ADICHUNCHANAGIRI INSTITUTE OF TECHNOLOGY

DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING

Analysis & Design of Algorithms Lab

2023-24

Even Semester

Prepared By,

Deepashri K S
Asst. Professor
IS&E Department

Lab Program List

Sl. No.	Lab Programs	Page No.
1	Design and implement C/C++ Program to find Minimum Cost Spanning Tree of a given connected undirected graph using Kruskal's algorithm.	
2	Design and implement C/C++ Program to find Minimum Cost Spanning Tree of a given connected undirected graph using Prim's algorithm.	
3	a. Design and implement C/C++ Program to solve All-Pairs Shortest Paths problem using Floyd's algorithm. b. Design and implement C/C++ Program to find the transitive closure using Warshal's algorithm.	
4	Design and implement C/C++ Program to find shortest paths from a given vertex in a weighted connected graph to other vertices using Dijkstra's algorithm..	
5	Design and implement C/C++ Program to obtain the Topological ordering of vertices in a given digraph.	
6	Design and implement C/C++ Program to solve 0/1 Knapsack problem using Dynamic Programming method.	
7	Design and implement C/C++ Program to solve discrete Knapsack and continuous Knapsack problems using greedy approximation method..	
8	Design and implement C/C++ Program to find a subset of a given set S = {sl , s2,.....,sn} of n positive integers whose sum is equal to a given positive integer d.	
9	Design and implement C/C++ Program to sort a given set of n integer elements using Selection Sort method and compute its time complexity. Run the program for varied values of n> 5000 and record the time taken to sort. Plot a graph of the time taken versus n. The elements can be read from a file or can be generated using the random number generator	
10	Design and implement C/C++ Program to sort a given set of n integer elements using Quick Sort method and compute its time complexity. Run the program for varied values of n> 5000 and record the time taken to sort. Plot a graph of the time taken versus n. The elements can be read from a file or can be generated using the random number generator".	
11	Design and implement C/C++ Program to sort a given set of n integer elements using Merge Sort method and compute its time complexity. Run the program for varied values of n> 5000, and record the time taken to sort. Plot a graph of the time taken versus n. The elements can be read from a file or can be generated using the random number generator	
12	Design and implement C/C++ Program for N Queen's problem using Backtracking	

ADICHUNCHANAGIRI INSTITUTE OF TECHNOLOGY

DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING

Institution Vision and Mission

Vision

To develop Adichunchanagiri Institute of Technology as a center of excellence and to strive for continuous improvement of technical education and human resource advancement.

Mission

To achieve Excellence in Education, Entrepreneurship, and Innovation by producing Engineers with high Ethical Standards, Integrity, and Credibility.

ADICHUNCHANAGIRI INSTITUTE OF TECHNOLOGY

DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING

Department Vision and Mission

Vision

To be recognized as a center of excellence in information technology and allied areas with quality learning and research environment

Mission

- Provide intellectual & professional leadership in ethical and social areas pertaining to information in contemporary society.
- Advancing the state of knowledge of information studies through research and development.
- Providing a platform to discuss cutting edge technologies.

ADICHUNCHANAGIRI INSTITUTE OF TECHNOLOGY

DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING

Subject: Analysis & Design of Algorithms Lab
Subject code: BCSL404
Semester: IV

Course Outcomes

At the end of the course the student will be able to:

CO1	Develop programs to solve computational problems using suitable algorithm design strategy
CO2	Compare algorithm design strategies by developing equivalent programs and observing running times for analysis (Empirical).
CO3	Make use of suitable integrated development tools to develop programs.
CO4	Choose appropriate algorithm design techniques to develop solution to the computational and complex problems.
CO5	Demonstrate and present the development of program, its execution and running time(s) and record the results/inferences

1. **Design and implement C/C++ Program to find Minimum Cost Spanning Tree of a given connected undirected graph using Kruskal's algorithm.**

```
#include<stdio.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()
{
        printf ("\n\n\implementation 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("\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);
  }
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: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
```

