# KCT COLLEGE OF ENGG AND TECHNOLOGY

DEPTT. OF COMPUTER SCIENCE AND ENGG.
LAB MANUAL

**Subject: Design and analysis of Algorithms**

**Semester-5th**

| S.NO. | Name of program |
|---|---|
| 1 | Write a code and analyze to sort an array of integers using merge sort. |
| 2 | Write a code and analyze to sort an array of integers using quick sort. |
| 3 | Write a code and analyze to find the median element in an array of integers. |
| 4 | Write a code and analyze to sort an array of integers using heap sort |
| 5 | From a given vertex in a weighted connected graph, find shortest paths to other vertices using dijkstra's algorithm. |
| 6 | Find minimum cost spanning tree of a given undirected graph using kruskal's algorithm. |
| 7 | Obtain the topological ordering of vertices in a given digraph. |
| 8 | Code and analyze to find the majority element in an array of integers. |
| 9 | Code and analyze to find all occurrences of a pattern p in a given string S. |
| 10 | Code and analyze to compute the convex hull of a set of points in the plane. |
| 11 | Code and analyze to compute the greatest common divisor (GCD) of two numbers. |
| 12 | Code and analyze to find an optimal solution to matrix chain multiplication using dynamic programming. |

1.Code and analyze to sort an array of integers using Merge sort!.

```
 #include<iostream.h>
#include<conio.h>
#include<time.h>
void merge(int *,int,int,int);

void ms(int *,int ,int );//Divide data into subparts and merge them
int main()
{
//clrscr();
int n, a[10], i;
clock_t st,et;
double ts;
cout<<"Enter the number of elemets =";
cin>>n;
cout<<" the elements of arry for sorting = ";
for(i=0;i<n;i++)
{
cin>>a[i];
}
st=clock();
ms(a,0,n-1);
et=clock();
ts=(double)(et-st)/CLOCKS_PER_SEC;
cout<<"Sorted elements : ";
for(i=0;i<n;i++)
cout<<a[i]<<' ';
cout<<"The time taken is"<<ts;
getch();
}
void ms(int *a,int low,int high)
{
if(low<high)
{
```

```
int mid=(low+high)/2;
ms(a,low,mid);
ms(a,mid+1,high);
merge(a,low,mid,high);
}
}
void merge(int *a, int low, int mid,int high)
{
int temp[10];
int h=low,i=low,j=mid+1;
while(i<=mid&&j<=high)
{
if(a[i]<=a[j])
{
temp[h++]=a[i++];
}
else
temp[h++]=a[j++];
}
if(i>mid)
{
for(int k=j;k<=high;k++)
temp[h++]=a[k];
}
else
{
for(int k=i;k<=mid;k++)
temp[h++]=a[k];
}
for(int k=low;k<=high;k++)
{
a[k]=temp[k];
}
}
```

2.Code and analyze to son an array of integers using Quick sort.

```
#include<iostream.h>
#include<conio.h>
#include<time.h>
void Exch(int *p, int *q)
{
int temp = *p;
*p = *q;
*q = temp;
}
void QuickSort(int a[], int low, int high)
{
int i, j, key, k;
if(low>=high)
return;
key=low; i=low+1; j=high;
while(i<=j)
{
while ( a[i] <= a[key] ) i=i+1;
while ( a[j] > a[key] ) j=j-1;
if(i<j) Exch(&a[i], &a[j]);
}
Exch(&a[j], &a[key]);
QuickSort(a, low, j-1);
QuickSort(a, j+1, high);
}
int main()
{
int n, a[1000],k;
clock_t st,et;
double ts;
cout<<" Enter How many Numbers: ";
cin>>n;
cout<<endl<<"enter Numbers"<<endl;
for(k=1;k<=n;k++)
```

```
{
cin>>a[k];
}
st=clock();
QuickSort(a, 1, n);
et=clock();
ts=(double)(et-st)/CLOCKS_PER_SEC;
cout<<"Sorted Numbers are:";
cout<<endl;
for(k=1; k<=n; k++)
cout<<"    "<<a[k];
cout<<"The time taken is"<<ts;
getch();
}
```

