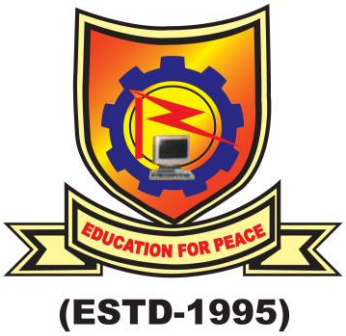
DESIGN AND ANALYSIS OF ALGORITHMS LAB

(A0598194)

LAB MANUAL

RGM R-19 Regulations



Department of Computer Science and Engineering

Rajeev Gandhi Memorial College of Engineering and Technology (Autonomous)

Nandyal, Kurnool – Dist. A.P.

VISION OF THE DEPARTMENT

- To empower students with cutting edge technologies in computer science and engineering
- To train the students as entrepreneurs in computer science and engineering to address the needs of the society
- To develop smart applications to disseminate information to rural people

MISSION OF THE DEPARTMENT

- To become the best computer science and engineering department in the region offering undergraduate, post graduate and research programs in collaboration with industry
- To incubate, apply and spread innovative ideas by collaborating with relevant industries and R & D labs through focused research groups.
- To provide exposure to the students in the latest tools and technologies to develop smart applications for the society

R G M COLLEGE OF ENGINEERING AND TECHNOLOGY
AUTONOMOUS
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

`II B.Tech. II-Sem (CSE) P C
3 1.5

(A0598194) DESIGN AND ANALYSIS OF ALGORITHMS LAB

COURSE OBJECTIVES:

- The principle objective of this course is to build solid foundation in algorithms and their applications.
- To implement various divide and conquer techniques examples.
- To implement various Greedy techniques examples.
- To implement various Dynamic Programming techniques examples.
- To provide a practical exposure of all algorithms.
- To understand the importance of algorithm and its complexities.

COURSE OUTCOMES:

- Students will be able to calculate the time complexity of algorithm.
- Students will be able to sort the given numbers using various sorting algorithms.
- Students will be able to write programs for the problems using Divide and Conquer.
- Students will be able to write programs for the problems using Greedy Method.
- Students will be able to write programs for the problems using Dynamic programming.
- Students will be able to write programs for the problems using Backtracking.

MAPPING OF COs & POs

CO/PO	PO1	PO2	PO3	PO 4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO 3
CO1	2	2			2				2	3					
CO2	2	2			2				2	2		1			
CO3	3	2		1	3				2	1					
CO4	2	3			2	1			2	1		1			
CO5	3	2			2				2	2					
CO6	2	2	1		2				2	2					

EXPERIMENTS

1. Write a program to perform operation count for a given pseudo code
2. Write a program to perform Bubble sort for any given list of numbers.
3. Write a program to perform Insertion sort for any given list of numbers.
4. Write a program to perform Quick Sort for the given list of integer values.
5. Write a program to find Maximum and Minimum of the given set of integer values.
6. Write a Program to perform Merge Sort on the given two lists of integer values.
7. Write a Program to perform Binary Search for a given set of integer values recursively and non-recursively.
8. Write a program to find solution for knapsack problem using greedy method.
9. Write a program to find minimum cost spanning tree using Prim's Algorithm.
10. Write a program to find minimum cost spanning tree using Kruskal's Algorithm.
11. Write a program to perform Single source shortest path problem for a given graph.
12. Write a program to find solution for job sequencing with deadlines problem.
13. Write a program for all pairs shortest path problem.
14. Write a program to solve N-QUEENS problem.
15. Write a program to solve Sum of subsets problem for a given set of distinct numbers.

REFERENCES:

1. Data Structures and Algorithms by G.A.V. Pai, 2017, TMH.
2. Fundamentals of Computer Algorithms by Ellis Horowitz, Sartaj Sahni and Sanguthevar Rajasekaran, 2nd edition, University Press.

