Data Structure Final Exam Question Bank

1. Term explanation:
    a. Singly Linked List
    b. Stable Sort
    c. Max Heap
    d. Connected Component

2. The below shows an algorithm to find equivalence classes. If the input file "equiv.in" is as follows:

    12 (0, 4) (3, 1) (6, 10) (8, 9) (7, 4) (6, 8) (3, 5) (2, 11) (11, 0) (1, 6)

    Please list the lists after pairs have been input. (Show the results when the program has finished the line 34)

```
class ENode {
friend void Equivalence( );
public:
      ENode(int d = 0)    // constructor
            {data = d; link = 0;}
private:
      int data;
      ENode *link;
};
void Equivalence( )
{// input equivalence pairs and outpu the equivalene classes
       ifstream inFile( "equiv.in", ios::in); // "equiv.in" is the input file
       if (!inFile) throw "Cannot open input file.";
       int i, j, n;
       inFile >> n;    // read no. of objects
       // initialize first and out
       ENode **first = new ENode* [n];
       bool *out = new bool[n];
       // use STLfunction fill to initialize
      fill (first, first + n, 0);
      fill (out, out + n, false);
// phase 1: input equivalence pairs
      inFile >> i >> j;
      while (inFile.good()) { //check EOF
            first[i] = new ENode (j, first[i])
            first[j] = new ENode (i, first[j])
             inFile >> i >> j;
      }
```

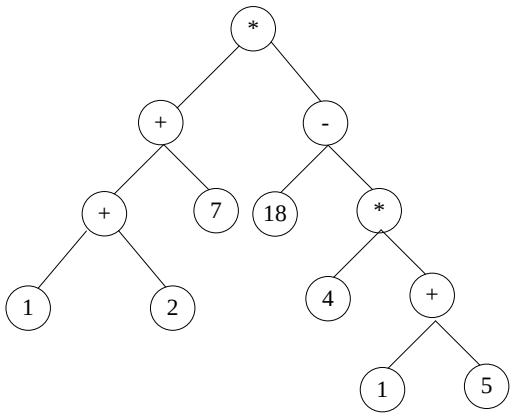
```
// phase 2: output equivalence classes
for (i = 0; i < n; i++)
    if (!out[i]) { // needs to e output
        cout << endl << "A new class: " << i;
        out[i] = true;
        ENode *x = first[i]; ENode *top = 0; // initialize stack
        while (1) {    // scan rest of class
            while (x) { // process the list
                j = x→data;
                if (!out[j]) {
                    cout << ", " << j;
                    out[j] = true;
                ENode *y = x →link;
                x→link = top;
                top = x;
                x = y;
                }
                else x = x→link;
            } // End of while(x)
            if (!top) break;
            x = first [top→data];
            top = top→link; // delete
        } // while(1) ends
    } // if (!out[i]) ends
    for (i = 0; i < n; i++)
            while (first[i]) {
                ENode *delnode = first[i];
                first[i] = delnode →link;
                delete delnode;
            }
        delete [] first; delete [] out;
    }
```

Ans:

Three equivalent classes {0,2,4,7,11}, {1,3,5}, {6,8,9,10}

3. 請圖示說明下列Linked Lists.
   - **a.** Singly Linked Lists.
   - **b.** Circular Linked Lists.
   - **c.** Doubly Linked Lists (circular and with head node).

4. What is the result of a) a **preorder** traversal, b) an **in-order** traversal, and c) a **postorder** traversal on the following expression tree?



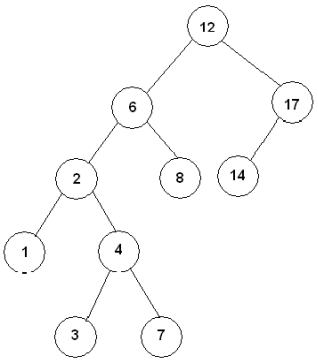
Ans:

a. * + + 1 2 7 – 18 * 4 + 1 5

b. 1 + 2 + 7 * 18 – 4 * 1 + 5

c. 1 2 + 7 + 18 4 1 5 + * - *

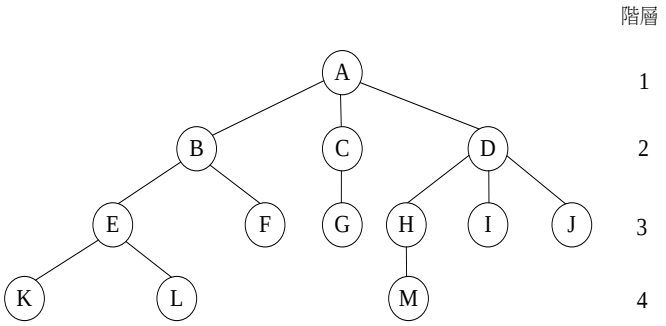
5. Consider the tree pictured, is it a valid binary search tree? Why or why not?



