【不可使用計算機】＊作答前請先核對試題、答案卷(試卷)與准考證之所組別與考科是否相符！！

1. (5%) In the Knuth-Morris-Pratt string-pattern matching algorithm, we need a failure function. Compute the failure function for the following pattern.

index	0	1	2	3	4	5	6	7	8
pattern	a	b	a	a	b	a	a	a	a
failure									

Figure 1 Pattern matching

2 (5%) In a binary tree, there may be many null pointers. We may use these pointers to point to in-order predecessors and successors. These pointers are called thread pointers. Draw all the thread pointers in the following binary tree.

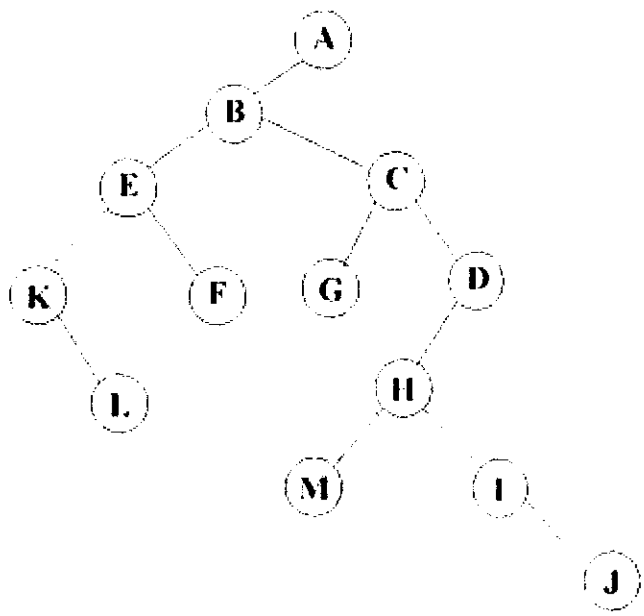


Figure 2 Threaded binary tree

3 (5%) Assume there are only variables such as a, b, c, … and binary operators, such as +, -, *, and / in an arithmetic expression. The expression is fully parenthesized, that is, there is a pair of parentheses for each operator. Examples are (((a+b)-c)/d), ((a+(c-(b-a))/d)+f), ((((a+b)+c)+d)+e), etc.

Write a program that translated a fully-parenthesized expression into its prefix form. The input is a correct, fully-parenthesized arithmetic expression, stored in a read-only array. This program can use 3 queues together with the usual stack pointers. You cannot use additional storage, such as arrays, except a fixed number of simple temporary variables. In the program, you can only push, pop, examine the contents of a cell, and compare two contents of two cells for equality. The program can scan the input from left to right only once. It cannot store the input expression somewhere else for repeated examinations. You can use C or Java or Pascal or other similar programming languages to write this program. Your program should clearly show the underlying algorithm. Minor details and errors in the program will be tolerated. In particular you need to explain your data structures with examples.

4 (5%) Assume each node contains a data, lthread, rthread, lchild, and rchild fields. lthread and rthread are Boolean fields. lchild and rchild are pointers. If lthread is true, the lchild field will be considered as a left thread; otherwise, lchild will point to the left child. Similarly for the rthread and

rchild fields. Consider the following algorithm for pre-order traversal of a binary tree using thread pointers:

```
1.  presucc(node) {
2.      if node→lthread = false
3.          then return node→lchild; /*has left child*/
4.      if node→rthread = false
5.          then return node→rchild; /*has right child*/
6.      _______________ /* in-order successor */
7.      /* find the in-order successor of the in-order successor
of the node's in-order successor */
8.      while node→rthread = true do
9.          node := node→rchild;
10.     return node→rchild;
11. }
```

Figure 3 Tree traversal algorithm

Please fill in the missing line 6 with an assignment statement.

5 (5%) This problem contains five yes-no questions. Each question counts as 1 point. You do not need to explain your answers.

(a) $n^{2^n} + 5 \cdot 3^{2n} = \Theta(n^{2^n})$.

