

Write a Program to Sort a list of N elements Using Selection Sort Technique.

### **Selection Sort Algorithm**

```
Read n, a[]
for i ← 0 to n do
    pos=i
    for j ← i+1 to n do
        if(a[j]<a[pos])
            pos ← j
    End for
    Swap a[pos] and a[i]
End for
```

### **Program**

```
#include<stdio.h>

void main()
{
    int a[10],j,i,n,temp,pos;
    printf("Enter the number of limit\n");
    scanf("%d",&n);
    printf("Enter the number of element\n");
    for(i=0;i<n;i++)
        scanf("%d",&a[i]);
    for(i=0;i<n;i++)
    {
        pos=i;
        for(j=i+1;j<n;j++)
        {
```

```
        if(a[j]<a[pos])
pos=j;
    }
    temp=a[pos];
    a[pos]=a[i];
    a[i]=temp;
}
printf("The selection sorted element\n");
for(i=0;i<n;i++)
printf("%d\t",a[i]);
}
```

**Input/Output:**

Enter the number of limit

5

Enter the number of element

25 12 6 3 4

The selection sorted element

3 4 6 12 25

**Input/Output:**

Enter the number of limit

6

Enter the number of element

45 20 40 5 58 15

The selection sorted element

5 15 20 40 45 58

Write a Program to find minimum and maximum value in an array using divide and conquer.

### Algorithm

maxmin( a[], i, j, \*max, \*min)

Read mid,max2,min2,max1,min1;

if(i==j)

    \*max=\*min=i

End if

else if(i==j-1)

    if(a[i]>a[j])

        \*max=i

        \*min=j

End if

else

    \*max=j

    \*min=i

End else

End else

else

    mid=(i+j)/2

Call recursively maxmin(a,i,mid,&max2,&min2)

Call recursively maxmin(a,mid+1,j,&max1,&min1)

    if(a[max2]<=a[max1])

        \*max=max1

    End if

else

    \*max=max2

```
End else
if(a[min2]>a[min1])
    *min=min1
End if
else
    *min=min2
End else
End else
Return
```

### Program

```
#include<stdio.h>

void maxmin(int [ ],int,int,int*,int*);

void main( )
{
    int a[10],n,i,min,max;
    printf("enter the array limit\n");
    scanf("%d",&n);
    printf("enter the array element\n");
    for(i=0;i<n;i++)
    {
        scanf("%d",&a[i]);
    }
    maxmin(a,0,n-1,&max,&min);
    printf("max=%d\n",a[max]);
    printf("min=%d\n",a[min]);
}
```

```
void maxmin(int a[],int i,int j,int *max,int *min)
```

```
{
```

```
    int mid,max2,min2,max1,min1;
```

```
    if(i==j)
```

```
    {
```

```
        *max=*min=i;
```

```
    }
```

```
    else if(i==j-1)
```

```
    {
```

```
        if(a[i]>a[j])
```

```
        {
```

```
            *max=i;
```

```
            *min=j;
```

```
        }
```

```
    else
```

```
    {
```

```
        *max=j;
```

```
        *min=i;
```

```
    }
```

```
}
```

```
else
```

```
{
```

```
    mid=(i+j)/2;
```

```
    maxmin(a,i,mid,&max2,&min2);
```

```
    maxmin(a,mid+1,j,&max1,&min1);
```

```
    if(a[max2]<=a[max1])
```

```
    {
```

```
*max=max1;  
}  
else  
{  
    *max=max2;  
}  
if(a[min2]>a[min1])  
{  
    *min=min1;  
}  
else {  
    *min=min2;  
} } }
```

### **Input/Output**

enter the array limit

5

enter the array element

58 5 45 20 32

max=58

min=5

### **Input/Output**

enter the array limit

6

enter the array element

25 -8 34 77 40 15

max=77

min=-8

Write a Program to implement Divide and Conquer strategy for Quick Sort algorithm to sort list of integers in ascending order.