Experiment No: 1

Aim: Write a program to perform operation count for a given pseudo code

```
#include<stdio.h>
//#include<conio.h>
void main()
{
  int count=0,sum=0,n,i,a[50];
  //clrscr();
  count=count+1;
  printf("\n Enter the n value:");
  scanf("%d",&n);
  count=count+1;
  printf("\n Enter %d values to sum:",n);
  for(i=0;i<n;i++)
  {
    count=count+1;
    scanf("%d",&a[i]);
  }
  count=count+1;
  for(i=0;i<n;i++)
  {
    count=count+1;
    sum=sum+a[i];
    count=count+1;
  }
  count=count+1;
  printf("\n The of %d values is:%d and count is=%d",n,sum,count);
  //getch();
}
```

Output:

```
Enter the n value:5

Enter 5 values to sum:1 2 3 4 5
The of 5 values is:15 and count is=19
```

Experiment No: 2

Aim: Write a program to perform Bubble sort for any given list of numbers.

Program:

```
#include<stdio.h>
#include<conio.h>
void bubblesort(int[],int);
void display(int[],int);
int main()
{
  int a[20],n,i;
  clrscr();
  printf("\n Enter the number of elements in array are:");
  scanf("%d",&n);
  printf("\n Enter %d elements in the array:",n);
  for(i=0;i<n;i++)
     scanf("%d",&a[i]);
  bubblesort(a,n);
  printf("\n The sorted elements in the array are:");
  display(a,n);
  getch();
  return 0;
}
void bubblesort(int a[],int n)
{
   int i,j,temp,excg=0;
   int last=n-1;
   for(i=0;i<n;i++)
   {
      excg=0;
      for(j=0;j<last;j++)
      {
              if(a[j]>a[j+1])
              {
               temp=a[j];
               a[j]=a[j+1];
               a[j+1]=temp;
               excg++;
              }
      }
   }
   if(excg==0)
     return ;
   else
    last=last-1;
```

```
}
void display(int a[],int n)
{
  int i;
  for(i=0;i<n;i++)
    printf("%d \t",a[i]);
}
```

Output:

```
Enter the number of elements in array are:7

 Enter 7 elements in the array:7 8 9 5 4 2 1

 The sorted elements in the array are:1     2    4    5    7    8    9
```

Experiment No: 3

Aim: Write a program to perform Insertion sort for any given list of numbers.

Program:

```
#include<stdio.h>
#include<conio.h>
void inssort(int[],int);
void display(int[],int);
int main()
{
  int a[20],n,i;
  clrscr();
  printf("\n Enter the number of elements in array are:");
  scanf("%d",&n);
  printf("\n Enter %d elements in the array:",n);
  for(i=0;i<n;i++)
     scanf("%d",&a[i]);
  inssort(a,n);
  printf("\n The sorted elements in the array are:");
  display(a,n);
  getch();
  return 0;
}
void inssort(int a[],int n)
{
   int i,j,index=0;
   for(i=1;i<n;i++)
   {
     index=a[i];
     j=i;
     while((j>0)&&(a[j-1]>index))
     {
       a[j]=a[j-1];
       j--;
     }
     a[j]=index;
   }
}
void display(int a[],int n)
{
   int i;
   for(i=0;i<n;i++)
   {
     printf("%d\t",a[i]);
   }
}
```

Output:

```
Enter the number of elements in array are:7

 Enter 7 elements in the array:33 56 98 12 34 9 4

 The sorted elements in the array are:4  9  12  33  34  56  98
```

Experiment No: 4

Aim: Write a program to perform Quick sort for any given list of numbers.

Program:

```
#include<stdio.h>
#include<conio.h>
void qsort(int [],int,int);
int partition(int [],int,int);
void qsort(int a[],int first,int last)
{
  int j;
  if(first<last)
  {
    j=partition(a,first,last+1);
    qsort(a,first,j-1);
    qsort(a,j+1,last);
  }
}
int partition(int a[],int first,int last)
{
  int v=a[first];
  int i=first;
  int j=last;
  int temp=0;
  do
  {
    do
    {
      i++;
    }while(a[i]<v);
    do
    {
      j--;
    }while(a[j]>v);
    if(i<j)
    {
       temp=a[i];
       a[i]=a[j];
       a[j]=temp;
    }
  }while(i<j);
  a[first]=a[j];
  a[j]=v;
  return j;
}
```

```
int main()
{
  int a[40],i,n;
  clrscr();
  printf("\n Enter the no of elements (size):");
  scanf("%d",&n);
  printf("\n Enter the ELements to sort:");
  for(i=0;i<n;i++)
    scanf("%d",&a[i]);
  qsort(a,0,n-1);
  printf("\n The ELements after sorting are:");
  for(i=0;i<n;i++)
  {
    printf("%d\t",a[i]);
  }
  getch();
  return 0;
}
```