6. Given a list representation of a tree, draw the tree.

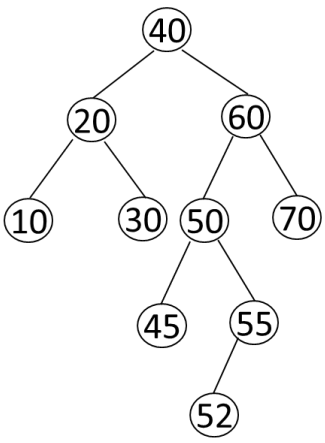
(*A*(*B*(*E*(*K*,*L*),*F*),*C*(*G*),*D*(*H*(*M*),*I*,*J*)))

Ans:

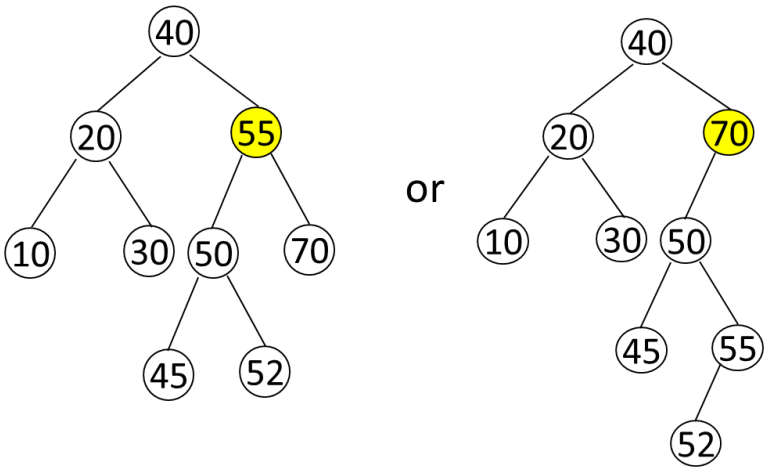


7. Given an arbitrary binary tree *T* with integer keys stored at the nodes, **design an efficient algorithm** which determines whether or not T is a binary search tree.

8. Given the below **binary search tree**, show the results after 60 is deleted.

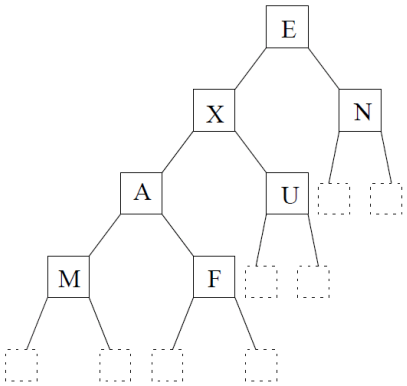


Ans:



9. Consider binary trees that have single characters stored at each internal node. Draw a binary tree that will spell out **EXAMFUN** upon a *pre-order* traversal and **MAFXUEN** upon an *in-order* one. Inserting a node r as the right child of a node s. (Hint: Every binary tree has a unique preorder-inorder sequences.)

Ans:



10. write a recursive function that counts and returns number of non-leaf nodes in binary tree.(10p)

Ans: (just FYR)

```
/* Computes the number of non-leaf nodes in a tree. */
int countNonleaf(struct Node* root)
{
    // Base cases.
    if (root == NULL || (root->left == NULL &&
                         root->right == NULL))
        return 0;

    // If root is Not NULL and its one of its
    // child is also not NULL
    return 1 + countNonleaf(root->left) +
               countNonleaf(root->right);
}
```

11. write a recursive function that counts and returns number of leaf nodes in binary tree.(10p)

Ans: (just FYR)

```
/* Function to get the count of leaf nodes in a binary tree*/
unsigned int getLeafCount(struct node* node)
{
 if(node == NULL)
     return 0;
 if(node->left == NULL && node->right==NULL)
     return 1;
 else
     return getLeafCount(node->left)+getLeafCount(node->right);
}
```

12. Write an Append() function that takes two linked lists, listA and listB, appends listB onto the end of listA.

listA: 13->5->42->null

listB: 7->4->9->45->null

after call to append listA becomes

listA: 13->5->42->7->4->9->45->null

Ans: (just FYR)

```
/* Function to append listB onto the end of listA.*/
unsigned int Append(struct node* node)
{
 if(node == NULL)
     return 0;
 if(node->left == NULL && node->right==NULL)
     return 1;
 else
     return getLeafCount(node->left)+getLeafCount(node->right);
}
```

