list2 point to the tails of the lists.*/

   listPointer temp;

   if (list1==NULL)
    return list2;
   if (list2==NULL)
    return list1;

   listPointer tmp = list1->link;
   list1->link = list2->link;
   list2->link = tmp;

   return list2;
}
```

Problem 3. The Tower of Hanoi is a mathematical game. The game consists of 3 rods, and n disks of different sizes which can slide onto any rod. The puzzle starts with the disks stack in ascending order of size on rod number 1 (the smallest disk is on the top). The objective of the game is to move the entire stack to rod number 3, obeying the following rules:

1. Only one disk can be moved at a time.
2. Each move consists of taking the upper disk from one of the rods and sliding it onto another rod, on top of the other disks that may already be present on that rod.
3. No disk may be placed on top of a smaller disk.

In this problem, please write an algorithm using the recursive technique to output a series of moves which solve the Tower of Hanoi problem. The input and output are as follows:

Input: one single number which represents the number of disks on the rod

Output: Each line of output consists of a pair (i, j), which represents a move from rod number i to rod number j. The sequence of moves should move the entire stack from rod number 1 to rod number 3.

Example:

Input:

3

Output:

(1,3)

(1,2)

(3,2)

(1,3)

(2,1)

(2,3)

(1,3)

(10 points)

Ans:

```
hanoi(from, tmp, to, n){
  if(n==1){
          move(from,to);
          return;
  }
  hanoi(from, to, tmp, n-1);
  printf ( "(%d,%d)", from, to );
```

```
    hanoi(tmp, from, to, n-1);
    return;
  }
```

Problem 4. A *lower triangular matrix* is one in which all elements above the main diagonal of a square matrix are zero. Assume that we have a lower triangular matrix L with n rows (Figure 2). Please answer the following related questions: (10 points)

1. What is the total number of nonzero terms of L (3 points)? If each element can be stored in a 32-bit signed integer, how much memory would be needed to store all the nonzero elements of matrix L (2 points)?

   Ans:

   (a) $1+2+\ldots+n = \frac{n\times(n+1)}{2}$

   (b) $32 \times \frac{n\times(n+1)}{2} = 16(n^2+n)$ bits.

2. Since storing a triangular matrix as a two dimensional array wastes sapce, we would like to find a way to store only the nonzero terms in the triangular matrix. Find an addressing formula for the elements $L_{i,j}$ to be stored in a one dimensional array a. In other words, where in the array a would you store the element $L_{i,j}$? (You only need to explain how to map each nonzero element to the one dimensional array. No need to write down the actual function.) (5 points)

   Ans:

   Pack the traiagular matrix into array a row by row like the following Fig 1

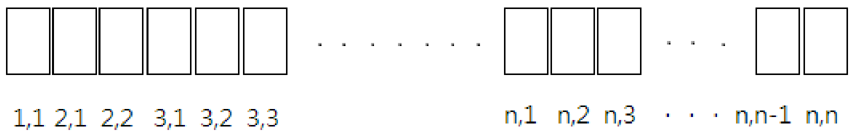

Figure 1: Stored into array a

$$1+2+\cdots+(i-1)+j-1 = \frac{(1+i-1)\times(i-1)}{2}+(j-1) = \frac{i^2-i}{2}+(j-1)$$

The minus one is because we store the first element at a[0].

Problem 5. In the following questions, consider the list of numbers:
$62, 31, 70, 91, 25, 11, 9, 61, 73, 6$. (20 points)

1. Show the result of inserting the numbers in the list in the same order specified above into an initially empty minimum heap. Note that you need to show how the heap looks like after each number is inserted. (5 points)

   Ans: Here shows the 3 middle trees of the results. see Figure 3.

2. Show the result of inserting the numbers in the list in the same order specified above into an initially empty binary search tree. Note that you need to show how the binary search tree looks like after each number is inserted. (5 points)

   Ans: Here shows the 3 middle trees of the results. see Figure 4.

3. Use the binary search tree you created in question 2. What are the two possible binary search trees after 62 is deleted? (5 points)

   Ans: see Figure 5.

4. Explain how you can utilize a minimum heap to sort the list of numbers in descending order. (3 points) Let n be the number of elements in the list. What is the time complexity of your sorting algorithm? (2 points)

   Ans:

   (1)

   step1: Delete the root from the min heap and put it into stackA. *O(1)*

   step2: Restructure the min heap. $O(\log n)$

   step3: Go step1 until the min heap is empty. *n times*

   step4: Pop items from stackA.(The result is in descending order.) *O(n)*

   (2) $O(n \log n)$

Problem 6. In this problem, we consider logic expressions with the following operands and operators:

Operands: 0 and 1, which represents false and true, respectively.

Operators: & (and), | (or), and ! (not).

Note that you also have to consider left and right parenthesis. The precedences of the three operators are ! > & > |. (25 points)

1. Draw the logical expression tree of the expression !(0&!1&0|0)&0. Since ! (not) is an unary operator, we ask you to put its only operand to its right child. (3 points) Write down its prefix expression. (2 points)

   (a) See Figure 6

   (b) Prefix: & ! | & & 0 ! 1 0 0 0

2. Please write down an algorithm to take a string of logic expression and convert it to an expression tree. You can use any data structure you like to represent the tree. Since ! (not) is an unary operator, we ask you to put its only operand to its right child. However, please explain how your data structure relate to the tree logically in details. (Comment: this is **the hardest question** in this exam. Do this after you finish all other questions!) (10 points)

   (a) Observe that the higher precedence the operator is, the lower level the corresponding tree node will be.

   (b) Define *isp*[] as in-stack-precedence and *icp*[] as incoming-precedence.

   (c)
```
typedef struct ExpNode{
ExpNode *left, *right;
op v;
/*if the node represent digit, v is the value; if not, v is the operator*/
};
int icp[] = {7, 1, 5, 4, 3}; /*( ) ! & |*/
int isp[] = {2, 1, 5, 4, 3}; /*( ) ! & |*/