3.Code and analyze to find the median element in an array of integers.

```
#include<iostream.h>
#include<conio.h>
Int main()
{
 Int a[10], n,i,j,up,lr,temp;
Cout<<”enter no of elements”;
Cin>>n;
Cout<<”enter elements of set”;
For(i=0;i<n;i++)
{
   Cin>>a[i];
}
For(i=0;i<n;i++)
{
  For(j=i+1;j<n;j++)
{
  If(a[i]>a[j])
{
  Temp=a[i]
A[i]=a[j];
A[j]=temp;
}
}
}
Cout<<”ordered set”;
For(i=0;i<n;i++)
{
   Cout<<a[i];
}
If(n%2==0)
{
Lr=n/2;
Up=(n/2)+1;
Cout<<”lower median is at”<<lr<<”which is”<<a[lr-1]<<endl<<”upper
median is at”<<up<<”which is ”<<a[up-1];
}
Else
{
 Lr=(n+1)/2;
```

```
Cout<<”median is at”<<lr;
}
Getch();
}
```

4. Write a code and analyze to sort an array of integers using heap sort

```
#include<iostream>
#include<algorithm>
using namespace std;
int x;
class heap
{
   int a[30],n;// a[30] to store the input
public:
   void read()
   {
      cout<<"Enter the size\n";
      cin>>n;
      x=n;
      cout<<"Enter the elements\n";
      for(int i=1;i<=n;i++)cin>>a[i];
   }
   void heapsort();
   void maxd()
   {
      swap(a[1],a[n]);   //swap first and last element
      n--;                  //decrease size of array by 1
      heapsort();           //call heapsort function
   }
   void disp()
   {
      cout<<"Sorted Array is\n";
      for(int i=1;i<=x;i++)
         cout<<a[i]<<" ";
      cout<<endl;
   }
};
void heap::heapsort()
{
   if(n==0)return;
   int i,k,v,h,j;
   for(i=n/2;i>0;i--) //we start at i=n/2 as it gives index of right most sub-
tree's parent
   {
      k=i;
```

```
        v=a[k];       //this assigns the value of parent node of the sub-tree under
consideration to variable v
        h=0;
        while(!h&&2*k<=n) //checking whether array is heapified
        {
           j=2*k;
           if(j<n)         //checking if given sub-tree has 2 children
           {
              if(a[j]<a[j+1])j++;  //variable j points to higher valued node
between the 2 children
           }
           if (v>=a[j])  //if parent node of the sub-tree is >= to the greater child,
no need to heapify
              h=1;
           else              //else swap contents of parent and child using variables
k and j
           {
              a[k]=a[j];
              k=j;
           }
        }
        a[k]=v;
     }
     maxd();
}
int main()
{
     heap H;//object created
     H.read();//read function to read input
     H.heapsort();//to perform heapsort
     H.disp();//to display sorted array
     return 0;
}
```

Output:

```
Enter the size
5
Enter the elements
5 4 3 2 1
Sorted Array is:
1 2 3 4 5
```

5. Code and analyze to find shortest paths in a graph with positive edge weights using Dijkstra,s algorithm.

```
#include<stdio.h>
#include<conio.h>
#define infinity 999
void dij(int n,int v,int cost[10][10],int dist[100])
{
int i,u,count,w,flag[10],min;
for(i=1;i<=n;i++)
flag[i]=0,dist[i]=cost[v][i];
count=2;
while(count<=n)
{
min=99;
for(w=1;w<=n;w++)
if(dist[w]<min && !flag[w])
min=dist[w],u=w;
flag[u]=1;
count++;
for(w=1;w<=n;w++)
if((dist[u]+cost[u][w]<dist[w]) && !flag[w])
dist[w]=dist[u]+cost[u][w];
}
}
void main()
{
int n,v,i,j,cost[10][10],dist[10];
clrscr();
printf("\n Enter the number of nodes:");
scanf("%d",&n);
printf("\n Enter the cost 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]=infinity;
```