2. **Design and implement C/C++ Program to find Minimum Cost Spanning Tree of a given undirected graph using Prim's algorithm.**

```
#include<stdio.h>
int a,b,u,v,n,i,j,ne=1;
int visited[10]={0},min,mincost=0,cost[10][10];
void main()
{
        printf("\n\tImplementation of Prim's algorithm\n");
        printf("\n Enter the number of nodes:");
        scanf("%d",&n);
        printf("\n Enter the 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;
                }
        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("\n Edge %d:(%d %d) cost:%d",ne++,a,b,min);
                        mincost+=min;
                        visited[b]=1;
                }
                cost[a][b]=cost[b][a]=999;
        }
printf("\n Minimun cost=%d",mincost);

}
```

Output:

```
    Implementation of Prim's algorithm

 Enter the number of nodes:4

 Enter the adjacency matrix:
0   20  10  50
20  0   60  999
10  60  0   40
50  999 40  0


 Edge 1:(1 3) cost:10
 Edge 2:(1 2) cost:20
 Edge 3:(3 4) cost:40
 Minimun cost=70
```

3a. Design and implement C/C++ Program to solve All-Pairs Shortest Paths problem using Floyd's algorithm.

```
#include<stdio.h>
int min(int a,int b);
void floyd(int w[10][10],int n);
void main ()
{
        int a[10][10],n,i,j;
        printf ("Enter the no of vertices\n");
        scanf ("%d",&n);
        printf ("Enter the cost adjacency matrix,0-self loop and 999-no edge\n");
        for (i=1;i<=n;i++)
                for (j=1;j<=n;j++)
                        scanf ("%d",&a[i][j]);
        floyd (a,n);
        printf ("Shortest path matrix\n");
        for (i=1;i<=n;i++)
        {
                for (j=1;j<=n;j++)
                        printf ("%d\t",a[i][j]);
                printf ("\n");
        }

}
void floyd (int w[10][10],int n)
{
        int i,j,k;
        for (k=1;k<=n;k++)
                for (i=1;i<=n;i++)
                        for (j=1;j<=n;j++)
                                w[i][j]=min(w[i][j],w[i][k]+w[k][j]);
}

int min(int a,int b )
{
        if(a<b)
                return a;

        return b;

}
```

Output:

```
Enter the no of vertices
4
Enter the cost adjacency matrix,0-self loop and 999-no edge
0   999 3   999
2   0   999 999
999 7   0   1
6   999 999 0
Shortest path matrix
0   10  3   4
2   0   5   6
7   7   0   1
6   16  9   0
```

3b. Design and implement C/C++ Program to find the transitive closure using Warshal's algorithm.

```
# include <stdio.h>
int n,a[10][10],p[10][10];
void path()
{
        int i,j,k;
        for (i=0;i<n;i++)
                for (j=0;j<n;j++)
                        p[i][j]=a[i][j];
        for (k=0;k<n;k++)
                for(i=0;i<n;i++)
                        for (j=0;j<n;j++)
                                if (p[i][k]==1&&p[k][j]==1) p[i][j]=1;
}
void main()
        {
                int i,j;
                printf ("Enter the number of nodes:");
                scanf ("%d",&n);
                printf ("\nEnter the adjacency matrix:\n");
                for(i=0;i<n;i++)
                        for(j=0;j<n;j++)
                                scanf("%d",&a[i][j]);
                path();
                printf("\nThe path matrix is showm below\n");
                for(i=0;i<n;i++)
                {
                        for(j=0;j<n;j++)
                                printf("%d ",p[i][j]);
                        printf("\n");
                }

}
```

Output:

```
Enter the number of nodes:4

Enter the adjacency matrix:
0   1   0   0
0   0   0   1
0   0   0   0
1   0   1   0

The path matrix is showm below
1 1 1 1
1 1 1 1
0 0 0 0
1 1 1 1
```

4. Design and implement C/C++ Program to find shortest paths from given vertex in a weighted connected graph to other vertices using Dijkstra's algorithm

```
#include<stdio.h>
#define infinity 999
void dij(int n,int v,int cost[10][10],int dist[10])
{
	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];
	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]);

}
```

Output:

```
 Enter the number of nodes:5

 Enter the cost matrix:
0 3 4 5 999
13 0 7 999 999
4 7 0 8 1
5 999 8 0 6
999 999 1 6 0

 Enter the source matrix:1

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

5. Design and implement C/C++ Program to obtain the Topological ordering of vertices in a given digraph.

