國立臺灣師範大學 112 學年度碩士班招生考試試題

科目：軟體基礎　　適用系所：資訊工程學系

注意：1.本試題共 6 頁，請依序在答案卷上作答，並標明題號，不必抄題。2.答案必須寫在指定作答區內，否則依規定扣分。

1. (9 points) The following C++ code defines an array to store the non-zero terms in a sparse matrix.

Sparse_Matrix Create(*max_row*, *max_col*) ::=

```
#define MAX_TERMS 101 /* maximum number of terms +1*/
typedef struct {
        int col;
        int row;
        int value;
        } term;
term a[MAX_TERMS];
```

Besides, **a[0].*value*** is used to store the number of non-zero terms in the sparse matrix. **a[0].*col*** and **a[0].*row*** are used to store the number of columns and the number of rows of the matrix, respectively. The non-zero terms in the sparse matrix are stored **a[*i*]** (***i*** = 1, …, **a[0].*value***) in row-major order.

(a) (6 points) The following sparse matrix is stored in **a[]**. Please show the results in **rowTerms[]**, **startingPos[]**, and **b[]** after performing the following transpose() function.

$$\begin{bmatrix} 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 5 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 10 & 0 & -5 & 0 & -2 \\ 0 & 4 & 0 & 0 & 0 & 6 \end{bmatrix}$$

```
void transpose(term a[], term b[])
{   /* perform transpose on a and output to b */
    int rowTerms[MAX_COL], startingPos[MAX_COL];
    int i,j, numCols = a[0].col, numTerms = a[0].value;
    b[0].row = numCols;
    b[0].col = a[0].row;
    b[0].value = numTerms;
    if (numTerms > 0)   {
        for (i = 0; i < numCols; i++)
            rowTerms[i] = 0;
        for (i = 1; i <= numTerms; i++)
            rowTerms[a[i].col]++;
        startingPos[0] = 1;
        for (i = 1; i < numCols; i++)
            startingPos[i] = startingPos[i-1] + rowTerms[i-1];
        for (i = 1; i <= numTerms; i++)   {
            j = startingPos[a[i].col]++;
            b[j].row = a[i].col;
            b[j].col = a[i].row;
            b[j].value = a[i].value;
        }
    }
}
```

(b) (3 points) What is the time complexity of performing the above transpose() function if the sparse matrix is a $m \times n$ matrix with k non-zero terms?

2. (10 points) The following C++ code is used to insert a node to the front of a circular linked list. Besides, ***last** is a pointer points to the last node of the circular linked list. **node** is a pointer points to the inserted node. If the circular linked list is empty, ***last** is **null**.

(a) (6 points) Please give program instructions filled in the blanks to complete the code.

```
typedef struct list_node *list_pointer;
typedef struct list_node {
    char data [4];
    list_pointer link;
    };
void insertFront (listPointer *last, listPointer node)
{ if (! (*last)){
            *last = node;
            node->link = _____<1>_____;
 }
 else {
     node->link =_____<2>_____;
     _____<3>_____ = node;
  }
 }
```

(b) (4 points) How to add an instruction to modify the above code to insert the node to the last of the circular linked list?

3. (6 points) The following pseudo code is used to construct an expression tree from a postfix expression and finally return the root of the constructed expression tree. Please complete the part of (a), (b), and (c) by using the following operations: **create**(V), **Push**(), **Pop**(), and necessary variable.

N = **create**(V): will return the tree node which stores the input data V. **Push**(N): will insert N into the stack. N = **Pop**(): N will get the popped result from the stack.

It is assumed that all the operators in the postfix expression are binary operators (+, -, *, /) and N.***leftChild*** and N.***rightChild*** are pointers of tree nodes for pointing to the left child and right child of N, respectively.

Step 1) Create a stack to store tree node.
Step 2) Scan the given postfix expression and do the following for each scanned element E.
If the element E is a number
(a)
else if the element E is an operator
(b)
Step 3) When the expression is ended
(c)

4. (9 points) Given the *adjacency matrix* with cost weights of an undirected graph ***G***.

$$G = \begin{bmatrix} 0 & 5 & 6 & 2 & 0 & 0 \\ 5 & 0 & 6 & 0 & 7 & 0 \\ 6 & 6 & 0 & 4 & 5 & 1 \\ 2 & 0 & 4 & 0 & 0 & 3 \\ 0 & 7 & 5 & 0 & 0 & 6 \\ 0 & 0 & 1 & 3 & 6 & 0 \end{bmatrix}$$

(a) (5 points) Apply Prim's algorithm to find the Minimum Spanning Tree of this graph. Please write the table on your answer page and give number (starting from node 1) on the selected edge to describe the order that the edges are selected by the algorithm. If some edges are not selected, write "-" on their **Selected order** column.

			Selected order
E1	1	2	
E2	1	3	
E3	1	4	
E4	2	3	
E5	2	5	
E6	3	4	
E7	3	5	
E8	3	6	
E9	4	6	
E10	5	6	

(b) (4 points) If you want to apply Kruskal's Algorithm to find the Minimum Spanning Tree of the graph ***G***. Which data structure can be applied in the implementation to make the algorithm perform efficiently? (**there may be more than one correct answer**)

A) binary search tree
B) minimum heap
C) hash table
D) adjacency matrix
E) disjoint tree

5. (6 points) Given the following array for sorting the numbers from small to large by **heap sort algorithm**.

index	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
value	5	1	2	4	3	7	6	8	9

(a) (3 points) Please show the resultant **max-heap** after initially adjustment on the array.
(b) (3 points) Please show the adjusted array after the **largest two** numbers have been exchanged into a[9] and a[8].

