NANYANG TECHNOLOGICAL UNIVERSITY

SEMESTER 1 EXAMINATION 2015-2016

EE2008 / IM1001 – DATA STRUCTURES AND ALGORITHMS

November / December 2015 Time Allowed: 2½ hours

INSTRUCTIONS

1. This paper contains 4 questions and comprises 3 pages.
2. Answer ALL questions.
3. All questions carry equal marks.
4. This is a closed-book examination.

---

1. (a) Consider the following algorithm:

Input: n (a non-negative integer), $A[0]$, $A[1]$, $A[2]$, … , $A[n]$ (an array of integers).

```
for $i = 0$ to $n$
    for $j = i+1$ to $n$
        if ( $A[i] + A[j] == k$ )
            return true
return false
```

(i) What does the algorithm compute?

(ii) Express the worst-case running time of the algorithm using the Big-Oh notation. Give details of your working.

(iii) If it takes 10 seconds to run the algorithm on an array of 10000 elements, how long will it take to run the algorithm on an array with 40000 elements?

(8 Marks)

Note: Question No. 1 continues on page 2.

(b) Prove or disprove each of the following statements.

(i) $\log n! = O(n\log(n))$

(ii) If $f(n)$ is $O(g(n))$, then $g(n)$ is $O(f(n))$

(6 Marks)

(c) (i) A pointer *start* points to the first element of a singly-linked list L. Write an algorithm that inserts a new node (with data field *val*) at the beginning of the linked list L and returns *start*. What is the worst-case running time of your algorithm?

(ii) A priority queue is implemented using an array. Items stored in the array are ordered by the values of their keys in non-decreasing order. Write an algorithm to insert a new item with a specified key into the array.

(11 Marks)

2. (a) (i) Solve the following recurrence relation: $a_n = 2a_{n-1} + 1$, $a_1 = 1$.

(ii) Using pseudo-code, describe the implementation of the method *isFirst(p)* of the LIST ADT. Assume that the LIST ADT is implemented using a doubly linked list.

(9 Marks)

(b) Draw the 8-item hash table resulting from hashing the keys 1055, 1492, 1776, 1812, 1918, and 1945 with the hash function $h(x) = (5x) \bmod 8$. Assume that collisions are handled by linear probing.

(6 Marks)

(c) Write a recursive algorithm that checks whether the largest value in an array A is larger than a given value x.

(10 Marks)

3. (a) Write an algorithm that reverses the left most node and the right most node in a binary tree. Assume that both left and right subtrees of the root T in the binary tree are not empty, and each node contains left, right, parent and value information.

(8 Marks)

(b) Write an algorithm which can sort an array A in increasing order:

(i) Assume that all the elements of the array are unique.

(ii) The main sorting strategy is as follows: check the elements in the array and reverse $A[i]$ and $A[i+1]$ if $A[i]>A[i+1]$.

(iii) Repeat part (ii) until such reversal is no longer required.

(10 Marks)

(c) Write a **divide-and-conquer** algorithm that finds the smallest value in a maxheap.

(7 Marks)

4. (a) When both depth-first-search and preorder algorithms are used to traverse a binary tree, will the same vertex visiting orders be obtained? Justify your answer. If it is true, prove it; otherwise, give a counter example.

(8 Marks)

(b) Write an algorithm that computes the sum of the weights of all edges in a weighted undirected graph represented by an adjacency list *adj*.

(12 Marks)

(c) Assume that in a connected undirected graph G, the weights of all edges incident to a vertex v are unique. Is the edge with the second smallest weight that is incident to vertex v included in all minimum spanning trees of G? If yes, prove it; otherwise, give a counter example.

(5 Marks)

END OF PAPER

Q1. (a) (i) The algorithm checks whether there 2 number in array A[n] that their sum equals to k. If exists, return ture. Else, return false.

(ii) The worst case is that the sum of last two integer is k.

$$T(n) = \sum_{i=0}^{n} \sum_{j=i+1}^{n} 1$$

$$= \sum_{i=0}^{n} n-i$$

$$= \frac{(n+0)\times(n+1)}{2} = \frac{n^2+n}{2} \le kn^2$$

$$\therefore T(n) = O(n^2)$$

(iii) $\because T(n) = O(n^2)$

When $n = 10000$, $T(n) = 10\,s$

If $n = 40000$,

$$T(n) = O(40000^2) \approx 16\,O(10000)^2 = 160\,s$$

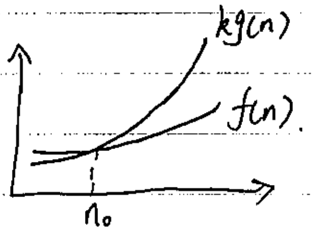
(b)(i) $\log n!$

$= \log 1\times2\times3\times\cdots\times n$

$= \underbrace{\log 1 + \log 2 + \log 3 + \cdots + \log n}_{n} \le \underbrace{\log n + \log n + \cdots + \log n}_{n} = n\log n$

$\therefore \log n! = O(n\log n)$ is correct.

(ii) If $f(n) = O(g(n))$, then there are constant k and integer n_0 such that for $n \ge n_0$, $f(n) \le kg(n)$



Suppose $f(n) = n$, $g(n) = n^2$

It is obvious that $f(n) = O(g(n))$

If $g(n) = O(f(n))$

we need to find sonstant k and an integer n_0 such that

for $n \ge n_0$, $g(n) \le kf(n)$

$\Rightarrow n^2 \le kn$

$(n^2 - kn) \le 0 \Rightarrow n(n-k) \le 0 \quad \therefore 0 < n \le k$,

$\therefore$ no such n_0 exist.