```
#include<stdio.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;
                        }
                }
        }
        printf("The topological Sequence is:\n");
        for(i=0;i<n;i++)
                printf("%d ",t[i]);
}

void main()
{
        int i,j;
        printf("Enter number of jobs:");
        scanf("%d",&n);
        printf("\nEnter the adjacency matrix:\n");
        for(i=0;i<n;i++)
        {
```

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

Output:

```
Enter number of jobs:5

Enter the adjacency matrix:
0   0   1   0   0
0   0   1   0   0
0   0   0   1   1
0   0   0   0   1
0   0   0   0   0
The topological Sequence is:
1 0 2 3 4
```

6. Design and implement C/C++ Program to solve 0/1 Knapsack problem using Dynamic Programming method.

```
#include<stdio.h>
int w[10],p[10],v[10][10],n,i,j,cap,x[10]={0};
int max(int i,int j)
{
	return ((i>j)?i:j);
}
int knap(int i,int j)
{
	int value;
	if(v[i][j]<0)
	{
		if(j<w[i])
		value=knap(i-1,j);
	else
		value=max(knap(i-1,j),p[i]+knap(i-1,j-w[i]));
		v[i][j]=value;
	}
	return(v[i][j]);
}
void main()
{
	int profit,count=0;
	printf("\nEnter the number of elements\n");
	scanf("%d",&n);
	printf("Enter the profit and weights of the elements\n");
	for(i=1;i<=n;i++)
	{
		printf("For item no %d\n",i);
		scanf("%d%d",&p[i],&w[i]);
	}
	printf("\nEnter the capacity \n");
	scanf("%d",&cap);
	for(i=0;i<=n;i++)
		for(j=0;j<=cap;j++)
			if((i==0)||(j==0))
				v[i][j]=0;
			else
				v[i][j]=-1;
		profit=knap(n,cap);
	i=n;
	j=cap;
	while(j!=0&&i!=0)
	{
		if(v[i][j]!=v[i-1][j])
```

```
                {
                        x[i]=1;
                        j=j-w[i];
                        i--;
                }
                else
                        i--;
        }
        printf("Items included are\n");
        printf("Sl.no\tweight\tprofit\n");
        for(i=1;i<=n;i++)
        if(x[i])
                printf("%d\t%d\t%d\n",++count,w[i],p[i]);
printf("Total profit = %d\n",profit);

}
```

Output:

```
Enter the number of elements
4
Enter the profit and weights of the elements
For item no 1
12  2
For item no 2
10  1
For item no 3
20  3
For item no 4
15  2

Enter the capacity
5
Items included are
Sl.no   weight  profit
1       2       12
2       1       10
3       2       15
Total profit = 37
```

8. Design and implement C/C++ Program to find a subset of a given set S = {sl , s2,.....,sn} of n positive integers whose sum is equal to a given positive integer d. For example, if S={1,2,5,6,8} and d=9 there are two solutions {1,2,6} and {1,8}. A suitable message is to be displayed if the given problem instance doesn't have a solution.