Output:

```
Enter the no of elements (size):6

 Enter the ELements to sort:98 45 21 34 90 43

 The ELements after sorting are:21  34  43  45  90  98
```

Experiment No: 5

Aim: Write a program to find Maximum and Minimum of the given set of integer values.

Program:

```
#include<stdio.h>
#include<conio.h>
void minmax(int,int,int,int);
int i,j,a[50],n,fmax,fmin;
int main()
{
  clrscr();
  printf("\n Enter the number of elements in array are:");
  scanf("%d",&n);
  printf("\n Enter %d elements in the array:",n);
  for(i=0;i<n;i++)
     scanf("%d",&a[i]);
  printf("\n The Elements in the array are:");
  for(i=0;i<n;i++)
     printf("%d\n",a[i]);
  //fmax=fmin=a[0];
  minmax(0,n-1,a[0],a[0]);
  printf("\n The minimum Element of the list of elements is:%d",fmin);
  printf("\n The maximum Element of the list of elements is:%d",fmax);
  getch();
  return 0;
}
void minmax(int i,int j,int max,int min)
{
   int gmax,gmin,hmax,hmin;
   gmax=hmax=max;
   gmin=hmin=min;
   if(i==j)
   {
     fmax=fmin=a[i];
   }
   else if(i==(j-1))
   {
     if(a[i]>a[j])
     {
            fmax=a[i];
            fmin=a[j];
     }
     else
     {
       fmax=a[j];
```

```
      fmin=a[i];
    }
  }
  else
  {
   int mid=(i+j)/2;
   minmax(i,mid,a[i],a[i]);
   gmax=fmax;
   gmin=fmin;
   minmax(mid+1,j,a[mid+1],a[mid+1]);
   hmax=fmax;
   hmin=fmin;
   if(gmax>hmax)
   {
    fmax=gmax;
   }
   else
   {
    fmax=hmax;
   }
   if(gmin>hmin)
   {
    fmin=hmin;
   }
   else
   {
    fmin=gmin;
   }
  }
}
```

Output:

```
Enter the number of elements in array are:7
Enter 7 elements in the array:2 5 1 6 88 99 22


 The Elements in the array are:2
5
1
6
88
99
22

 The minimum Element of the list of elements is:1
 The maximum Element of the list of elements is:99
```

Experiment No: 6

Aim: Write a Program to perform Merge Sort on the given two lists of integer values.

Program:

```
#include<stdio.h>
#include<conio.h>
void merge(int[],int,int,int);
void mergesort(int[], int,int);
void merge(int a[25], int low, int mid, int high)
{
  int b[25],h,i,j,k;
  h=low;
  i=low;
  j=mid+1;
  while((h<=mid)&&(j<=high))
  {
    if(a[h]<a[j])
    {
      b[i]=a[h];
      h++;
    }
    else
    {
      b[i]=a[j];
      j++;
    }
    i++;
  }
  if(h>mid)
  {
   for(k=j;k<=high;k++)
   {
     b[i]=a[k];
     i++;
   }
  }
  else
  {
   for(k=h;k<=mid;k++)
   {
     b[i]=a[k];
     i++;
   }
  }
  for(k=low;k<=high;k++)
```

```
  {
    a[k]=b[k];
  }
}
void mergesort(int a[25],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);
  }
}
void main()
{
  int a[25],i,n;
  clrscr();
  printf("\n Enter the size of the elements to be sorted:");
  scanf("%d",&n);
  printf("\n Enter the elements to sort:");
  for(i=0;i<n;i++)
    scanf("%d",&a[i]);
  printf("\n The Elements before sorting are:");
  for(i=0;i<n;i++)
     printf("%d\t",a[i]);
  mergesort(a, 0,n-1);
  printf("\n The elements after sorting are:");
  for(i=0;i<n;i++)
     printf("%d\t",a[i]);
  getch();
}
```