6. (10 points) Given the *adjacency matrix* of an undirected graph G2.

$$G2 = \begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

(a) (2 points) Please write the corresponding adjacency list graph[] for the graph ***G2***, where ***graph[i]*** stores a pointer points to the adjacency list of ***node[i]***. The vertices are numbered starting from 0.

```
#define MAX_VERTICES 50 /* Max # of Vertices */
typedef struct node *node_pointer;
typedef struct node {
        int vertex;
        struct node *link;
        };
node_pointer graph[MAX_VERTICES];

#define MIN2 (x, y) ((x) < (y) ? (x) : (y))
short int dfn[MAX_VERTICES];
short int low[MAX_VERTICES];
int num;
void init (void)   {
     int i;
     for (i=0; i<n; i++) {
          dfn[i] = low[i] = -1;
     }
     num = 0;
}

void dfnlow(int u, int v){
   nodePointer ptr;
   int w;
   dfn [u] = low [u] = num++;
   for (ptr = graph[u]; ptr; ptr = ptr->link) {
      w = ptr->vertex;
      if (dfn[w] < 0) {
         dfnlow (w,u);
         low[u] = MIN2(low[u], low[w]);
      }
      else if (w != v)
         low[u] = MIN2(low[u], dfn[w]);
   }
}

int main(){
   init();
   dfnlow(0, -1);
   return 0;
}
```

(b) (6 points) Please perform ***dfnlow***(0, -1) according to the code shown left on the constructed adjacency list ***graph***[] and show the results in ***dfn***[] and ***low***[].

(c) (2 points) Please explain how to decide the ***articulation points*** of the graph according to ***dfn***[] and ***low***[].

7. (10 points) Divide-and-conquer algorithms usually perform more efficiently than brute-force algorithms. Please briefly describe *what computations are eliminated* in comparison with brute-force sorting algorithms (e.g., bubble sort, selection sort) for each of the following divide-and-conquer based methods.
   (a) (5 points) Merge sort
   (b) (5 points) Quicksort

8. (20 points) In the longest-common-subsequence (LCS) problem, we are given two sequences $X = [x_1, x_2, \ldots, x_m]$ and $Y = [y_1, y_2, \ldots, y_n]$ and want to find a maximum-length common subsequence of X and Y. For example, the length of LCS of ATTGA and CCTTAG is 3 (TTA or TTG).

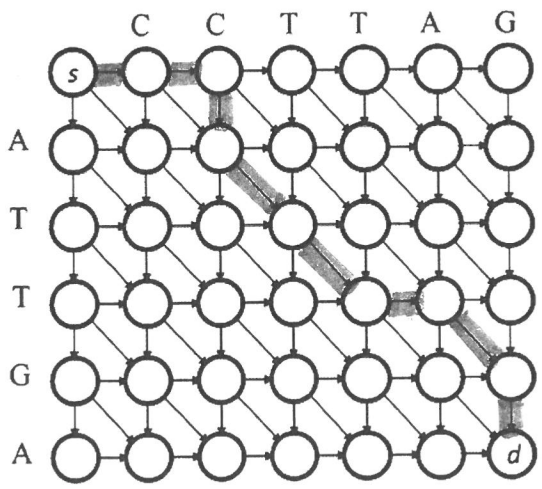
   (a) (5 points) Please describe a brute-force approach for solving the problem. Analyze the time complexity.

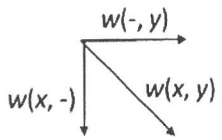
   (b) (5 points) The problem can be solved by using a dynamic programming approach. We create a table $L[i, j]$, in which each element stores the length of a LCS of prefixes $[x_1, x_2, \ldots, x_i]$ and $[y_1, y_2, \ldots, y_j]$. Please write down the recurrence of the dynamic programming approach.

   (c) (5 points) Continuing (b), please fill the table L for the above instance X = [ATTGA] and Y = [CCTTAG].

   (d) (5 points) The LCS problem can also be considered as a longest simple path problem, where we find a path from the source s to the destination d that has the maximal path weight. For example, below we show a graph and the edge weights for the example X = [ATTGA] and Y = [CCTTAG].

   In general, the longest simple path problem is NP-complete. Please describe the reason why this instance of the longest simple path problem can be solved in polynomial time.





w	A	T	C	G	-
A	1	0	0	0	0
T	0	1	0	0	0
C	0	0	1	0	0
G	0	0	0	1	0
-	0	0	0	0	0

9. (15 points) A Huffman code is an optimal variable-length coding approach that assigns bit strings to characters based on their frequencies. The algorithm builds a binary tree in a bottom-up manner by iteratively combining two nodes with least frequencies.

   (a) (5 points) Please describe the reason why Huffman's algorithm is a greedy algorithm.

   (b) (5 points) Consider the 5-character alphabet {A, B, C, D, E} with the following occurrence probabilities:

Character	A	B	C	D	E
Probability	0.35	0.1	0.2	0.2	0.15

   What is the output of Huffman's algorithm?

   (c) (5 points) What is the maximal length of a codeword possible in a Huffman coding of an alphabet of n characters?

10. (5 points) The maximum character frequency is less than twice the minimum character frequency. Does Huffman coding in this case more effective than using an 8-bit fixed-length code? Why?