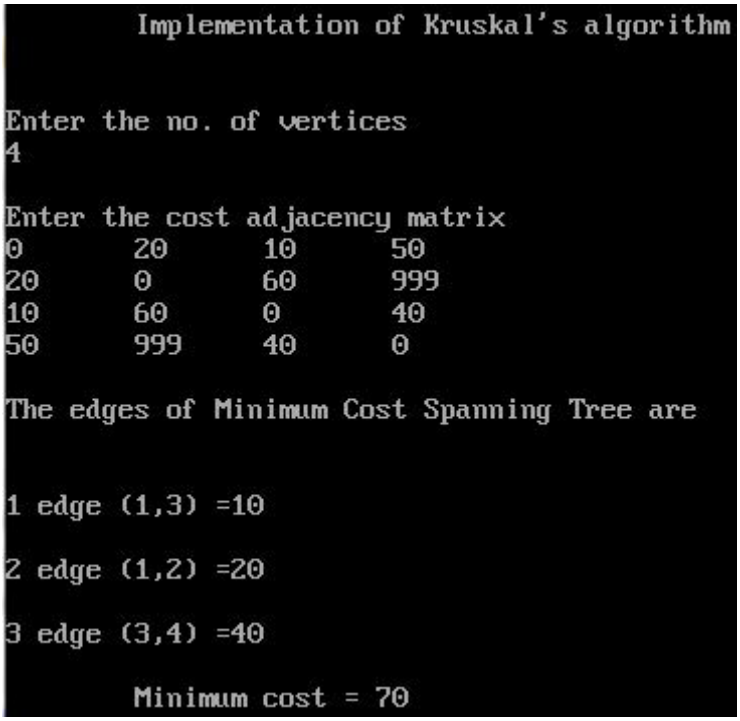
```
}
printf("\n Enter the source matrix:");
scanf("%d",&v);
dij(n,v,cost,dist);
printf("\n Shortest path:\n");
for(i=1;i<=n;i++)
if(i!=v)
printf("%d->%d,cost=%d\n",v,i,dist[i]);
getch();

}
```

```
 Enter the number of nodes:5

 Enter the cost matrix:
0       5       12      17      999
999     0       999     8       7
999     999     0       9       999
999     999     999     0       999
999     999     999     999     0

 Enter the source matrix:1

 Shortest path:
1->2,cost=5
1->3,cost=12
1->4,cost=13
1->5,cost=12
```

6. **Find Minimum Cost Spanning Tree of a given undirected graph using Kruskal's algorithm.**

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
int i,j,k,a,b,u,v,n,ne=1;
int min,mincost=0,cost[9][9],parent[9];
int find(int);
int uni(int,int);
void main()
{
clrscr();
printf("\n\n\tImplementation of Kruskal's algorithm\n\n");
printf("\nEnter the no. of vertices\n");
scanf("%d",&n);
printf("\nEnter the cost adjacency 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;
}
}
printf("\nThe edges of Minimum Cost 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;
}
```

```
}
}
u=find(u);
v=find(v);
if(uni(u,v))
{
printf("\n%d edge (%d,%d) =%d\n",ne++,a,b,min);
mincost +=min;
}
cost[a][b]=cost[b][a]=999;
}
printf("\n\tMinimum cost = %d\n",mincost);
getch();
}
int find(int i)
{
while(parent[i])
i=parent[i];
return i;
}
int uni(int i,int j)
{
if(i!=j)
{
parent[j]=i;
return 1;
}
return 0;