(b) $3n^{1.0001} + 99n \log n = O(n^{1.1})$.

(c) $n \log(2n) = O(n \ln n)$.

(d) $n^2 \log(5n) = \Theta(n^2)$.

(e) $n^{0.999} \log n = \Theta(n^2)$.

【不可使用計算機】*作答前請先核對試題、答案卷(試卷)與准考證之所組別與考科是否相符！！

No partial credits will be given for question 6 ~ question 10. You either answer all the sub-questions correctly and get full credits or get 0 credits for the whole question.

6. (5%) You have the following data structures **A. *list***, **B. *hash table***, and **C. *AVL tree*** at your disposal. Assume you have **a lot of data** in the form of key-value pairs <key1,value1>, <key2, value2>, …,<keyN,valueN>. Please answer the following questions by **making your choices from the three data structures**:

(1). If your application requires frequent random access (i.e. retrieving the value for a given key), using the data structure ________ to store the data will give the longest average access time.

(2). Assume there is a total ordering of data, and your application requires frequent range queries to the data (i.e. retrieving data with keys in the range [lower_bound, upper_bound]). Range query to your data can be efficiently implemented with the data structure ________.

(3). If we want to emulate a priority queue, which of the three data structures will be the best choice (i.e. both element addition and extract-max have to be efficient) ________.

7. (5%) Figure 4 shows the two tree rotation operations commonly used for balancing binary search trees.

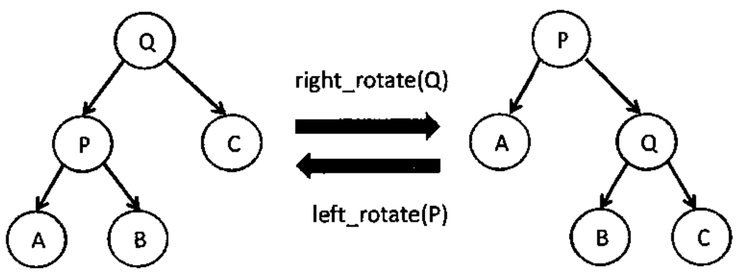


Figure 4. Tree rotations

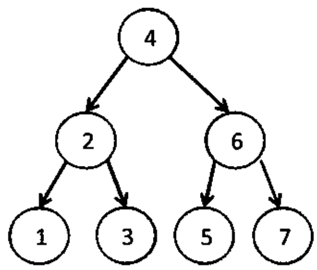


Figure 5. Binary search tree *T* to be rotated

Given the binary search tree *T* in Figure 5, let's apply the following tree rotations in sequence on *T*:

left_rorate(6) → right_rorate(4) → left_rorate(4)

(1). After the tree rotations, the parent of node 4 will be node ___________

(2). After the tree rotations, the parent of node 7 will be node ___________

Let's apply the following additional tree rotations in sequence

left_rorate(1) → right_rorate(4) → left_rorate(6) → right_rorate(3).

(3). After the seven tree rotations are applied to *T*, an in-order traversal of *T* will output a sequence of the tree nodes as ______________

8. (5 %) We have an open addressing hash table of size 7 to store integer keys, with hash function h(x) = x mod 7. Assume we use linear probing for collision resolution and insert elements in the order 1, 15, 14, 3, 9, 5, 27.

(1). Draw the hash table after the insertions ___________
(2). What is the worst-case time complexity for searching in a hash table of size N __________
(3). What is the best-case time complexity for searching in a hash table of size N ___________

```
void swap(int &v1, int &v2)
{
    int t;
    t = v2;
    v2 = v1;
    v1 = t;
}

void partition(int data[], int size, int pivot_v, int& low, int& high) {
    low = -1;
    high = size;

    for (int i = 0; i < high;) {
        if (data[i] > pivot_v) {
            swap(data[i], data[++low]);
            ++i;
        } else if (data[i] < pivot_v) {
            swap(data[i], data[--high]);
        } else {
            ++i;
        }
    }
}

void xsort(int data[], int size)
{
    int pivot, low, high;

    if ( size <=1 )
        return;

    pivot = size/2;

    partition(data, size, data[pivot], low, high);
    xsort(data, low+1);
    xsort(data+high, size - high);
}
```

Figure 6. xsort function for sorting an integer array

9. (5 %) Figure 6 presents the code for sorting an integer array. Assume that we make a call to the `xsort(...)` function with

```
int data[] = {2,0,1,3,2,4,1,0,0,1};
int size = 10;
```

(1). After the call to `xsort` returns, the content of the `data[]` array will be ________.