### Algorithm

```
quicksort(a[ ], l, h)

    input j
    if(l<h)
        Call function j=partition(a,l,h)
        Call recursively quicksort(a,l,j-1)
        Call recursively quicksort(a,j+1,h)
    End if
End

partition(a[ ], l, h)
    Initialize p=a[l], i=l+1, j=h
    while(1)
        while(i<h&& p<=a[i])
            Increment i
        while(p<a[j])
            Decrement j
        if(i<j)
            Swap a[i] and a[j]
        End if
    else
        Swap a[l] and a[j]
        return j
    End else
End while

Return
```

### Program

```
#include<stdio.h>

#include<conio.h>

int quicksort(int [ ],int,int);

int partition(int [ ],int,int);

void main( )

{

    int a[20],i,n;

    clrscr( );

    printf("enter the value of n:\n");

    scanf("%d",&n);

    printf("enter the number to be sort\n");

    for(i=0;i<n;i++)

        scanf("%d",&a[i]);

    quicksort(a,0,n-1);

    printf("Quick sorted array is:\n");

    for(i=0;i<n;i++)

        printf("%d\t",a[i]);

    getch( );

}

int quicksort(int a[ ],int l,int h)

{

    int j;

    if(l<h)

    {

        j=partition(a,l,h);
```



```
    quicksort(a,l,j-1);
    quicksort(a,j+1,h);
}
}
int partition(int a[ ],int l,int h)
{
    int i,j,p,t;
    k=a[l],i=l+1,j=h;
    while(1)
    {
        while(i<h&& p>=a[i])
            i++;
        while(p<a[j])
            j--;
        if(i<j)
        {
            t=a[ i ];
            a[ i ]=a[ j ];
            a[ j ]=t;
        }
        else
        {
            t=a[l];
            a[l]=a[j];
            a[j]=t;
            return j;
        }
    }
}
```

```
}  
}
```

### **Input/Output**

enter the value of n:

5

enter the number to be sort

60 50 25 10 34

Quick sorted array is:

10 25 34 50 60

### **Input/Output**

enter the value of n:

6

enter the number to be sort

75 35 15 -7 50 23

Quick sorted array is:

-7 15 23 35 50 75

Write a Program to implement Merge Sort algorithm for sorting a list of integers in ascending order.

**Algorithm**

mergesort(a[ ], low, high)

    Input mid

    if(low<high)

        mid=(low+high)/2

        Call recursively mergesort(a,low,mid)

        Call recursively mergesort(a,mid+1,high)

        merge(a,low,mid,high)

    End if

return

merge(a[ ], low, mid, high)

initialize k=low, i=low, j=mid+1

    while((i<=mid)&&(j<=high))

        if(a[i]<a[j])

            c[k++]=a[i++]

        else

            c[k++]=a[j++]

    End while

    while(i<=mid)

        c[k++]=a[i++]

    while(j<=high)

        c[k++]=a[j++]

    for i ← low to high do

        a[i]=c[i]

return

### Program

```
#include<stdio.h>

#include<conio.h>

int mergesort(int[ ],int,int);

void merge(int [ ],int,int,int);

void main( )
{
    int a[20],i,n;

    printf("enter the value for n\n");

    scanf("%d",&n);

    printf("enter the element of the merge\n");

    for(i=0;i<n;i++)

        scanf("%d",&a[i]);

    mergesort(a,0,n-1);

    printf("merge sorted array are\n");

    for(i=0;i<n;i++)

        printf("%d\t",a[i]);

    getch();
}

int mergesort(int a[ ],int low,int high)
{
    int mid;

    if(low<high)

    {
        mid=(low+high)/2;

        mergesort(a,low,mid);
```

```
mergesort(a,mid+1,high);
merge(a,low,mid,high);
}
return 0;
}
void merge(int a[ ],int low,int mid,int high)
{
    int i,j,k,c[20];
    k=low;
    i=low;
    j=mid+1;
    while((i<=mid)&&(j<=high))
    {
        if(a[i]<a[j])
            c[k++]=a[i++];
        else
            c[k++]=a[j++];
    }
    while(i<=mid)
        c[k++]=a[i++];
    while(j<=high)
        c[k++]=a[j++];
    for(i=low;i<=high;i++)
        a[i]=c[i];
}
```