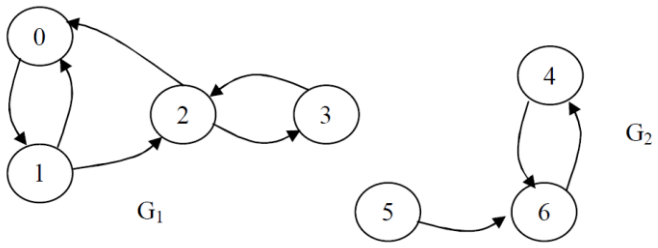
13. Complete the Insert() and InsertBack() function of the Chain class.

```
template < class T > class Chain;    // forward declaration
template < class T >
class ChainNode {
friend class Chain <T>;
private:
      T data;
      ChainNode<T>* link;
};
template <class T>
class Chain {
public:
      Chain( ) {first = 0;} // constructor initializing first to 0
      // Chain manipulation operations
      InsertBack(const T &e); //insert e to the back of the list
      Insert(const T &e); //insert e to the beginning of the list
      //…
private:
      ChainNode<T> * first;
}
```

```
template < class T >
void Chain<T>::InsertBack(const T& e) //insert e to the back of the lis
{
      if (first) { // nonempty chain
            //fill your code here
      }
      else first = new ChainNode<T>(e);
}
```

```
template < class T >
void Chain<T>::Insert(const T& e) //insert e to the begin of the list
{
      if (first) { // nonempty chain
            //fill your code here
      }
      else first = new ChainNode<T>(e);
}
```

14. For the digraph G shown as below.



a. Write out in-degree and out-degree of the vertices 1, 3, 5.

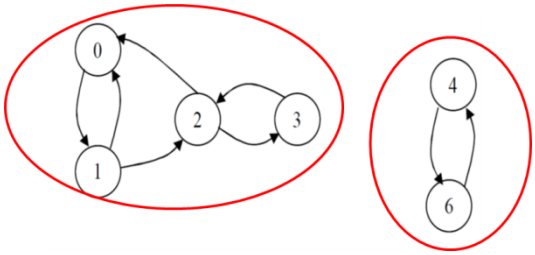
b. Write out the strongly connected components of G.

c. Find a cycle in subgraph G_1 of G.

Ans: a.

vertex	In-degree	Out-degree
1	1	2
3	1	1
5	0	1

b. G has two strongly connected components as below:



c. For example, nodes 2 and 3 have a cycle. Nodes 0, 1, 2 have a cycle.

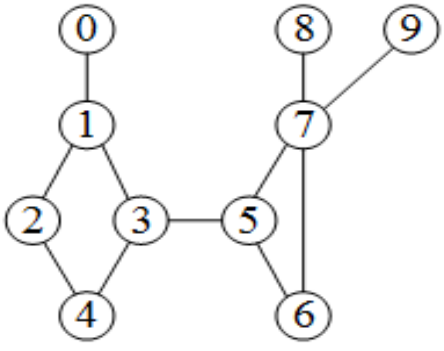
15. $dfn(w)$ is defined as the order that w is discovered by DFS and $low(w)$ is the lowest $dfn(w)$ that can be reached from *w* using a path of descendants followed by, at most, one back edge, i.e.,

$$low(w) = min\{\, dfn(w),\ min\{\, low(x) | x \text{ is a child of } w\},\ min\{\, dfn(x) | (w,x) \text{ is a back edge}\}\}$$

Given the graph below, **starting from node 3**,

a. show the Depth-First Spanning Tree and the back edges in dotted line

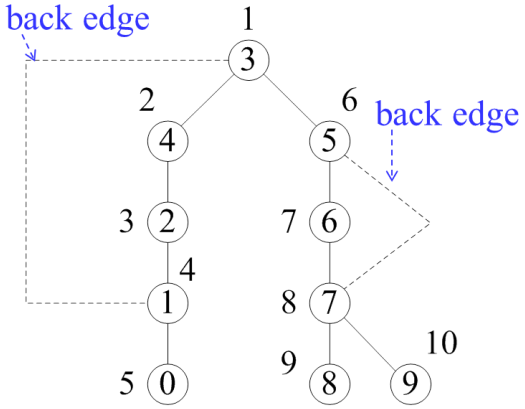
b. fill the below table and figure out which nodes are the articulation point.



vertex	0	1	2	3	4	5	6	7	8	9
dfn										
low										

Ans: (本題答案假設從3開始進行DFS tree展開)