Output:

```
Enter the size of the elements to be sorted:7

Enter the elements to sort:33 44 77 22 11 0 9

The Elements before sorting are:33 44  77  22  11  0   9
The elements after sorting are:0   9   11  22  33  44  77
```

Experiment No: 7

Aim: Write a Program to perform Binary Search for a given set of integer values recursively and non-recursively.

Program:

```
/* Aim: To write a C program to demonstrate the Binary Search */
#include<stdio.h>
#include<conio.h>
void bubblesort(int[],int);
int binsrch(int[],int,int,int);
void display(int[],int);
int i,j;
int main()
{
  int a[20],n,key,pos=-1;
  clrscr();
  printf("\n Enter the number of elements in array are:");
  scanf("%d",&n);
  printf("\n Enter %d elements in the array:",n);
  for(i=0;i<n;i++)
     scanf("%d",&a[i]);
  printf("\n Enter the element to be searched:");
  scanf("%d",&key);
  bubblesort(a,n);
  printf("\n The sorted elements in the array are:");
  display(a,n);
  pos=binsrch(a,key,0,n-1);
  if(pos!=-1)
    printf("\n The Element %d is found in position %d",key,pos);
  else
    printf("\n Element not found");
  getch();
  return 0;
}
int binsrch(int a[],int key,int low,int high)
{
    int mid;
    while(low<=high)
    {
       mid=(low+high)/2;
       if(key<a[mid])
                high=mid-1;
       else if(key>a[mid])
               low=mid+1;
```

```
      else
              return mid;
    }
    return -1;
}
void bubblesort(int a[],int n)
{
   int i,j,temp,excg=0;
   int last=n-1;
   for(i=0;i<n;i++)
   {
      excg=0;
      for(j=0;j<last;j++)
      {
             if(a[j]>a[j+1])
             {
              temp=a[j];
              a[j]=a[j+1];
              a[j+1]=temp;
              excg++;
             }
      }
   }
   if(excg==0)
     return ;
   else
    last=last-1;
}
void display(int a[],int n)
{
  int i;
  for(i=0;i<n;i++)
    printf("%d\t",a[i]);
}
```

Output:

1.

```
Enter the number of elements in array are:7
Enter 7 elements in the array:1 7 9 3 5 2 0
Enter the element to be searched:7
The sorted elements in the array are:0  1   2   3   5   7   9
The Element 7 is found in position 5
```

2.

```
Enter the number of elements in array are:7
Enter 7 elements in the array:3 5 7 9 1 2 6
Enter the element to be searched:55

 The sorted elements in the array are:1 2   3   5   6   7   9
 Element not found
```

Experiment No: 8

Aim: Write a program to find solution for knapsack problem using greedy method.

Program:

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

void readf();
void knapsack(int,int);
void dsort(int n);
void display(int);
int p[20],w[20],n,m;
double x[20],d[20],temp,res=0.0,sum=0.0;
void readf()
{
  int m,n,i;
  printf("\n Enter the no of Profits and weights:");
  scanf("%d",&n);
  printf("\n Enter the Maximum Capacity of the Knapsack:");
  scanf("%d",&m);
  printf("\n Enter %d profits of the weights:",n);
  for(i=0;i<n;i++)
    scanf("%d",&p[i]);
  printf("\n Enter %d Weights:",n);
  for(i=0;i<n;i++)
    scanf("%d",&w[i]);
  for(i=0;i<n;i++)
     d[i]=(double)p[i]/w[i];
  dsort(n);
  knapsack(m,n);
  display(n);
}
void dsort(int n)
{
 int i,j,t;
 for(i=0;i<n;i++)
 {
   for(j=0;j<n-1;j++)
   {
     if(d[j]<d[j+1])
     {
            temp=d[j];
            d[j]=d[j+1];
            d[j+1]=temp;
            t=p[j];
```