**Input/Output:**

Enter the value for n

5

Enter the element :

15

80

6

77

40

Merge sorted array are :

6    15    40    77    80

**Input/Output:**

enter the value for n

6

enter the element of the merge

45

12

-8

35

60

merge sorted array are

-8    12    35    45    60

Write C Program that accepts the vertices and edges for a graph and stores it as an adjacency matrix.

**Algorithm**

```
printGraph(adj[no_vertices][no_vertices])  
for i ← 0 to no_vertices do  
  for j ← 0 to no_vertices do  
    Print adj[i][j]  
  End for  
End for  
End  
Read s, d, no_vertices, adj[no_vertices][no_vertices];  
for i ← 0 to no_vertices do  
  for j ← 0 to no_vertices do  
    Initialize adj[i][j]=0  
  End for  
End for  
while(s!=-1&& d!=-1)  
  Input s,d  
  adj[s][d]=1  
  adj[d][s]=1  
End while  
Call function printGraph(adj)  
End
```

### Program

```
#include<stdio.h>

#include<conio.h>

int no_vertices;

void printGraph(int adj[no_vertices][no_vertices])
{
    for(int i=0;i<no_vertices;i++)
    {
        for(int j=0;j<no_vertices;j++)
        {
            printf("%d\t",adj[i][j]);
        }
        printf("\n");
    }
}

int main()
{
    int s,d;

    printf("\nEnter the number of vertices :");

    scanf("%d",&no_vertices);

    int adj[no_vertices][no_vertices];

    for(int i=0;i<no_vertices;i++)
    {
        for(int j=0;j<no_vertices;j++)
        {
            adj[i][j]=0;
        }
    }
```



```
}  
while(s!=-1&& d!=-1)  
{  
    printf("Enter the Edge from node(0 to %d) to node(0 to %d)  
    :",no_vertices,no_vertices);  
    scanf("%d%d",&s,&d);  
    adj[s][d]=1;  
    adj[d][s]=1;  
}  
printGraph(adj);  
return 0;  
}
```

**Input/Output:**

Enter the number of vertices :5  
Enter the Edge from node(0 to 5) to node(0 to 5) :0 1  
Enter the Edge from node(0 to 5) to node(0 to 5) :1 2  
Enter the Edge from node(0 to 5) to node(0 to 5) :2 3  
Enter the Edge from node(0 to 5) to node(0 to 5) :3 1  
Enter the Edge from node(0 to 5) to node(0 to 5) :2 4  
Enter the Edge from node(0 to 5) to node(0 to 5) :4 1  
Enter the Edge from node(0 to 5) to node(0 to 5) :3 2  
Enter the Edge from node(0 to 5) to node(0 to 5) :-1 -1

|   |   |   |   |   |
|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 |

Implement a function to print In-Degree, Out-Degree and to display that adjacency matrix.

**Algorithm**

```
initializeMatrix(matrix[MAX_VERTICES][MAX_VERTICES], n)
    for i ← 0 to n do
        for j ← 0 to n do
            Initialize matrix[i][j] = 0
        end for
    End for

End

addEdge(matrix[MAX_VERTICES][MAX_VERTICES], start, end)
    Define matrix[start][end] = 1
end

calculateDegree( matrix[MAX_VERTICES][MAX_VERTICES], n) {
    Initialize inDegree[MAX_VERTICES] = {0}
    initialize outDegree[MAX_VERTICES] = {0}
    for i ← 0 to n do
        for j ← 0 to n do
            outDegree[i] += matrix[i][j]
            inDegree[j] += matrix[i][j]
        End for
    End for
    for i ← 0 to n do
        Print i, inDegree[i], outDegree[i])
    End for
End

displayMatrix(matrix[MAX_VERTICES][MAX_VERTICES], n)
    for i ← 0 to n do
```

```
        for j ← 0 to n do
            Print matrix[i][j])
        End for
    End for

End

Read n

Call function initializeMatrix(adj, n)

Read e, start, end

Call function addEdge(adj, start, end)

Call function calculateDegree(adj, n)

Call function displayMatrix(adj, n)
```