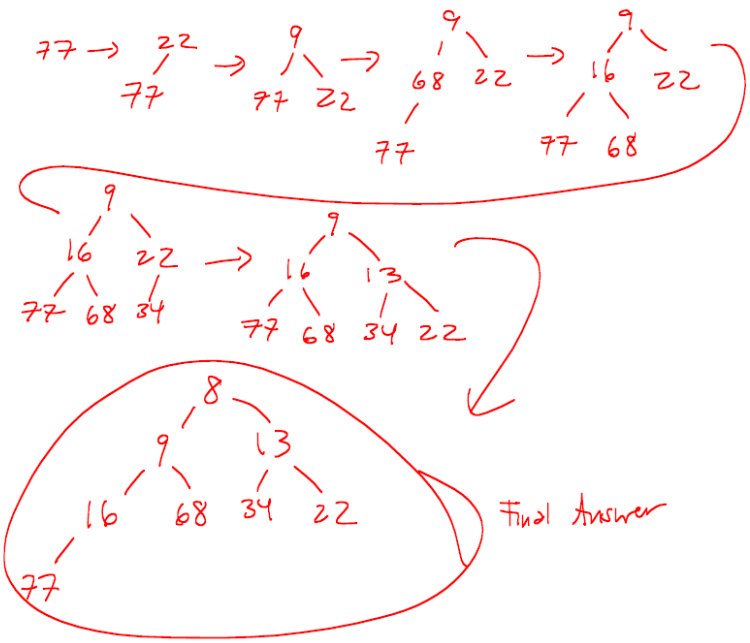
a.



b.

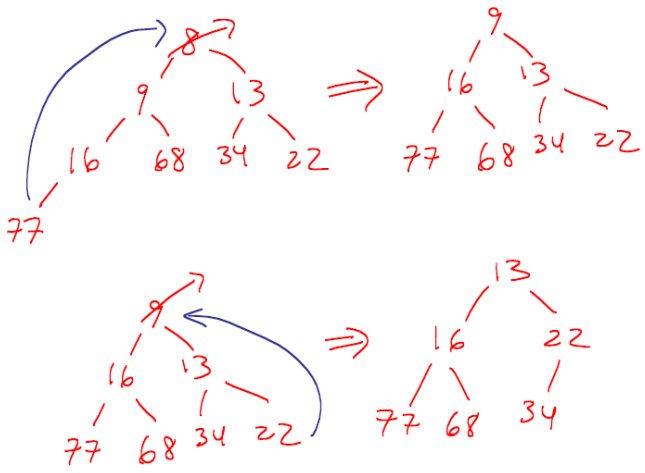
vertex	0	1	2	3	4	5	6	7	8	9
dfn	5	4	3	1	2	6	7	8	10	9
low	5	1	1	1	1	6	6	6	10	9

16. (8 pts) Draw the **binary min heap** that results from inserting: **77, 22, 9, 68, 16, 34, 13, 8** in that order into an initially empty binary min heap. You do not need to show the array representation of the heap. You are only required to show the final heap, although if you draw intermediate heaps, ***please circle your final result.***

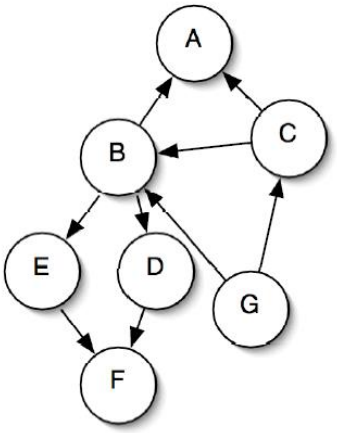


17. (5 pts) Draw the binary min heap that results from doing 2 ***deletemins*** on the

heap that you created in Question 16. You are only required to shows the final heap, although if you draw intermediate heaps, ***please circle your final result.***



18. (10pts) What is a **topological ordering** of the following graph? Assume adjacency List is used to represent the graph, write out an algorithm that can find topological orders.



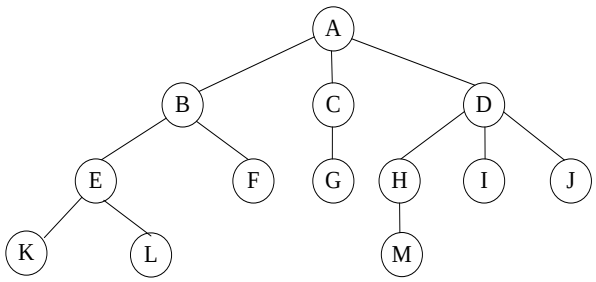
Ans:

a. A *topological order* is a linear ordering of the vertices of a graph such that, for any two vertices i and j, if i is a predecessor of j in the network, then i precedes j in the linear ordering. G ->C ->B ->(A-> D-> E)->F <=括弧內 ADE 順序可任意調換