```
#include<stdio.h>
int s[10] , x[10],d ;
void sumofsub ( int , int , int ) ;
void main ()
{
        int n , sum = 0 ;
        int i ;
        printf ( " \n Enter the size of the set : " ) ;
        scanf ( "%d" , &n ) ;
        printf ( " \n Enter the set in increasing order:\n" ) ;
        for ( i = 1 ; i <= n ; i++ )
        scanf ("%d", &s[i] ) ;
        printf ( " \n Enter the value of d : \n " ) ;
        scanf ( "%d" , &d ) ;
        for ( i = 1 ; i <= n ; i++ )
                sum = sum + s[i] ;
        if ( sum < d || s[1] > d )
                printf ( " \n No subset possible : " ) ;
        else
                sumofsub ( 0 , 1 , sum ) ;
}
void sumofsub ( int m , int k , int r )
{
        int i=1 ;
        x[k] = 1 ;
        if ( ( m + s[k] ) == d )
        {
                printf("Subset:");
                for ( i = 1 ; i <= k ; i++ )
                if ( x[i] == 1 )
                        printf ( "\t%d" , s[i] ) ;
                printf ( "\n" ) ;
        }
        else
                if ( m + s[k] + s[k+1] <= d )
                        sumofsub ( m + s[k] , k + 1 , r - s[k] ) ;
                if ( ( m + r - s[k] >= d ) && ( m + s[k+1] <=d ) )
                {
                        x[k] = 0;
                        sumofsub ( m , k + 1 , r - s[k] ) ;
                }
}
```

Output:

```
 Enter the size of the set : 5

 Enter the set in increasing order:
1 2 5 6 8

 Enter the value of d :
 9
Subset: 1   2   6
Subset: 1   8
```

9. Design and implement C/C++ Program to sort a given set of n integer elements using Selection Sort method and compute its time complexity. Run the program for varied values of n> 5000 and record the time taken to sort. Plot a graph of the time taken versus n. The elements can be read from a file or can be generated using the random number generator.

```
#include<stdio.h>
#include<time.h>
void selectionSort(int A[],int n)
{
        int i, j, min,temp;
        for (i = 0; i < =n-2; i++)
                {
                    min = i;
                    for(j = i+1; j < =n; j++)
                    {
                       if (A[j] < A[min])
                        min = j;
                    }
                    temp=A[i];
                    A[i]=A[min];
                    A[min]=temp;
                 }

}
void main()
{
                int i,n,a[200000];
                clock_t st,et,ts;
                printf("Enter the size of the array\n");
                scanf("%d",&n);
                printf("The array elements are:\n");
                for(i=0;i<n;i++)
                {
                        a[i]=rand()%100;
                        printf("%d\t",a[i]);
                }
                printf("\n");
                st=clock();
                selectionSort(a,n-1);
                et=clock();
                printf("\n");
                printf("Elements in sorted order\n");
                for(i=0;i<n;i++)
                        printf("%d\t",a[i]);
                printf("\n");
                printf("Time taken=%e seconds\n",(et-st)/CLOCKS_PER_SEC);

}/*End of main()*/
```

Output:

```
Enter the size of the array
10
The array elements are:
83  86  77  15  93  35  86  92  49  21

Elements in sorted order
15  21  35  49  77  83  86  86  93  92
```

10. Design and implement C/C++ Program to sort a given set of n integer elements using Quick Sort method and compute its time complexity. Run the program for varied values of n> 5000 and record the time taken to sort. Plot a graph of the time taken versus n. The elements can be read from a file or can be generated using the random number generator.

```
#include<stdio.h>
#include<time.h>
void quicksort(int*,int,int);
void partition(int*,int,int,int*);
void main()
{
	//long int k;
	int i,n,a[200000];
	clock_t st,et,ts;
	printf("Enter the size of the array\n");
	scanf("%d",&n);
	printf("The array elements are:\n");
	for(i=0;i<n;i++)
	{
		a[i]=rand()%100;
		printf("%d\t",a[i]);
	}
	printf("\n");
	st=clock();

	quicksort(a,0,n-1);

	et=clock();
	printf("\n");

	printf("Elements in sorted order\n");
	for(i=0;i<n;i++)
		printf("%d\t",a[i]);
	printf("\n");

	printf("Time taken=%e seconds\n",(et-st)/CLOCKS_PER_SEC);