### Program

```
#include <stdio.h>

#define MAX_VERTICES 100

void initializeMatrix(int matrix[MAX_VERTICES][MAX_VERTICES], int n)
{
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            matrix[i][j] = 0;
        }
    }
}

void addEdge(int matrix[MAX_VERTICES][MAX_VERTICES], int start, int end)
```

```
{
    matrix[start][end] = 1;
}

void calculateDegree(int matrix[MAX_VERTICES][MAX_VERTICES], int n)
{
    int inDegree[MAX_VERTICES] = {0};
    int outDegree[MAX_VERTICES] = {0};
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            outDegree[i] += matrix[i][j];
            inDegree[j] += matrix[i][j];
        }
    }
    printf("Vertex\tIn-Degree\tOut-Degree\n");
    for (int i = 0; i < n; i++)
    {
        printf("%d\t%d\t\t%d\n", i, inDegree[i], outDegree[i]);
    }
}

void displayMatrix(int matrix[MAX_VERTICES][MAX_VERTICES], int n) {
    printf("\nAdjacency Matrix:\n");
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            printf("%d ", matrix[i][j]);
        }
    }
}
```

```
        printf("\n");
    }
}

int main() {
    int n, e; // n: number of vertices, e: number of edges

    printf("Enter the number of vertices: ");
    scanf("%d", &n);

    int adj[MAX_VERTICES][MAX_VERTICES];
    initializeMatrix(adj, n);

    printf("Enter the number of edges: ");
    scanf("%d", &e);

    printf("Enter the edges (start and end vertices):\n");
    for (int i = 0; i < e; i++) {
        int start, end;

        scanf("%d %d", &start, &end);

        addEdge(adj, start, end);
    }

    calculateDegree(adj, n);
    displayMatrix(adj, n);
    return 0;
}
```

**Input/Output**

Enter the number of vertices: 5

Enter the number of edges: 6

Enter the edges (start and end vertices):

0 1

1 2

2 4

4 3

3 1

4 2

| Vertex | In-Degree | Out-Degree |
|--------|-----------|------------|
|--------|-----------|------------|

|   |   |   |
|---|---|---|
| 0 | 0 | 1 |
|---|---|---|

|   |   |   |
|---|---|---|
| 1 | 2 | 1 |
|---|---|---|

|   |   |   |
|---|---|---|
| 2 | 2 | 1 |
|---|---|---|

|   |   |   |
|---|---|---|
| 3 | 1 | 1 |
|---|---|---|

|   |   |   |
|---|---|---|
| 4 | 1 | 2 |
|---|---|---|

Adjacency Matrix:

0 1 0 0 0

0 0 1 0 0

0 0 0 0 1

0 1 0 0 0

0 0 1 1 0

Write a Program to perform Knapsack Problem using Greedy Solution

**Algorithm**

knapsack(n, weight[], profit[], capacity)

tp=0

u=capacity

for i ← 0 to n do

initialize x[i]=0.0

for i ← 0 to n do

if(weight[i]>u) then

break

else

x[i]=1.0

tp= tp+profit[i]

u=u-weight[i]

End else

End for

if(i<n) then

x[i]=u/weight[i]

tp= tp + (x[i]\*profit[i]);