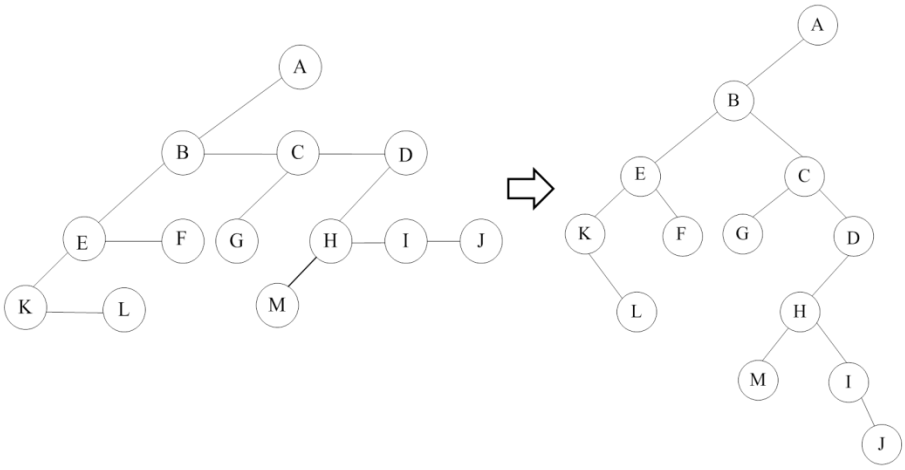
b.

```
1  void LinkDigraph::TopologicalOrder()
2  {// the n vertices of a network are listed in topological order
3     int top = -1;
4     for (int i = 0; i<n; i++) //generate a linked stack of vertices with no predecessors
5        if (count[i] == 0) { count[i] = top; top = i;}
6     for (i = 0; i< n; i++)
7        if (top = =-1) throw "Network has a cycle.";
8        int j = top; top = count[top];    //unstack a vertex
9        count << j <<endl;
10       Chain<int>::ChainIterator ji = adjLists[j].begin();
11       while (ji) { // decrease the count of the successor vertices of j
12            count[*ji]- -;
13            if (count[*ji] == 0) { count[*ji] = top; top = *ji;}    // add *ji to stack
14            ji++;
15       }
}
```

19. (5pts) Show the binary tree representation of the below tree.

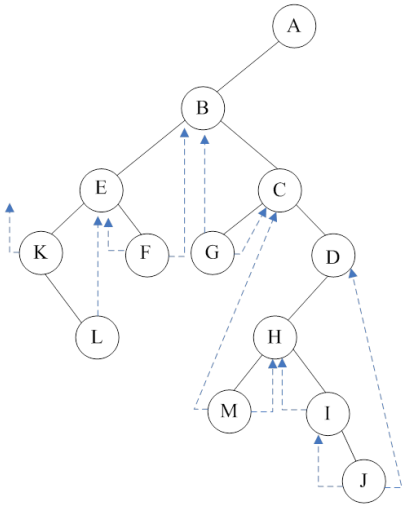


Ans: (下圖左 tree 或右 tree 都可)



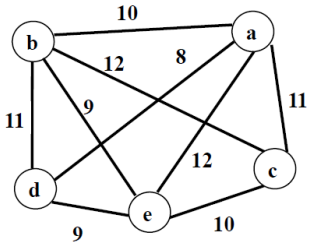
20. Convert the binary tree that you created in Question 19 into a threaded binary tree.(Hint: inorder traversal)

Ans:



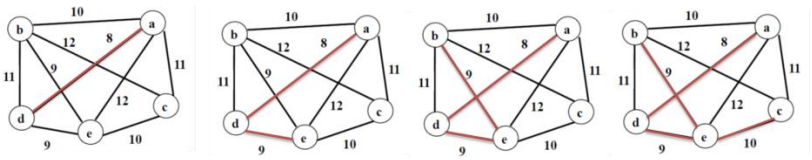
21. Following Question 20, show the threaded binary tree after inserting a node R as the right child of node C.

22. Find the minimum cost spanning tree (MST) with using **Kruskal's algorithm** and the **Prim's algorithm** and show the detailed steps.