ToExpTree (Exp input){
  Create Stack1 for operator;
  Create Stack2 for ExpNode;
```

```
//read the string left to right
while(end of input is not reached)
{
  read from input, store at v1
  if(v1 == digit)
  {
  create ExpNode k for v1;
  push k into Stack2;
  }else /*v1 is & | ! ( )*/
  {
      if(Stack1 is empty || the icp of v1 > isp of Stack1[top])
          push v1 into Stack1;
      else
      {
          while(the icp of v1 <= isp of Stack1[top])
          {
  if(Stack1[top] is "(")
  {
       pop Stack1;
       break;
  }
  op1 = pop Stack1;
  creat ExpNode t for op1;
  r = pop Stack2;
  t->right = t;
  if(op1 is & or |)
  {
      l = pop Stack2;
      t->left = l;
  }
```

```
    push t into Stack2;
}
push v1 into Stack1;
            }
         }
    }
  }
```

3. Please write down an algorithm to evaluate the outcome of the logical expression using the tree generated by your algorithm in question 1. (5 points)

   (a) Postorder traversal

   (b)
```
typedef struct ExpNode{
ExpNode *left, *right;
int type; /*digit or operator*/
int value;
};
cal(ExpNode *tree){
  if(!tree){
   cal(tree->left);
   cal(tree->right);
   if(tree->type is equal to operator & |)
   tree->value = (tree->left->value) op (tree->right->value);
   else if(tree->type is equal to operator !)
   tree->value = !(tree->right->value);
  }
}
```

4. Please write down an algorithm to output the prefix expression of the logic expression using the tree generated by your algorithm in question 1. (5 points)

   (a) Preorder traversal

(b)
```
typedef struct ExpNode{
ExpNode *left, *right;
op v;
/*if the node represent digit, v is the value; if not, v is the operator*/
};
pre(ExpNode *tree){
  if(!tree){
   printf(tree->v);
   pre(tree->left);
   pre(tree->right);
  }
}
```

Problem 7. Please provide constructive suggestions to this course in terms of lectures, homeworks, or any other aspects. (5 points)