}/*End of main()*/

void quicksort(int *a,int low,int high)
{
	int j;
```

```
	if(low>high)
		return;

	partition(a,low,high,&j);
	quicksort(a,low,j-1);
	quicksort(a,j+1,high);
}

void partition(int *a,int low,int high,int *pf)
{
	int l=low,h=high,temp,pivot;
	pivot=a[low];
	while(l<h)
	{
		while(a[l]<=pivot&&l<h)
			l++;
		while(a[h]>pivot)
			h--;
		if(l<h)
		{
			temp=a[l];
			a[l]=a[h];
			a[h]=temp;
		}
	}/*while*/
	a[low]=a[h];
	a[h]=pivot;
	*pf=h;
}
```

Output:

```
Enter the size of the array
10
The array elements are:
83  86  77  15  93  35  86  92  49  21

Elements in sorted order
15  21  35  49  77  83  86  86  93  92
```

11.Design and implement C/C++ Program to sort a given set of n integer elements using Merge Sort method and compute its time complexity. Run the program for varied values of n> 5000, and record the time taken to sort. Plot a graph of the time taken versus n. The elements can be read from a file or can be generated using the random number generator.

```
#include <stdio.h>
#include<time.h>
int a[100000],b[100000];
void Merge(int low, int mid, int high)
{
        int i, j, k;
        i=low; j=mid+1; k=low;
        while ( i<mid && j<=high )
        {
                if( a[i] <= a[j] )
                        b[k++] = a[i++] ;
                else
                        b[k++] = a[j++] ;
        }
        while (i<=mid)
                b[k++] = a[i++] ;
        while (j<=high)
                b[k++] = a[j++] ;
        for(i=low; i<=high; i++)
                a[i] = b[i];
}

void MergeSort(int low, int high)
{
        int mid;
        if(low < high)
                {
                        mid = (low+high)/2 ;
                        MergeSort(low, mid);
                        MergeSort(mid+1, high);
                        Merge(low, mid, high);
                }
}

void main()
{
        int n,i;
        clock_t st,et;
        printf("\n Enter size of the array:");
        scanf("%d", &n);
        printf("\nThe Array elements are:\n");
        for(i=0;i<n;i++)
        {
                a[i]=rand()%100;
                printf("%d\t",a[i]);
        }
        printf("\n");
```

```
        st=clock();
        MergeSort(0,n-1);
        et=clock();

        printf("\n Sorted Elements are : \n ");
        for(i=0; i<n; i++)
                printf("%d\t", a[i]);
        printf("\nThe time taken is %e",(et-st)/CLOCKS_PER_SEC);
}
```

Output:

```
Enter the size of the array
10
The array elements are:
83  86  77  15  93  35  86  92  49  21

Elements in sorted order
15  21  35  49  77  83  86  86  93  92
```

12. Design and implement C/C++ Program for N Queen's problem using Backtracking.

```
#include<stdio.h>
#include<math.h>
int a[30],count=0;
int place(int pos)
{
	int i;
	for(i=1;i<pos;i++)
	{
		if((a[i]==a[pos])||((abs(a[i]-a[pos])==abs(i-pos))))
			return 0;
	}
	return 1;
}
void print_sol(int n)
{
	int i,j;
	count++;
	printf("\n\nSolution #%d:\n",count);
	for(i=1;i<=n;i++)
	{
		for(j=1;j<=n;j++)
		{
			if(a[i]==j)
				printf("Q\t");
			else
				printf("*\t");
		}
		printf("\n");
	}
}
void queen(int n)
{
	int k=1;
	a[k]=0;
	while(k!=0)
	{
		a[k]=a[k]+1;
		while((a[k]<=n)&&!place(k))
			a[k]++;
		if(a[k]<=n)
		{
			if(k==n)
					print_sol(n);
				else
```

```
                        {
                                k++;
                                a[k]=0;
                        }
                }
                else
                        k--;
                }
}
void main()
{
        int i,n;
        printf("Enter the number of Queens\n");
        scanf("%d",&n);
        queen(n);
        printf("\nTotal solutions=%d",count);
}
```

Output:

```
Enter the number of Queens
4


Solution #1:
*       Q       *       *
*       *       *       Q
Q       *       *       *
*       *       Q       *


Solution #2:
*       *       Q       *
Q       *       *       *
*       *       *       Q
*       Q       *       *
```