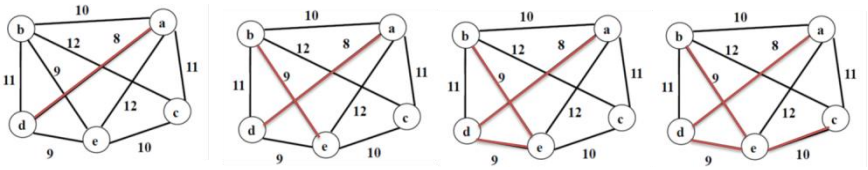


Ans:

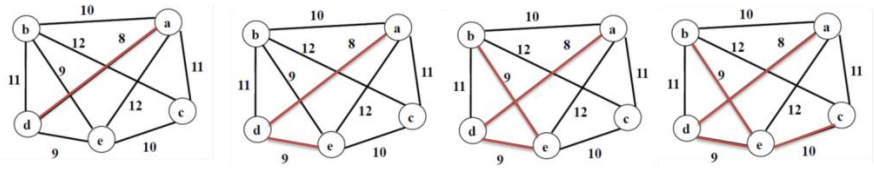
Kruskal's



or

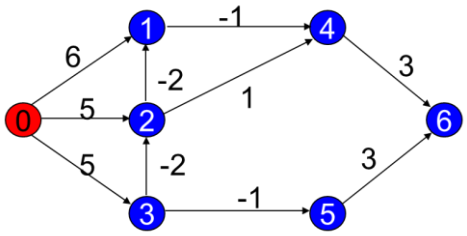


Prim's



23. Given the BellmanFord algorithm and the graph below, compute the shortest path from node 0 to all the other nodes.

```
1 void MatrixWDigraph::BellmanFord(const int n, const int v)
2 { // single source all destination shortest paths with negative edge lenths.
3    for (int i = 0; i < n ; i++) dist[i] = length[v][i]; // initialize dist
4    for (int k = 2; i <= n−1 ; k++)
5       for (each u such that u != v and u has at least one incomeing edge)
6          for (each    <i, u> in the graph)
7             if (dist[u] > dist[i] + length[i][u]) dist[u] = dist[i] + length[i][u];
8 }
```



Ans:

k	$dist^k[7]$						
	0	1	2	3	4	5	6
1	0	6	5	5	∞	∞	∞
2	0	3	3	5	5	4	∞
3	0	1	3	5	2	4	7
4	0	1	3	5	0	4	5
5	0	1	3	5	0	4	3
6	0	1	3	5	0	4	3

24. Here is an array of ten integers:

5 3 8 9 1 7 0 2 6 4

a. Draw the array after every iteration in a selection sort (sorting from smallest to largest).

b. Draw the array after every iteration in an insertion sort (sorting from smallest to largest). This iteration has shifted at least one item in the array!

d. Draw the array after every iteration in a quick sort (sorting from smallest to largest). Draw a circle for each pivot in each iteration.

Ans:

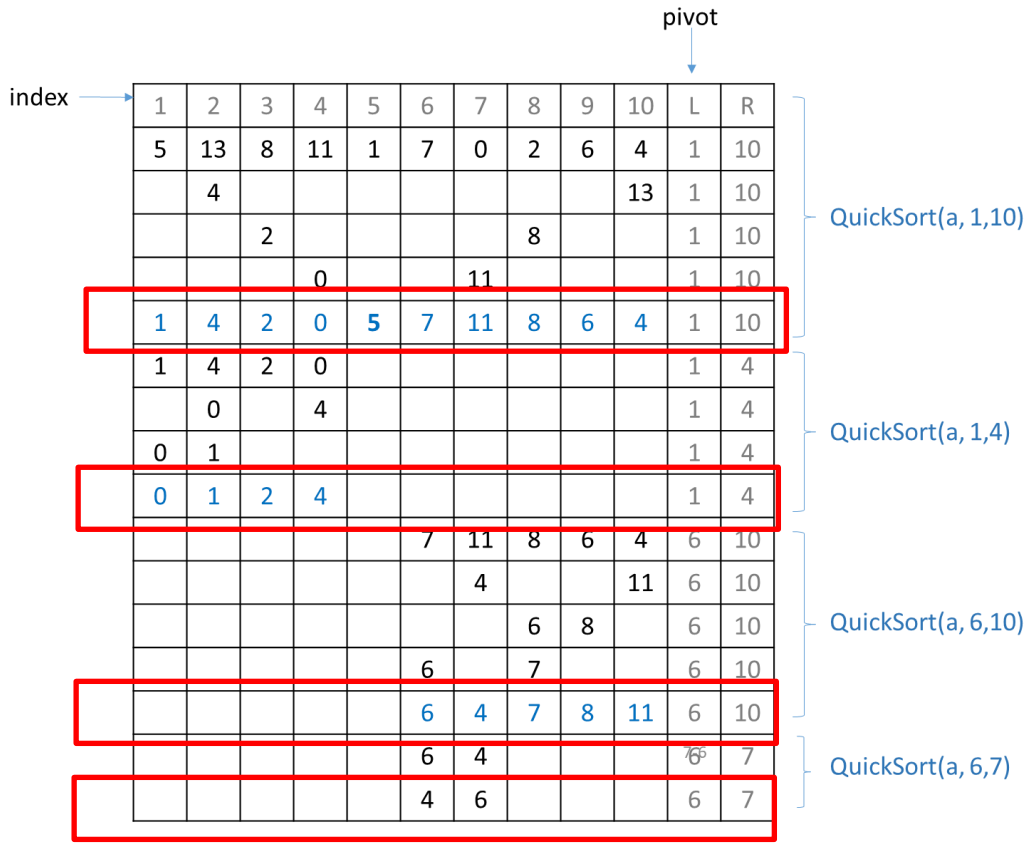
(a)