```
            p[j]=p[j+1];
            p[j+1]=t;
            t=w[j];
            w[j]=w[j+1];
            w[j+1]=t;
      }
    }
  }
}
void display(int n)
{
  int i,m;
  printf("\n The Required Optimal solution is:\n");
  printf("Profits Weights Xvalue\n");
  for(i=0;i<n;i++)
  {
    printf("%d\t%d\t%f\n",p[i],w[i],x[i]);
    sum=sum+(p[i]*x[i]);
    res=res+(w[i]*x[i]);
  }
  printf("\n The Total Resultant Profit is:%f",sum);
  printf("\n The total resultant Weight into the knapsack is:%f",res);
}
void knapsack(int m,int n)
{
  int i,cu=m;
  for(i=0;i<n;i++)
  {
    x[i]=0.0;
  }
  for(i=0;i<n;i++)
  {
    if(w[i]<cu)
    {
     x[i]=1.0;
     cu=cu-w[i];
    }
    else
      break;
  }
  if(i<=n)
  {
     x[i]=(double)cu/w[i];
  }
}
```

```
int main()
{
        clrscr();
        readf();
        getch();
        return 0;

}
```

Output:

```
 Enter the no of Profits and weights:3

 Enter the Maximum Capacity of the Knapsack:20

 Enter 3 profits of the objects:25 24 15

 Enter 3 Weights:18 15 10

 The Required Optimal solution is:
Profits Weights Xvalue
24      15      1.000000
15      10      0.500000
25      18      0.000000

 The Total Resultant Profit is:31.500000
 The total resultant Weight into the knapsack is:20.000000
```

Experiment No: 9

Aim: Write a program to find minimum cost spanning tree using Prim's Algorithm.

Program:

```
#include<stdio.h>
#include<conio.h>
int n,cost[10][10],temp,nears[10];
void readv();
void primsalg();
void readv()
{
  int i,j;
  printf("\n Enter the No of nodes or vertices:");
  scanf("%d",&n);
  printf("\n Enter the Cost Adjacency matrix of the given graph:");
  for(i=1;i<=n;i++)
  {
    for(j=1;j<=n;j++)
    {
      scanf("%d",&cost[i][j]);
      if((cost[i][j]==0) && (i!=j))
      {
            cost[i][j]=999;
      }
    }
  }
}
void primsalg()
{
  int k,l,min,a,t[10][10],u,i,j,mincost=0;
  min=999;
  for(i=1;i<=n;i++)          //To Find the Minimum Edge E(k,l)
  {
    for(u=1;u<=n;u++)
    {
      if(i!=u)
      {
       if(cost[i][u]<min)
       {
            min=cost[i][u];
            k=i;
            l=u;
       }
      }
    }
```

```
  }
  t[1][1]=k;
  t[1][2]=l;
  printf("\n The Minimum Cost Spanning tree is...");
  printf("\n(%d,%d)-->%d",k,l,min);
  for(i=1;i<=n;i++)
  {
    if(i!=k)
    {
     if(cost[i][l]<cost[i][k])
     {
           nears[i]=l;
     }
     else
     {
       nears[i]=k;
     }
    }
   }
   nears[k]=nears[l]=0;
  mincost=min;
 for(i=2;i<=n-1;i++)
 {
   j = findnextindex(cost,nears);
   t[i][1]=j;
   t[i][2]=nears[j];
   printf("\n(%d,%d)-->%d",t[i][1],t[i][2],cost[j][nears[j]]);
   mincost=mincost+cost[j][nears[j]];
   nears[j]=0;
   for(k=1;k<=n;k++)
   {
     if(nears[k]!=0 && cost[k][nears[k]]>cost[k][j])
     {
           nears[k]=j;
     }
   }
 }
  printf("\n The Required Mincost of the Spanning Tree is:%d",mincost);
}
int findnextindex(int cost[10][10],int nears[10])
{
  int min=999,a,k,p;
  for(a=1;a<=n;a++)
  {
    p=nears[a];
    if(p!=0)
```