(You have to list all 10 elements of the `data[]` array sequentially, starting from `data[0]`, `data[1]`,..,etc.)

(2). How many times will the function `partition` be invoked? ______________ times.

(3). The worst-case time complexity of the `xsort` function for an input array `data[]` with `size` elements is ____________.

10. (5%) Figure 7 presents the code for computing the number of connected components in an undirected graph. In the code, the undirected graph is stored in the form of adjacency lists. The graph has N vertices, which correspond to `vertices[N]` at Line 25. Each vertex is a `tVertex` structure as given at Line 9~20. The function `ConnectUndirected(v1, v2)` is used for creating an undirected edge between vertex `v1` and vertex `v2`. The function `NumberOfComponents()` at Line 36 will return the number of connected components in the graph.

Please complete the code by filling the blanks in Figure 7. There are a total of **three** blanks at Line 30, 32, and 43 to be filled.

```
1   #define N 10
2
3   struct tListNode
4   {
5      int   v;
6      tListNode *pNext;
7   };
8
9   struct tVertex
10  {
11     bool visited;
12     tListNode* pAdjList;
13     tVertex() { pAdjList = 0;  visited = false;}
14
15     void Connect (int v)
16     {
17        tListNode *pNeighbor = new tListNode;
18        pNeighbor->v = v;
19        pNeighbor->pNext = pAdjList;
20        pAdjList = pNeighbor;
21     }
22
23  };
24
25  tVertex vertices[N];
26
27  void DFS (int v)
28  {
29     tListNode *pN;
30     vertices[v].visited = ____________;
31
32     for ( pN = vertices[v].pAdjList; pN ;______________)
33        if ( !vertices[pN->v].visited ) {  DFS( pN->v); }
34  }
35
36  int NumberOfComponents ()
37  {
38     int k;
39     int cnt = 0;
40
41     for ( k = 0; k < N; k++)    vertices[k].visited = false;
42     for ( k = 0; k < N; k++) {
43        if ( !vertices[k].visited ) { ____________  }
44     }
45     return cnt;
46  }
47
48  void ConnectUndirected(int v1, int v2)
49  {
50     vertices[v1].Connect (v2);  vertices[v2].Connect (v1);
51  }
```

Figure 7. Code for computing the number of connected components of an undirected graph

【不可使用計算機】＊作答前請先核對試題、答案卷(試卷)與准考證之所組別與考科是否相符！！

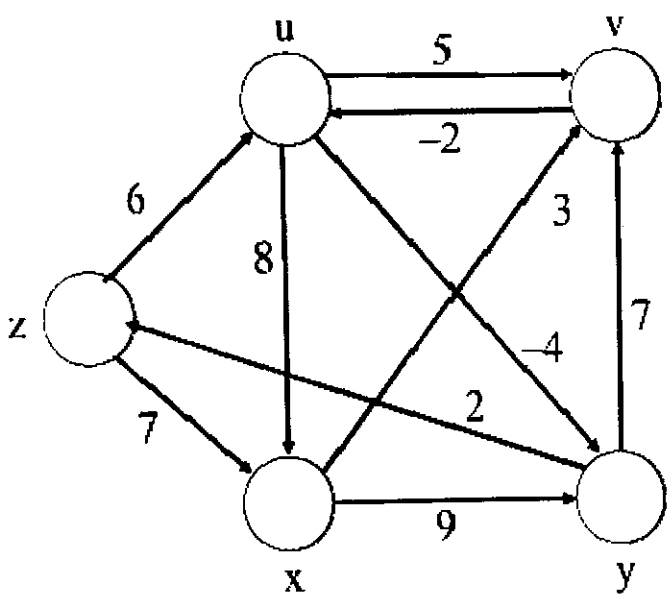
In the following problem 11-15, fill in the blanks. Please try to keep all 10 answers on the same sheet of the answer book. Each problem is worth 5 points. No partial credit will be given for only one correct blank. However you do not have to provide any justification.