5	13	8	11	1	7	0	2	6	4
0	13	8	11	1	7	5	2	6	4
0	1	8	11	13	7	5	2	6	4
0	1	2	11	13	7	5	8	6	4
0	1	2	4	13	7	5	8	6	11
0	1	2	4	5	7	13	8	6	11
0	1	2	4	5	6	13	8	7	11
0	1	2	4	5	6	7	8	13	11
0	1	2	4	5	6	7	8	11	13

(b)

5	13	8	11	1	7	0	2	6	4
5	8	13	11	1	7	0	2	6	4
5	8	11	13	1	7	0	2	6	4
1	5	8	11	13	7	0	2	6	4
1	5	7	8	11	13	0	2	6	4
0	1	5	7	8	11	13	2	6	4
0	1	2	5	7	8	11	13	6	4
0	1	2	5	6	7	8	11	13	4
0	1	2	4	5	6	7	8	11	13

(c) 演算過程如下(答案可僅列每個 recursive 的結果，紅色框部分)



23.

a. For the AOE network below obtain the early, e( ), and late l( ), start times for activities a3 and a10. Use the forward-backward approach.

b. What is the earliest time the project can finish?

c. Which activities are critical?

d. Is there any single activity whose speed up would result in a reduction of the project length?

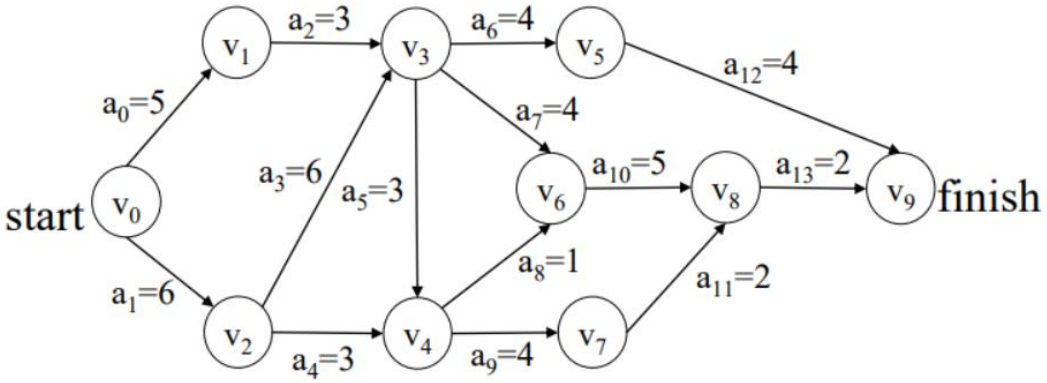
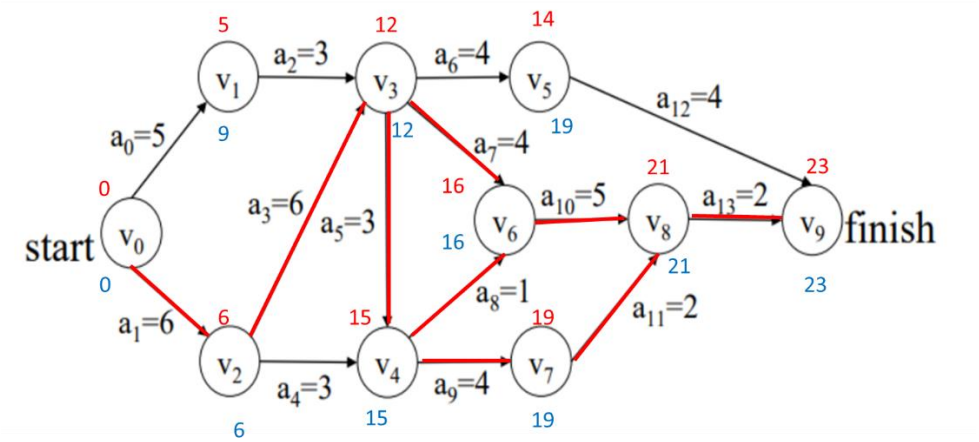