}
```

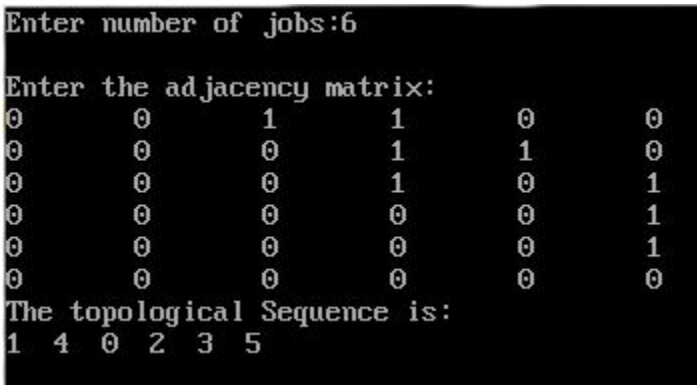

Implementation of Kruskal's algorithm
Enter the no. of vertices
4
Enter the cost adjacency matrix
0 20 10 50
20 0 60 999
10 60 0 40
50 999 40 0
The edges of Minimum Cost Spanning Tree are
1 edge (1,3) =10
2 edge (1,2) =20
3 edge (3,4) =40
Minimum cost = 70

**7.Obtain the Topological ordering of vertices in a given digraph.**

```
#include<iostream.h>
#include<conio.h>
int a[10][10],n,indegre[10];
void find_indegre()
{ int j,i,sum;
for(j=0;j<n;j++)
{
sum=0;
for(i=0;i<n;i++)
sum+=a[i][j];
indegre[j]=sum;
}
}
void topology()
{
int i,u,v,t[10],s[10],top=-1,k=0;
find_indegre();
for(i=0;i<n;i++)
{
if(indegre[i]==0) s[++top]=i;
}
while(top!=-1)
{
u=s[top--];
t[k++]=u;
for(v=0;v<n;v++)
{
if(a[u][v]==1)
{
indegre[v]--;
if(indegre[v]==0) s[++top]=v;
}
}
}
Cout<<"The topological Sequence is:\n";
for(i=0;i<n;i++)
cout<<t[i];
```

```
}
void main()
{
int i,j;
clrscr();
cout<<"Enter number of jobs:";
cin>>n;
cout<<"\nEnter the adjacency matrix:\n";
for(i=0;i<n;i++)
{
for(j=0;j<n;j++)
cin>>a[i][j];
}
topology();
getch();
}
```

**Output:**



8. Code and analyze to find the majority element in an array of integers.

```
# include <iostream.h>
# include <stdbool.h>
bool isMajority(int arr[], int n, int x)
{
 int i;
int lastindex = n%2? (n/2+1): (n/2);
  for (i = 0; i < lastindex; i++)
  {
   if (arr[i] == x && arr[i+n/2] == x)
      return 1;
  }
 return 0;
}

Void  main()
{
  int arr[] ={1, 2, 3, 4, 4, 4, 4};
  int n = sizeof(arr)/sizeof(arr[0]);
  int x = 4;
  if (isMajority(arr, n, x))
    cout<<x<<" appears more than"<<n/2<<" times in arr[]";
  else
     cout<<x<<" does not appears more than"<<n/2<<" times in arr[]";

  getchar();
}
```

**Time Complexity:** O(n)

9. Code and analyze to find all occurrences of a pattern p in a given string S.

```
#include <string.h>
#include <iostream.h>
using namespace std;
void  main()
{
      string str1 = "1,2,3,1,2,1,2,2,1,2," ;
      string str2 = "1,2," ;
      int count = 0 ;
      int pos = -4;
      while( (pos = str1.find(str2, pos+4) ) != -1 )
      {
         ++count ;
      }
      cout << count ;
      getch();
}
```

10. Code and analyze to compute the convex hull of a set of points in the plane.