```
    {
     if(cost[a][p]<min)
     {
           min=cost[a][p];
           k=a;
     }
    }
  }
  return k;
}
void main()
{
  clrscr();
  readv();
  primsalg();
  getch();
}
```

Output:

```
 Enter the No of nodes or vertices:7

 Enter the Cost Adjacency matrix of the given graph:0 28 0 0 0 10 0
28 0 16 0 0 0 14
0 16 0 12 0 0 0
0 0 12 0 22 0 18
0 0 0 22 0 25 24
10 0 0 0 25 0 0
0 14 0 0 24 0 0

 The Minimum Cost Spanning tree is...
(1,6)-->10
(5,6)-->25
(4,5)-->22
(3,4)-->12
(2,3)-->16
(7,2)-->14
 The Required Mincost of the Spanning Tree is:99
```

Experiment No: 10

Aim: Write a program to find minimum cost spanning tree using Kruskal's Algorithm.

Program:

```
#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\tImplementation of Kruskal's algorithm\n");
        printf("\nEnter the no. of vertices:");
        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("The edges of Minimum Cost Spanning Tree are\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("%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;
}
```

Output:

```
        Implementation of Kruskal's algorithm

Enter the no. of vertices:7

Enter the cost adjacency matrix:
0 28 0 0 0 10 0
28 0 16 0 0 0 14
0 16 0 12 0 0 0
0 0 12 0 22 0 18
0 0 0 22 0 25 24
10 0 0 0 25 0 0
0 14 0 0 24 0 0
The edges of Minimum Cost Spanning Tree are
1 edge (1,6) =10
2 edge (3,4) =12
3 edge (2,7) =14
4 edge (2,3) =16
5 edge (4,5) =22
6 edge (5,6) =25

        Minimum cost = 99
```

Experiment No: 11

Aim: Write a program to perform Single source shortest path problem for a given graph.

Program:

```
#include<stdio.h>
#include<conio.h>
void readf();
void SP();
int cost[20][20],dist[20],s[20];
int n,u,min,v,w;
void readf()
{
  int i,j;
  printf("\n Enter the no of vertices:");
  scanf("%d",&n);
  printf("\n Enter the Cost of vertices:");
  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;
   }
  }
}
void SP()
{
  int i,j;
  printf("\n Enter the source vertex:");
  scanf("%d",&v);
  for(i=1;i<=n;i++)
  {
    s[i]=0;
   dist[i]=cost[v][i];
  }
  s[v]=1;
  dist[v]=0;
  for(i=2;i<=n;i++)
  {
    min=dist[i];
    for(j=2;j<=n;j++)
    {
            if(s[j]==0)
            {
```

```
            if(min>dist[j])
            {
              min=dist[j];
              u=j;
            }
          }
    }
    s[u]=1;
    for(w=1;w<=n;w++)
    {
          if(cost[u][w]!=0 && s[w]==0)
          {
            if(dist[w]>(dist[u]+cost[u][w]))
            {
              dist[w]=dist[u]+cost[u][w];
            }
          }
    }
  }
  printf("\n From the Source vertex %d",v);
  for(i=1;i<=n;i++)
     printf("\n%d->%d",i,dist[i]);
}
void main()
{
  clrscr();
  readf();
  SP();
  getch();
}
```

Output:

```
Enter the no of vertices:6
Enter the Cost of vertices:
0 50 45 10 0 0

0 0 10 15 0 0

0 0 0 0 30 0

20 0 0 0 15 0

0 20 35 0 0 0

0 0 0 0 3 0
```

```
 Enter the source vertex:1

 From the Source vertex 1
1->0
2->45
3->45
4->10
5->25
6->999
```

Experiment No: 12

Aim: Write a program to find solution for job sequencing with deadlines problem.

Program:

```
#include<stdio.h>
#include<conio.h>
int jobseq();
void psort();
int tp,j[10],d[10],p[10],n;
void main()
{
   int i,k;
   clrscr();
   printf("\n Enter the n'o of jobs:");
   scanf("%d",&n);
   printf("\n Enter the %d Deadlines for the jobs:",n);
   for(i=1;i<=n;i++)
      scanf("%d",&d[i]);
   printf("\n Enter the Profits required for jobs:");
   for(i=1;i<=n;i++)
      scanf("%d",&p[i]);
   psort();
   for(i=1;i<n;i++)
      printf("%d",p[i]);
   k=jobseq();
   printf("\n The Required Solution is:");
   for(i=1;i<=k;i++)
   {
       tp=tp+p[j[i]];
       printf("%d-->",j[i]);
   }
   printf("\n Profits:%d",tp);
   getch();
}
int jobseq()
{
  int i,k,q;
  d[0]=0;
  j[0]=0;
  j[1]=1;
  k=1;
  for(i=2;i<=n;i++)
  {
    int r=k;
    while((d[j[r]]>d[i]) && (d[j[r]]!=r))
```

```
      r=r-1;
    if((d[j[r]]<=d[i]) && (d[i]>r))
    {
      for(q=k;q>=r+1;q--)
      {
            j[q+1]=j[q];
      }
      j[r+1]=i;
      k=k+1;
    }
  }
  return k;
}
void psort()
{
   int i,k,temp1;
   for(i=1;i<=n;i++)
   {
    for(k=1;k<=n-i;k++)
    {
      if(p[k]<p[k+1])
      {
            temp1=p[k];
            p[k]=p[k+1];
            p[k+1]=temp1;
            temp1=j[k];
            j[k]=j[k+1];
            j[k+1]=temp1;
            temp1=d[k];
            d[k]=d[k+1];
            d[k+1]=temp1;
      }
     }
   }
}
```

Output:

1.

```
Enter the n'o of jobs:9

 Enter the 9 Deadlines for the jobs:5 3 2 2 3 4 7 7 5
 Enter the Profits required for jobs:30 25 23 20 18 18 16 15 10
 The Required Solution is:3-->4-->2-->6-->1-->7-->8-->
 Profits:147
```

2.

```
Enter the n'o of jobs:5
Enter the 5 Deadlines for the jobs:2 2 1 1 3
Enter the Profits required for jobs:100 90 70 50 40

 The Required Solution is:1-->2-->5-->
 Profits:230
```

Experiment No: 13

Aim: Write a program for all pairs shortest path problem.

Program:

```
#include<stdio.h>
#include<conio.h>
void readf();
void amin();
int cost[20][20],a[20][20];
int i,j,k,n;
void readf()
{
  printf("\n Enter the no of vertices:");
  scanf("%d",&n);
  printf("\n Enter the Cost of vertices:");
  for(i=0;i<n;i++)
  {
   for(j=0;j<n;j++)
   {
            scanf("%d",&cost[i][j]);
            if(cost[i][j]==0 && (i!=j))
                cost[i][j]=999;
            a[i][j]=cost[i][j];
   }
  }
}
void amin()
{
  for(k=0;k<n;k++)
  {
    for(i=0;i<n;i++)
    {
            for(j=0;j<n;j++)
            {
             if(a[i][j]>a[i][k]+a[k][j])
             {
               a[i][j]=a[i][k]+a[k][j];
             }
            }
    }
  }
  printf("\n The All pair shortest path is:");
  for(i=0;i<n;i++)
  {
   printf("\n");
```

```
    for(j=0;j<n;j++)
    {
      printf("%d\t",a[i][j]);
    }
   }
}
void main()
{
  clrscr();
  readf();
  amin();
  getch();
}
```

Output:

```
Enter the no of vertices:3
Enter the Cost of vertices:
0 4 11
6 0 2
3 0 0
The All pair shortest path is:
0   4   6
5   0   2
3   7   0
```

Experiment No: 14

Aim: Write a program to solve N-QUEENS problem.

Program:

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
void readv();
void nqueen(int,int);
int place(int,int);
int x[25],count=0;
void readv()
{
  int n;
  printf("\n Enter the no of Queens to be placed:");
  scanf("%d",&n);
  printf("\n The Places in which the %d Queens are to placed in the %dx%d ChessBoard
is:",n,n,n);
  nqueen(1,n);
  printf("\n The No of Solutions for the %d Queens Problem are:%d",n,count);
}
void nqueen(int k,int n)
{
  int i,j;
  for(i=1;i<=n;i++)
  {
     if(place(k,i))
     {
            x[k]=i;
            if(k==n)
            {
              count++;
              if(count%10 == 0)
              getch();
              printf("\n");
              for(j=1;j<=n;j++)
              {
                 printf("%d\t",x[j]);
              }
            }
            else
            {
              nqueen(k+1,n);
            }
     }
```

```
   }
}
int place(int k,int i)
{
   int j;
   for(j=1;j<=k-1;j++)
   {
      if((x[j]==i)||(abs(x[j]-i)==abs(j-k)))
      {
               return 0;
      }
   }
   return 1;
}
void main()
{
  clrscr();
  readv();
  getch();
}
```

Output:

```
Enter the no of Queens to be placed:4

 The Places in which the 4 Queens are to placed in the 4 x 4 ChessBoard is:
 2  4   1   3
 3  1   4   2
 The No of Solutions for the 4 Queens Problem are:2
```

Experiment No: 15

Aim: Write a program to solve Sum of subsets problem for a given set of distinct numbers.

Program:

```
#include<stdio.h>
#include<conio.h>
void SumOfSub(int,int,int);
int x[25],n,m=0,sum=0,w[25];;
void readf()
{
  int i;
  printf("\n Enter the no of values in the set:");
  scanf("%d",&n);
  printf("\n Enter the %d weights of the values in the set:",n);
  for(i=1;i<=n;i++)
  {
     scanf("%d",&w[i]);
     sum=sum+w[i];
     x[i]=0;
  }
  printf("\n Enter the required sum of the values in the subset:");
  scanf("%d",&m);
  printf("\n The Total sum of the weights is:%d",sum);
  SumOfSub(0,1,sum);
}
void SumOfSub(int s,int k,int r)
{
  int i,j;
  x[k]=1;
if(sum>=m)
{
  if(s+w[k]==m)
  {
    printf("\n");
    for(j=1;j<=n;j++)
    {
       printf("%d\t",x[j]);
    }
    printf("\n-->");
    for(j=1;j<=k;j++)
    {
      if(x[j] == 1)
              printf("%d\t",w[j]);
    }
  }
```

```
  else if(s+w[k]+w[k+1]<=m)
      SumOfSub(s+w[k],k+1,r-w[k]);
  if((s+r-w[k]>=m) && (s+w[k+1]<=m))
  {
    x[k]=0;
    SumOfSub(s,k+1,r-w[k]);
  }
 }
 else
 {
    printf("\n No Solutions Available because sum of all weights is %d less than required sum
%d",sum,m);
 }
}
void main()
{
  clrscr();
  readf();
  getch();
}
```

Output:

```
Enter the no of values in the set:6
Enter the 6 weights of the values in the set:5 10 12 13 15 18

 Enter the required sum of the values in the subset:30

 The Total sum of the weights is:73
1   1   0   0   1   0
-->5    10  15
1   0   1   1   1   0
-->5    12  13
0   0   1   0   0   1
-->12   18
```

Evaluation Procedure for Internal Laboratory Examinations:

1. Of the 25 marks for internal, 10 marks will be awarded for day-to-day work and 10 marks to be awarded for the Record work and 5 marks to be awarded by conducting an internal laboratory test.
2. Concerned Teachers have to do necessary corrections with explanations.
3. Concerned Lab teachers should enter marks in index page.
4. Internal exam will be conducted by two Staff members.

Dr.K. Subba Reddy

Professor & Head Dept. of CSE.

Evaluation Procedure for External Laboratory Examinations:

1. For Practical subjects there is a continuous evaluation during the semester for 25 Sessional marks and 50 end examination marks.
2. The end examination shall be conducted by the teacher concerned (Internal Examiner) and another External Examiner, recommended by Head of the Department with the approval of principal.

Evaluation procedure for external lab examination:

1. Procedure for the program	---------- 20M
2. Execution of the program	---------- 15M
3. Viva voce	---------- 15M

Total	50M

Dr.K. Subba Reddy
Professor & Head Dept. of CSE.