11. (5%) Suppose that the amount of flow in the maximum flow from source to sink in the network G = (V, E) is f, and that all capacities are integral. Suppose $n = |V|$ and $m = |E|$. How long would it take at most to find this maximum flow using the Ford-Fulkerson algorithm? ________. Suppose that the minimum s-t cut in the network G has x arcs. Suppose that one adds y units of capacity to each arc of G, creating a transformed network G'. The capacity of the minimum cut of G' is at most ________.

12. (5%) The running time of Kruskal's algorithm for a connected undirected weighted graph G = (V, E) is ________. Suppose that all edge weights in a graph G are integers in the range from 1 to |V|. How fast can you make Kruskal's algorithm run? ________

13. (5%) If some of the edge weights in a graph are negative, the shortest path can be obtained using Dijkstra's algorithm by first adding a large constant C to each edge weight, where C is large enough that every resulting edge weight will be nonnegative (True or False). ________ The Bellman-Ford algorithm is not suitable if the input graph has negative-weight edges (True or False). ________

14. (5%) Given a graph as below. Show the weight on the edge between Node x and Node y after reweighting using Johnson's algorithm. ________ Show the fifth row of an adjacent matrix (with order of u, v, x, y, z) after using the Floyd-Warshall Shortest Paths algorithm with the intermediate Node u. ________



15. (5%) You are the program chair of a conference! Part of your job is to assign papers to reviewers. You have 6 papers P1, P2, P3, P4, P5, P6 and 3 reviewers R1, R2, R3. Initially, each reviewer constructs a list of papers he is willing to review as followings: R1 = {P1, P3, P5, P6}, R2 = {P1, P2, P4}, R3 = {P1, P2, P3, P4, P5, P6}. An assignment of papers to reviewers is valid if each paper is assigned to at least 2 distinct reviewers that are willing to review that paper. The maximum number of papers assigned to any one reviewer should not be greater than 4. You would like to maximize the total number of papers which are validly assigned. Please model this problem using a s-t flow network G = (V, E) (Draw a flow network with capacity labeling). ________ What is the maximum number of papers that can be validly assigned? ________

【不可使用計算機】＊作答前請先核對試題、答案卷(試卷)與准考證之所組別與考科是否相符！！

16. (5%) Consider the multiplication of four matrices with dimensions in the following order: 10×11, 11×25, 25×40, 40×2. Find the optimal parenthesization of the above product and the minimum number of scalar multiplications needed.

17. (5%) Consider the problem of finding a vector (x_1, x_2, x_3, x_4, x_5) satisfying the following constraints such that $x_1 + x_2 + x_3 + x_4 + x_5$ is maximized, where $x_i \le 0$ for i=1,…,5. What are the maximum value and the corresponding vector?

$x_1 - x_2 \le 1$,
$x_1 - x_5 \le -1$,
$x_2 - x_5 \le 1$,
$x_3 - x_1 \le 15$,
$x_4 - x_1 \le 4$,
$x_4 - x_3 \le -1$,
$x_5 - x_3 \le -3$,
$x_5 - x_4 \le 0$.

18. Consider a set of 6 nodes 1, 2, 3, 4, 5, 6 and their corresponding weights: 2, 3, 4, 4, 5, 6.
(a) (5%) Build a binary tree with these nodes appearing in the leaves such that the maximum of **w[i] × (1/2)^d[i]** is minimized, where w[i] is the weight and d[i] is the depth of node i in the tree. Note that the root has depth 0. What is the optimal value?
(b) (10%) Give a greedy algorithm for N nodes and explain your idea.