```
#include <iostream.h>
#include <stack.h>
#include <stdlib.h>
using namespace std;

struct Point
{
    int x;
    int y;
};
Point nextToTop(stack<Point> &S)
{
    Point p = S.top();
    S.pop();
    Point res = S.top();
    S.push(p);
    return res;
}
int swap(Point &p1, Point &p2)
{
    Point temp = p1;
    p1 = p2;
    p2 = temp;
}
int dist(Point p1, Point p2)
{
    return (p1.x - p2.x)*(p1.x - p2.x) + (p1.y - p2.y)*(p1.y - p2.y);
}
int orientation(Point p, Point q, Point r)
{
    int val = (q.y - p.y) * (r.x - q.x) -
              (q.x - p.x) * (r.y - q.y);

    if (val == 0) return 0;  // colinear
    return (val > 0)? 1: 2; // clock or counterclock wise
}
int compare(const void *vp1, const void *vp2)
{
```

```
  Point *p1 = (Point *)vp1;
  Point *p2 = (Point *)vp2;
  int o = orientation(p0, *p1, *p2);
  if (o == 0)
    return (dist(p0, *p2) >= dist(p0, *p1))? -1 : 1;

  return (o == 2)? -1: 1;
}
void convexHull(Point points[], int n)
{
  int ymin = points[0].y, min = 0;
  for (int i = 1; i < n; i++)
  {
    int y = points[i].y;
    if ((y < ymin) || (ymin == y && points[i].x < points[min].x))
       ymin = points[i].y, min = i;
  }
  swap(points[0], points[min]);
  p0 = points[0];
  qsort(&points[1], n-1, sizeof(Point), compare);
  stack<Point> S;
  S.push(points[0]);
  S.push(points[1]);
  S.push(points[2]);
  for (int i = 3; i < n; i++)
  {
 while (orientation(nextToTop(S), S.top(), points[i]) != 2)
        S.pop();
     S.push(points[i]);
  }
 while (!S.empty())
  {
     Point p = S.top();
     cout << "(" << p.x << ", " << p.y <<")" << endl;
     S.pop();
  }
}
int main()
{
   Point points[] = {{0, 3}, {1, 1}, {2, 2}, {4, 4},
```

```
                    {0, 0}, {1, 2}, {3, 1}, {3, 3}};
    int n = sizeof(points)/sizeof(points[0]);
    convexHull(points, n);
    return 0;
}
```

Output:

```
(0, 3)
(4, 4)
(3, 1)
(0, 0)
```

**Time Complexity:** Let n be the number of input points. The algorithm takes O(nLogn) time if we use a O(nLogn) sorting algorithm.
The first step (finding the bottom-most point) takes O(n) time. The second step (sorting points) takes O(nLogn) time. In third step, every element is pushed and popped at most one time. So the third step to process points one by one takes O(n) time, assuming that the stack operations take O(1) time.
Overall complexity is O(n) + O(nLogn) + O(n) which is O(nLogn)

11. Code and analyze to compute the greatest common divisor (GCD) of two numbers.

```
#include<iostream.h>
 #include<conio.h>
Void  main(){

   int n1,n2;

cout<<”Insert any two number”;
   cin>>n1>>n2;
   while(n1!=n2)
        {
        If(n1>=n2-1)
         N1=n1-n2;
        Else
          N2=n2-n1;
   }
Cout<<”gcd is”<<n1;
Getch();
}
```

12. Code and analyze to find an optimal solution to matrix chain multiplication using dynamic programming.

```
#include<iostream.h>
#include<limits.h>
 int MatrixChain(int p[], int n)
{
    int m[n][n];
    int i, j, k, L, q;
    for (i = 1; i < n; i++)
        m[i][i] = 0;
     for (L=2; L<n; L++)
    {
        for (i=1; i<=n-L+1; i++)
        {
            j = i+L-1;
            m[i][j] = INT_MAX;
            for (k=i; k<=j-1; k++)
            {
                q = m[i][k] + m[k+1][j] + p[i-1]*p[k]*p[j];
               if (q < m[i][j])
                   m[i][j] = q;
            }
        }
    }

    return m[1][n-1];
}
 Void  main()
{
    int arr[] = {1, 2, 3, 4};
    int size = sizeof(arr)/sizeof(arr[0]);
    cout<<"Minimum number of multiplications is "<<MatrixChain(arr, size);
    getch();
}
```

Time Complexity: O(n^3)
Auxiliary Space: O(n^2)