Print x[i] and tp

End

**Algorithm of main function**

input n,ratio[20],capacity;

for i ← n do

Input weight[i],profit[i]

for i ← n do

ratio[i]=profit[i]/weight[i];

```
End for
for i ← n do
    for j ← i+1 to n do
        if(ratio[i]<ratio[j]) then
            Swap ratio[j] and ratio[i]
            Swap weight[j] and weight[i]
            Swap profit[j] and profit[i]
        End if
    End for
End for
Call knapsack(n, weight, profit, capacity)
End
```

### **Program**

```
# include<stdio.h>
# include<conio.h>

void knapsack(int n, float weight[], float profit[], float capacity)
{
    float x[20], tp= 0;
    int i, j, u;
    u=capacity;
    for (i=0;i<n;i++)
        x[i]=0.0;
    for (i=0;i<n;i++)
    {
        if(weight[i]>u)
            break;
```



```
else
{
    x[i]=1.0;
    tp= tp+profit[i];
    u=u-weight[i];
}
}

if(i<n)
    x[i]=u/weight[i];
tp= tp + (x[i]*profit[i]);
printf("\n The result vector is:- ");
for(i=0;i<n;i++)
    printf("%ft",x[i]);
printf("\n Maximum profit is:- %f", tp);
}

void main()
{
    float weight[20], profit[20], capacity;
    int n, i ,j;
    float ratio[20], temp;
    printf ("\n Enter the no. of objects:- ");
    scanf ("%d", &n);
    printf ("\n Enter the weights and profits of each object\n ");
    for (i=0; i<n; i++)
    {
        scanf("%f %f", &weight[i], &profit[i]);
    }
}
```

```
printf ("\n enter the capacity of knapsack:- ");
```

```
scanf ("%f", &capacity);
```

```
for (i=0; i<n; i++)
```

```
{
```

```
ratio[i]=profit[i]/weight[i];
```

```
}
```

```
for(i=0; i<n; i++)
```

```
{
```

```
    for(j=i+1; j< n; j++)
```

```
    {
```

```
        if(ratio[i]<ratio[j])
```

```
        {
```

```
            temp= ratio[j];
```

```
            ratio[j]= ratio[i];
```

```
            ratio[i]= temp;
```

```
            temp= weight[j];
```

```
            weight[j]= weight[i];
```

```
            weight[i]= temp;
```

```
            temp= profit[j];
```

```
            profit[j]= profit[i];
```

```
            profit[i]= temp;
```

```
        }
```

```
    }
```

```
}
```

```
knapsack(n, weight, profit, capacity);
```

```
getch();  
}
```

### **Input/Output**

Enter the no. of objects:- 3

Enter the weights and profits of each object

14 20

6 16

10 8

enter the capacity of knapsack:- 19

The result vector is:- 1.000000 0.928571 0.000000

Maximum profit is:- 34.571426

### **Input/Output**

Enter the no. of objects:- 7

Enter the weights and profits of each object

1 6

2 10

4 18

5 15

1 3

3 5

7 7

enter the capacity of knapsack:- 15

The result vector is:- 1.000000 1.000000 1.000000 1.000000  
1.000000 0.666667 0.000000

Maximum profit is:- 55.333332

Write a Program to implement greedy algorithm for Job Sequencing with Deadlines.

### **Algorithm**

```
Read n, max=0
    for i ← 0 to n do
        Read p[i]
    End for
    for i ← 0 to n do
        Read d[i]
    End for
    for i ← 0 to n do
        for j ← i+1 to n do
            if(p[i]<p[j]) then
                Swap p[i] and p[j]
                Swap d[i] and d[j]
            End if
        End for
    End for
    Print "Profit in descending order with deadline"
    for i ← 0 to n
        Print p[i],d[i]
    End for
    for i ← 0 to n do
        slot[i]=0
        for i ← 0 to n do
            for j ← d[i] to 0 do
                if(check(slot,j)==1) then
                    slot[i]=j
                End if
            End for
        End for
    End for
    break
```

```
End if
End for
for i ← 0 to n do
    if(slot[i]>0) then
        Print job, profit, deadline and slot allotted
        max=max+p[i];
    End if
else
    Print REJECTED with job, profit and deadline
End for
Print max
end
```

