國立成功大學 105 學年度碩士班招生考試試題

系　　所：電腦與通信工程研究所

考試科目：資料結構　　考試日期：0228，節次：2

※ 考生請注意：本試題不可使用計算機。 **請於答案卷(卡)作答，於本試題紙上作答者，不予計分。**

1. (5 **pts**) Which of the following statement(s) is (are) true?
   (A) The worse case time complexity of QuickSort is O(nlogn).
   (B) QuickSort is a stable algorithm.
   (C) MergeSort is not an in-place sort.
   (D) Any algorithm that sorts only by comparions must have a worst case computing of $\Omega(n \log n)$.
   (E) RadixSort is a comparion based sort.
   (F) None of the above.
2. (5 **pts**) A depth-first algorithm classifies the edges of a graph into tree, back, forward, and cross edges. If we classify the edges reachable from a source in a breadth-first tree into the same four categories, please show which of the following statement(s) is (are) true after applying breath-first search algorithm to an **undirected graph**.
   (A) There are no back edges.
   (B) There are no cross edges.
   (C) For each tree edge (u, v), we have d[v]=d[u]+1.
   (D) For each cross edge (u, v), we have d[v]=d[u] or d[v]=d[u]+1;
   (E) There are no forward edges.
3. (5 **pts**) Which of the following statement(s) is (are) true?
   (A) With postfix notation, there is no need for parentheses because there is no ambiguity.
   (B) Evaluating the postfix expression "4 3 – 5 6 3 / +*" returns 27.
   (C) If we use a stack to evaluate the postfix expression in (B) from left to right, a total of 8 pop operations are performed.
   (D) "+*abcd" is a valid prefix expression.
   (E) None of the above.
4. (5 **pts**) Suppose each set contains only one element initially. Which of the following statement(s) about the disjoint-set forest is (are) true?
   (A) In the worst case, find an element in a set of size n takes O(logn) time.
   (B) The height of the tree with n elements is at most O(logn) if weighting rule for union is applied.
   (C) To have a set with n elements, we at least have to perform (n-1) unions.
   (D) By using disjorint-set data structure, Krusakal's algorithm can check if a new edge will form a cycle during constructing a MST.
   (E) None of the above.
5. (15 **pts**) Please answer the following questions shortly.
   (1) (2 **pts**) Each car of a deck has two keys, which are the suit (K_1) and face values (K_2), respectively. Please shows the sequence of K_1 and K_2 if want to sort a deck of cards by the most-significant-digit-first (MSD) sort.

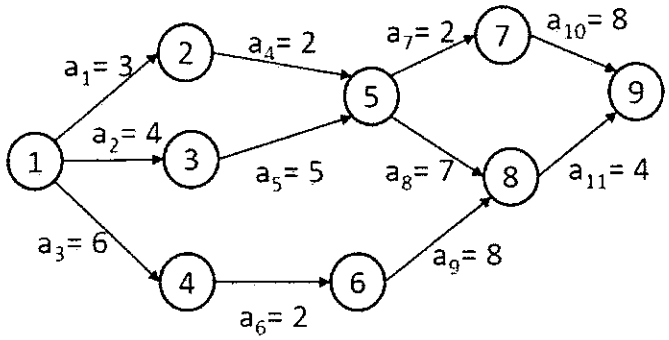
(2) (2 **pts**) What is your strategy while you put a value into bucket: (a) First-in First out. (b) First in Last out in LSD Radix Sort?

(3) (8 **pts**) Write the status of list (31, 42, 17, 2, 8, 28, 10) for each pass during LSD Radix Sort. Use r= 10. N=2.

	0	1	2	3	4	5	6	7	8	9
Iteration 1										
Iteration ...										

(4) (3 **pts**) Under what conditions would an MSD Radix Sort be more efficient than an LSD Radix Sort?

6. (10 **pts**) After performing the following operations into a binary search tree, please answer the following problems: insert (7), insert (4), insert (3), insert (2), insert (9), insert (1).
   (1) (2 **pts**) What is the resulting binary search tree?
   (2) (2 **pts**) What is result of postorder traversal on the tree in (1)?
   (3) (6 **pts**) What is the resulting binary search tree after deleting (7)?
7. (20 **pts**) Given an activity-on-Edge (AOV) network bellow:



(1) (6 pts) Please list earlies event occurrence times (i.e ee[]) and latest even occurrence times (i.e., le[]) for all events.

K	V1	V2	V3	V4	V5	V6	V7	V8	V9
ee(k)									
le(k)									

(2) (8 **pts**) Please list early activity time (i.e e[i]) and late activity time (i.e., l[i]) for each activity a_i, which crosses the edge (k, j), where e(i)=ee(k) and l(i)=le(j) –DUR(a_i)

	a1	a2	a3	a4	a5	a6	a7	a8	a 9	a10	a11
e(i)											
l(i)											

(3) (3 **pts**) Please list critical activities.