$\therefore$ It is wrong

Q1 (c) (i)

```
insert (val, start) {
    ~~if (start != null) {~~
        temp = new node
        temp. data = val
    if (start = null) {
        temp.next = null
        start = temp }
    else {
        temp.next = start
        start = temp }
    return start }
```

The worst-case running time is $T(n) = 1$

(ii)

```
insert ( k, e) {                          // A[n] ~~soto~~ stores element
  for (i=0 to n ) {                       // B[n] stores key
    if (k ≥ B[i]){
      temp element = A [i+1];
      temp key = B[i+1];
      temp = i+1 ; }
    for (j = i+1 to n ) {
      A[j+1] = A[j] ;
      B[j+1] = B[j];
    }
    A[temp] = e ;
    B[temp] = k ;
  }
```

Q2 (a) (i) $a_n = 2a_{n-1}+1$

$= \cancel{4}2(2a_{n-2}+1)+1 = 2^2 a_{n-2}+1+2$

$= 4(2a_{n-3}+1)+1+2 = 2^3 a_{n-3}+1+2+2^2$

$= \cdots$

$= 2^{n-1}a_1 + 1+2+2^2+\cdots+2^{n-2}$ $\quad \because a_1 = 1$

$\therefore a_n = 1+2+2^2+\cdots+2^{n-1} = 2^n - 1$

(ii)

```
isFirst(p) {
    if (p.prev == null)
        return ture
    else
        return false
}
```

(b) $h(1055) = 5 \times 1055 \bmod 8 = 3$ $\quad$ $h(1918) = 5 \times 1918 \bmod 8 = 6$

$h(1492) = 5 \times 1492 \bmod 8 = 4$ $\quad$ $h(1945) = 5 \times 1945 \bmod 8 = 5$

$h(1776) = 5 \times 1776 \bmod 8 = 0$

$h(1812) = 5 \times 1812 \bmod 8 = 4$

1776			1055	1492	1812	1918	1945	
0	1	2	3	4	5	6	7	8

(c) Suppose ~~A~~ array A[n], let m = x ~~≠~~.

```
max(n) {
  if (n>=0) {
    if (m < A[n]) {
      m = A[n];
      return max(n-1) }}
    if (x < m)
      return ture
    else
      return false
}
```

Q3. (a)

```
reverse LR(T) {
    L = leftmost (T);
    R = rightmost (T);
    temp = L ;
    temp. parent = L. parent;
    temp. right = L. right;
    L = R ;
    L. parent = R. parent ;
    L. left = R. left;
    R = temp;
    R. parent = temp. parent;
    R. right = temp. right;
}
leftmost (T){
    if T. left ! = null
        return leftmost (T. left) }
rightmost (T) {
    if T. right ! = null
        return rightmost (T. right) }
```

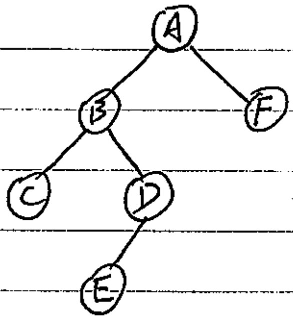
(b)

```
Sort (A) {
    for (i=0 to~~a~~ n) {
        for (j= 0 to n)
            if (A[i] > A[j] ) {
                temp = A[i] ;
                A[i] = A[j] ;
                A[j] = temp;
            }
    }
}
```

A [n] is the maxheap array

```
Q3(c). Min in Maxheap ( A ) {
          i = $\frac{n}{2}$
          if (i==0)
             return A[i]
          min = A[i+1]
          for ( count= i+2 to n )
             if ( A[count]< min)
                min=A[count]
       }
```

Q4(a) It is wrong. Suppose the binary tree is



Apply preorder, there is only one possibilty

A → B → C → D → E → F

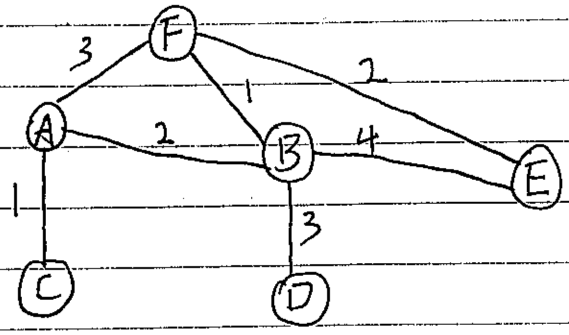
But apply DFS, there are multiple answers,

① A → B → C → D → E → F same as preorder

② A → B → D → E → C → F different. so is wrong.

```
(b) ~~sum~~ totalweight (adj) { sum=0;
          for (i=1 to n)
             sum = sum + weight(adj[i]);
          totalweight = sum/2 ;
       }
       weight (adj[i]) {
          ~~fo~~ if(adj[i].next != null)
                 return Weight + weight (adj[i].next)
             return 0; }
```

Q4(c). It is wrong. If we have this graph



Suppose the vertex v is A, the minimum spanning tree is

AC, AE, BF, BD, EF

which is $1+3+1+3+2=10$ and it doesn't include AB which is the second smallest weight ~~of~~ incident to v.