**Algorithm for check function**

```
check(s[], p)
Initialize ptr=0
for i ← 0 to n do
    if(s[i]==p) then
        Increment ptr by one unit
    End for
    if(ptr==0) then
        return 1
    else
        return 0
    End if
End for
```

### Program

```
#include<stdio.h>

#include<conio.h>

int n,i,j,k,t;

int check(int s[],int p)
{
    int ptr=0,i;
    for(i=0;i<n;i++)
    {
        if(s[i]==p)
        ptr++;
    }
    if(ptr==0)
    return 1;
    else
    return 0;
}

void main()
{
    int slot[10],profit,p[10],d[10],max=0;
    printf("Enter the no of jobs : ");
    scanf("%d",&n);

    printf("\n Enter the profit of job\n");
    for(i=0;i<n;i++)
    {
        scanf("%d",&p[i]);
    }

    printf("\n Enter the deadline of job\n");
```

```
for(i=0;i<n;i++)
{
scanf("%d",&d[i]);
}

for(i=0;i<n;i++)
for(j=i+1;j<n;j++)
if(p[i]<p[j])
{
    t=p[i];
    p[i]=p[j];
    p[j]=t;

    t=d[i];
    d[i]=d[j];
    d[j]=t;
}

printf("Profit in descending order with deadline\n");
for(i=0;i<n;i++)
{
printf("%d:%d\n",p[i],d[i]);
}

for(i=0;i<n;i++)
slot[i]=0;

for(i=0;i<n;i++)
for(j=d[i];j>0;j--)
{
    if(check(slot,j)==1)
```

```
{
    slot[i]=j;
break;
}
}
printf("\n\n INDEX PROFIT DEADLINE SLOT ALLOTTED ");
for(i=0;i<n;i++)
{
    if(slot[i]>0)
    {
printf("\n\n %d\t%d\t%d\t[%d - %d]", i+1,p[i],d[i],(slot[i]-1),slot[i]);

max=max+p[i];
    }
else
printf("\n\n %d\t %d\t %d\t REJECTED", i+1,p[i],d[i]);
}
printf("\nTotal profit=%d",max);
getch();
}
```



**Input/Output**

Enter the no of jobs : 7

Enter the profit of job

3 5 20 18 1 6 30

Enter the deadline of job

1 3 4 3 2 1 2

Profit in descending order with deadline

30:2

20:4

18:3

6:1

5:3

3:1

1:2

INDEX PROFIT DEADLINE SLOT ALLOTTED

|   |    |   |          |
|---|----|---|----------|
| 1 | 30 | 2 | [1 - 2]  |
| 2 | 20 | 4 | [3 - 4]  |
| 3 | 18 | 3 | [2 - 3]  |
| 4 | 6  | 1 | [0 - 1]  |
| 5 | 5  | 3 | REJECTED |
| 6 | 3  | 1 | REJECTED |
| 7 | 1  | 2 | REJECTED |

Total profit=74

Write Program that implements Prim's Algorithm to generate Minimum Cost Spanning Tree.

**Algorithm**

Read n, mincost=0, ne=1

for i ← 1 to n do

for j ← 1 to n do

Read cost[i][j]

if(cost[i][j]==0) then

cost[i][j]=999

End for

for i ← 2 to n do

Initialize visited[i]=0

Make visited[1]=1

while(ne<n)

{

for i ← 1 to 999 do

for j ← 1 to n do

if(cost[i][j]<min)

if(visited[i]==0)

continue

else

min=cost[i][j]

u=i

v=j

End else

End for

End for

```
if(visited[u]==0||visited[v]==0) then
```

```
Print ne++,u,v, and min
```

```
mincost+=min
```

```
visited[v]=1
```

```
End if
```

```
cost[u][v]=cost[v][u]=999;
```

```
End while
```

```
Print mincost
```

```
End
```