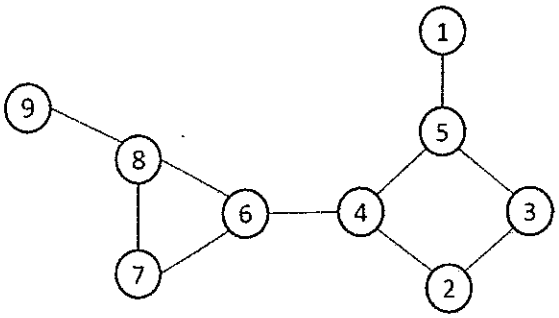
(4) (3 **pts**) Please list critical paths.

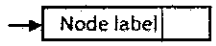
(5) (1 **pts**) Is there exists any single activity whose speed up would result in reduction of the project length?

8. (20 **pts**) A vertex *v* of *G* is an articulation point iff the deletion of *v*, together with the deletion of all edges incident to v, leaves behind a graph that has at least two connected components. Articulation points can be identified by calculating the *dfn* and *low* value of each vertex.

Low[v]=min{dfn[v],
min{low[w] | w is child of v}
min{dfn[w] | (v, w) is back edge} }

A vertex *x* is an articulation point iff (a) *x* is the root of the spanning tree with two or more children, or (b) *x* is not the root and has a child y such that *low*(*y*)>=*dfn*(*x*).



(1) (2 **pts**) Please represent the graph using the adjacency list (i.e, each node in the list is represent by 

). **Note that the order of the adjacency list of each vertex follows the ascending order of the vertex index.**

(2) (3 **pts**) The DFS sequence of G (assume that search always starts from the node with the smallest label), where the root of the depth-first spanning tree is set to vertex 5.

(3) (10 **pts**) Please list the dfn and low values of each vertex in the given table. Note that the order of the adjacency list of each vertex follows the ascending order of the vertex index.

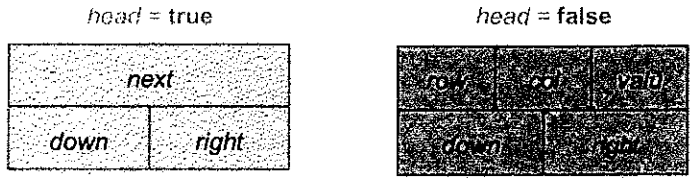
Vertex	1	2	3	4	5	6	7	8	9
Dfn									
low									

(4) (3 **pts**) Find all articulation points of graph G.

(5) (2 **pts**) Find all bin-connected components of G.

9. (15 **pts**) Please use a linked representation for a sparse matrix, where each nonzero element is in **two circular lists**. One is in a row list and the other is in a column list. The following figure shows the data structures (header node and element node.) used for a sparse matrix.

- Each header node has three fields: down, right, and next. The header node for row *i* is also the header node for column *i*. The *down* (*right*) field is used to link into a column (row) list. The *next* field links the header nodes together.
- Each nonzero element is recorded in an element node, where the *down* (*right*) field is used to link the next nonzero element in the same column (row). The fields *row* and *col* represent the row and column of the nonzero element in the matrix, which is label from 0. For example, (1, 0) =3 and (0, 1) =1 in the matrix below, and their values are 3 and 1, respectively.
- The list of header nodes has a header node that is identical to the nodes used to represent nonzero elements. The *row* and *col* fields of this node are used to store the matrix dimensions, and value denotes number of nonzero elements.



Header node　　Element node

(a) (5 **pts**) Please draw the linked representation of the spare matrix $A = \begin{bmatrix} 0 & 1 & 0 \\ 3 & 0 & 2 \\ 0 & 0 & 0 \\ -5 & 0 & 0 \end{bmatrix}$.

(b) (10 **pts**) A faster way to return the nodes is to set up an available-space list. Assume that *av* points to the fist node of this list and that this list is linked through of the field *right*. Please show the resulting available list after performing the function bellow. You only require to show the sequence of the nonzero element in the list. Assume av is NULL in the beginning.

```
Matrix::~Matrix() { // dtor
// Return all nodes to the av list, linked via right field. av is a static variable pointing to the first node of the av list
    if (!headnode) return; // no nodes to delete
    MatrixNode *x= headnode->right; // x points to H0
    headnode->right = av; av = headnode;   // return headnode
    while (x != headnode) { // erase by rows
        MatrixNode *y = x->right; // y points to headnode or 1st element
        x->right = av;
        av = y; // return an entire row
        x = x->next; // move to next row
    }.
    headnode = 0;
```