$$\mathbf{L} = \begin{bmatrix} l_{1,1} & & & & 0 \\ l_{2,1} & l_{2,2} & & & \\ l_{3,1} & l_{3,2} & \ddots & & \\ \vdots & \vdots & \ddots & \ddots & \\ l_{n,1} & l_{n,2} & \dots & l_{n,n-1} & l_{n,n} \end{bmatrix}$$

Figure 2: A Lower Triangular Matrix

Table 1: The failure function for the KMP algorithm

b	a	b	c	b	c	b	a	b	c	b	a	b	c
-1	-1	0	-1	0	-1	0	1	2	3	4	1	2	3

Table 2: The progress of converting the infix expression “2+1-(4-3*1)*3” to its postfix expression

Token	**Stack content (bottom to top)**	**Output so far**
2		2
+	+	2
1	+	21
-	-	21+
(	-(	21+
4	-(	21+4
-	-(-	21+4
3	-(-	21+43
*	-(-*	21+43
1	-(-*	21+431
)	-	21+431*-
*	-*	21+431*-
3		21+431*-3*-

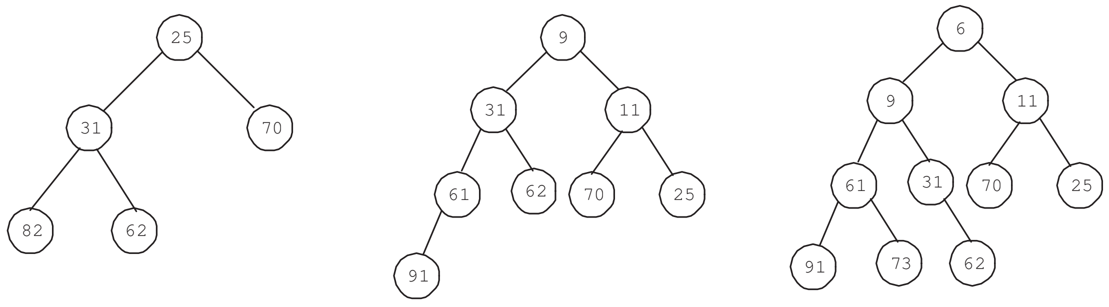

Figure 3: 5-1

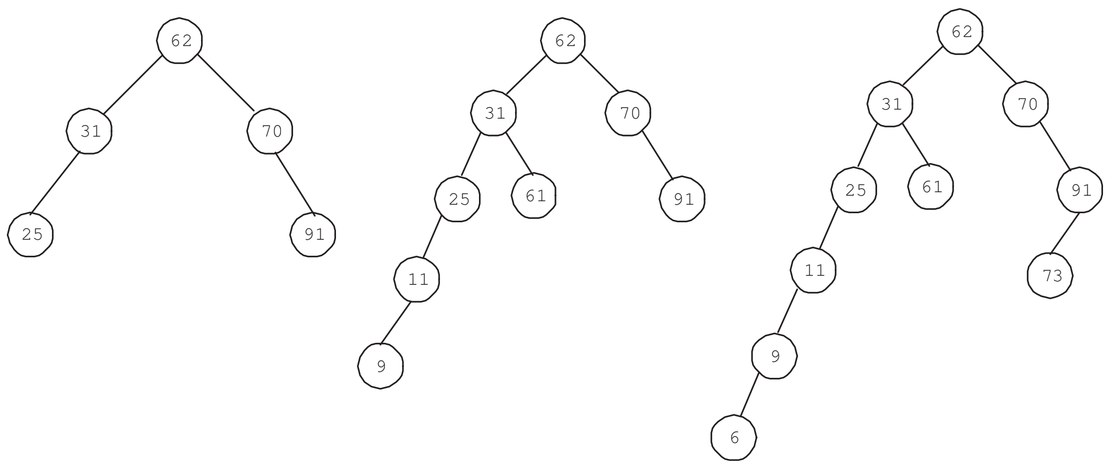

Figure 4: 5-2

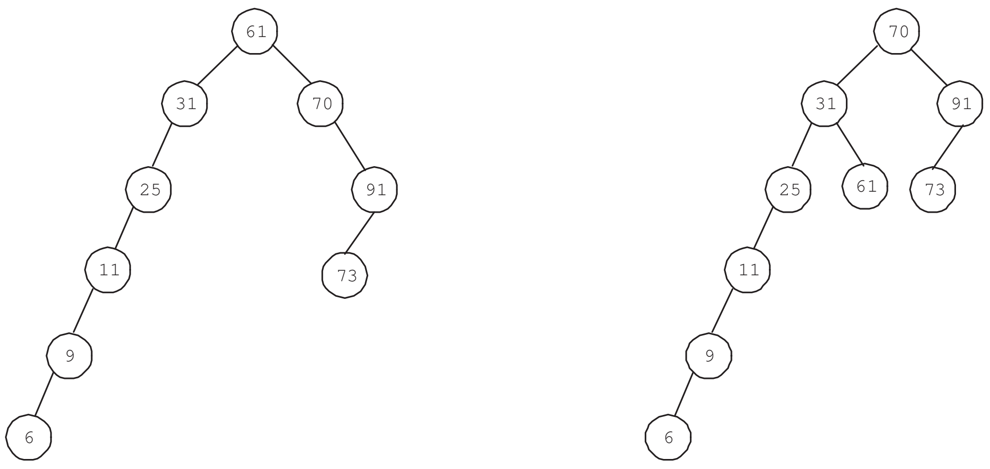

Figure 5: 5-3

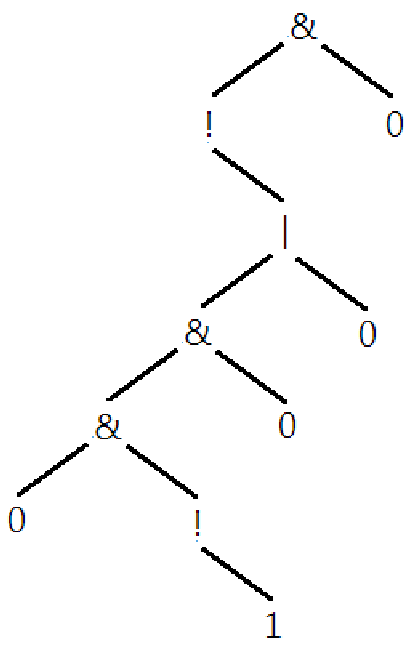

Figure 6: Expression Tree

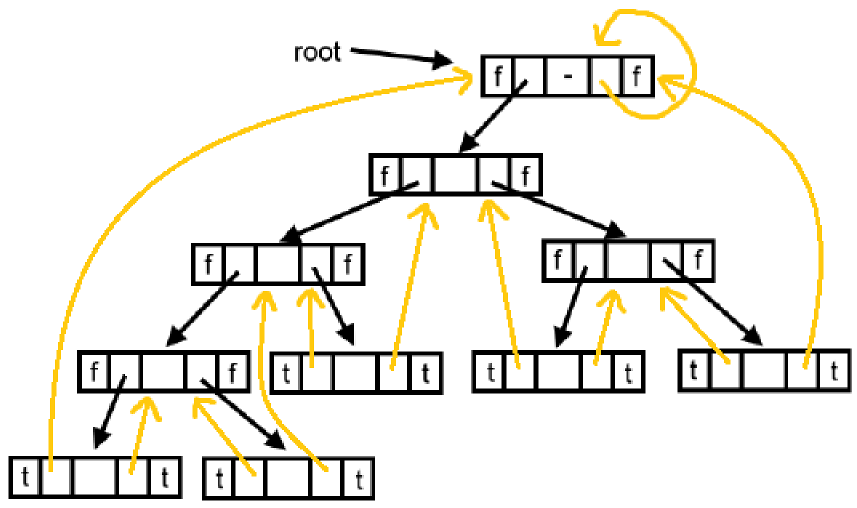

Figure 7: Memory representation of a threaded tree