### **Program**

```
#include<stdio.h>
```

```
#include<conio.h>
```

```
int i,j,k,v,u,n,ne=1;
```

```
int visited[9],min,mincost=0,cost[9][9];
```

```
void main( )
```

```
{
```

```
printf("Enter the number of vertices\n\n");
```

```
scanf("%d",&n);
```

```
printf("Enter the cost matrix\n\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;
```

```
}
```

```
for(i=2;i<=n;i++)
```

```
visited[i]=0;

printf("The edges of the spanning tree are \n\n");

visited[1]=1;

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)
continue;
else
{
min=cost[i][j];
u=i;
v=j;
}
}
}

if(visited[u]==0||visited[v]==0)
{
printf("%d\t Edge\t(%d,%d)=%d\n",ne++,u,v,min);
mincost+=min;
visited[v]=1;
```

```
}  
cost[u][v]=cost[v][u]=999;  
}  
printf("\n\t\t\tMINCOST=%d\n",mincost);  
getch( );  
}
```

### **Input/Output**

Enter the number of vertices

5

Enter the cost matrix

999 10 5 999 4

10 999 3 12 6

5 3 999 9 999

999 12 9 999 6

4 6 999 6 999

The edges of the spanning tree are

1      Edge (1,5)=4

2      Edge (1,3)=5

3      Edge (3,2)=3

4      Edge (5,4)=6

MINCOST=18

Write a Program that implements Kruskal's Algorithm to generate minimum cost spanning tree.

**Algorithm**

Global declaration visited[9],min,mincost=0,ne=1,

,cost[9][9],parent[9]

Read n

for i←1 to n do

for j ←1 to n do

Read cost[i][j];

if(cost[i][j]==0) then

Initializen cost[i][j]=999

End for

while(ne<n) then

{

for i ←1 to n and min=999 do

for j ←1 to n do

if(cost[i][j]<min) then

min=cost[i][j]

a=u=i;

b=v=j;

End if

End for

if(parent[u]) then

u=parent[u]

if(parent[v]) then

v=parent[v]

if(u!=v) then

```
Print ne++,a,b,min
mincost+=min
parent[v]=u
End if
cost[a][b]=cost[b][a]=999;
End while
Print mincost
End
```

### **Program**

```
#include<stdio.h>
#include<conio.h>
int i,j,k,a,b,v,u,n,ne=1;
int visited[9],min,mincost=0,cost[9][9],parent[9];
void main( )
{
printf("Enter the number of vertices\n\n");
scanf("%d",&n);
printf("Enter the cost matrix\n\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;
}
printf("The edged of the spanning tree are\n\n");
while(ne<n)
{
```

```
for(i=1,min=999;i<=n;i++)
for(j=1;j<=n;j++)
{
if(cost[i][j]<min)
{
min=cost[i][j];

a=u=i;
b=v=j;
}
}
if(parent[u] u=parent[u];
if(parent[v] v=parent[v];
if(u!=v)
{
printf("%d\tEdge\t(%d,%d)=%d\n",ne++,a,b,min);
mincost+=min;
parent[v]=u;
}
cost[a][b]=cost[b][a]=999;
}
printf("\n\t\t\tMINCOST=%d\n",mincost);
getch( );
}
```



**Input/Output**

Enter the number of vertices

6

Enter the cost matrix

0 60 10 999 999 999

60 0 999 20 40 70

10 999 0 999 999 50

999 20 999 0 999 80

999 40 999 999 0 30

999 70 50 80 30 0

The edged of the spanning tree are

1      Edge (1,3)=10

2      Edge (2,4)=20

3      Edge (5,6)=30

4      Edge (2,5)=40

5      Edge (3,6)=50

MINCOST=150

