FIRST SEMESTER (2020 SCHEME)

PRACTICAL EXAMINATION JUNE - JULY 2021

20MCA135 DATA STRUCTURES LAB

AISWARYA PAVITHRAN

ICE20MCA - 2004

Date : 30 June 2021

Time : 9:30 - 12:30

BATCH - I

D. Sorting of an integer Array.

Algorithm

Program ~~for~~ to accept N numbers and arrange them in ascending order.

Step 1: declare i, j, a, n, number[30]

Step 2: read limit of numbers sort

Step 3: read values of number to sort.

Step 4: using for loop store the number in an array declared number[i]

Step 5: ~~Print~~ After storing the numbers into an array ~~again~~ check the first number and second number using ~~for two~~ two for loop.

Step 6: check first number is greater than second number using if condition.

Step 7: If condition is true then store first number to a temporary variable 'a'.
and ~~store~~ swap first number in second number
then second number = a.

Step 8: reapeat step 5, 6, 7 to sort all numbers in the array.

Step 9: Print the numbers arranged in asscending order using for loop.
ie, Print the array.

Step 10: Stop.

Program output

```
Enter the value of N: 4
Enter the numbers:
6
4
10
3
The numbers arranged in asscending order are given below
3
4
6
10.
```

Program

```
#include <stdio.h>
#include <conio.h>
void main()
{
  int i,j,a,n,number[30];
  printf("Enter the value of N:\n");
scanf("%d",&n);
printf("Enter the numbers:\n");
for (i=0; i<n; i++)
      scanf("%d",&number [i]);
for (i=0; i<n; i++)
 {
   for (j=i+1; j<n; j++)
   {
    if (number [i] >number [j])
     {
        a=number[i];
        number[i]=number[j];
        number [j]=a;
     }
   }
 }
printf("The numbers arranged in ascending order are
               given bellow \n");
for (i=0;i<n;i++)
    printf("%d\n",number[i]);
}
```

2). Implementation of Prim's Algorithm.

Algorithm.

Step 1 : Start

Step 2 : declare visited node = {0} min, mincost =0, cost [10][10];

Step 3: read the number of nodes in a-tree.

Step 4: Enter the adjacency matrix using two for loop

~~Step 5: cost of two adjecenty node adjacent node=0~~
~~then set cost of these nodes = 999~~

Step 5: select one vertex as starting vertex (Vs)

Step 6: ~~From~~ Delete the vertex Vs from the vertex set.

Step 7: Find the vertex that is nearest to starting vertex.

Step 8: Set the new vertex as the Vs.

Step 9: Delete vertex from the vertex set.

Step 10: check vertex set is empty ~~using~~ ~~for~~ loop ~~if~~ condition

Step 11: True, then output the minimum ~~sa~~ spanning tree
else, exit, ~~from~~

Step 12: Stop.

Output

```
Enter the no.of nodes : 6
Enter the adjecency matrix: 6

0 3 1 6 0 0
                        Edge 1 : (1 3) cost : 1
3 0 5 0 3 0
                        Edge 2 : (3,6) cost : 1
1 5 0 5 6 1
                        Edge 3 : (6,4) cost : 2
6 0 5 0 0 2
                        Edge 4 : (1 2) cost : 3
0 3 6 0 0 6
                        Edge 5 : (2 5) cost : 3
0 0 4 2 6 0
                        minimum cost = 10
```

~~Edge 1 : (1 3) cost~~ : 1
~~Edge 2 : (3 6) cost~~ : 3
~~Edge 3 : (1 2) cost : 3~~
~~Edge 4 : (2 5) cost~~ : 3 ~~minimum cost = 13~~

Program

```
#include<stdio.h>
#include<conio.h>
int a,b,v,u,n,i,j,ne=1;
int visited[10]={0},min,mincost=0; cost[10][10];
void main()
{
  Printf("Enter the number of nodes:");
   scanf("%d",&n);
   Printf("Enter the adjecency matrix:\n");
   for(i=1; i<=n; i++)
   for(j=1; j<=n; j++)
   {
     scanf("%d",&cost[i][j]);
     if(cost[i][j]==0)
     cost[i][j]=999;
   }
   visited[1]=1;
   Printf("\n");
   while(ne<n)
   {
     for(i=1,min=999; i<=n; i++)
     for(j=1; j<=n; j++)
      if(cost[i][j]<min)
      if(visited[i]!=0)
      {
        min=cost[i][j];
        a=u=i;
        b=v=j;
      }
     if(visited[u]==0 || visited[v]=0)
     {
        Printf("Edge %d:(%d %d) cost:%d",ne++,a,b,min);
        mincost+=min;
        visited[b]=1;
   }
        cost[a][b]=cost[b][a]=999;
   }
        Printf("\n minimum cost %d",mincost);
  }
```