Figure 6.46

Ans:

Forward+Backward



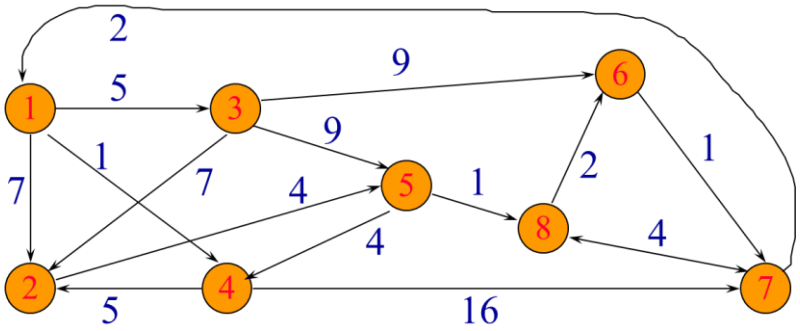
a. e(a_3)=5 l(a_3)=9 and e(a_{10})=16 l(a_{10})=16

b. 23

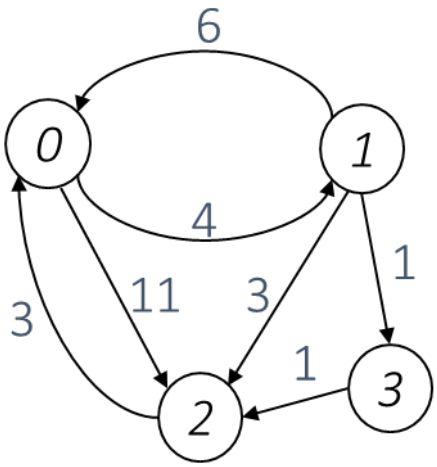
c. a_1, a_3, a_5, a_7, a_8, a_9, a_{10}, a_{11}, a_{13}

d. a_1(2) or a_{13} (1) or a_3(2)

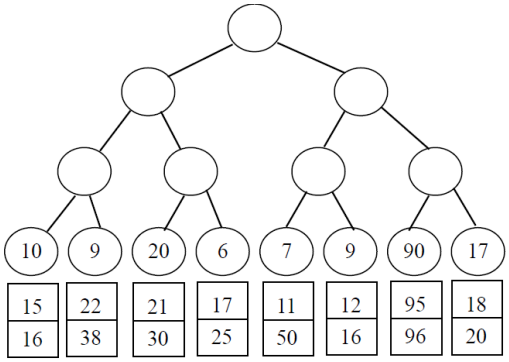
25. Given the following graph, compute the shortest path from node 1 to all the other nodes by using the Dijkstra's Algorithm.



26. Given the following graph, compute the all-pairs shortest paths by using Floyd-Warshall's algorithm.



27. Given a winner tree for $k = 8$ as follows



a. Finish the winner tree. (5 %)
b. If we output two elements from the winner tree, please draw the restructured winner tree. (5 %)
c. Draw its corresponding loser tree. (10%)

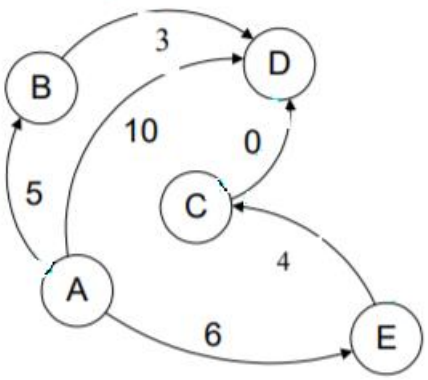
28. If we have the inorder sequence D B G E I A F C and preorder sequence A B D E G I C F, please draw out this binary tree.

29. Show the steps required to do a radix sort on the following set of values when using base 10.

346, 22, 31 212 157 102 568 435 8 14 5

0	1	2	3	4	5	6	7	8	9

30. (a) Draw both the adjacency matrix and adjacency list representations of this graph. Be sure to specify which is which.



(b) Give two valid topological orderings of the nodes in the graph

(c) Step through Dijkstra's Algorithm to calculate the single source shortest path from A to every other vertex. You only need to show your final table, but you should show your steps in the table below for partial credit. Show your steps by crossing through values that are replaced by a new value

Vertex	Distance	Path
A		
B		
C		
D		
E		

(d)In what order would Dijkstra's algorithm mark each node as known?

(e) What is the shortest (weighted) path from A to D?

(f) What is the length (weighted cost) of the shortest path you listed in part (e)?