Analyzing Design Algorithms (ADA) Lab Programs

1.Write a program to sort a list of N elements using Selection Sort Technique.

```
#include<stdio.h>
#include<conio.h>
void main ()
{
int a [100], i, j, n, swap, min;
printf ("Enter the size of the array\n");
scanf ("%d”, &n);
printf ("Enter %d elements \n”, n);
for (i=0; i<n; i++)
scanf ("%d", &a[i]);
for (i=0; i<n; i++)
{
        int min=i;
        for (j=i+1; j<n; j++)
        {
                if(a[i] < a[min])
                {
                        Min=j;
                }
        }
if (min! =i)
{
        swap= a[i];
        a[i]=a[min];
        a[min]=swap;
}
}
printf ("\n selection Sorted Array is:\n");
for (i=0; i < n; i++)
printf ("\n %d", a[i]);
getch ();
}
```

Output:

```
Enter the size of the array
Enter 5 elements
20
50
1
70
15
Selection Sorted Array is:
1
15
20
50
70
```

2.Write a program to implement divide and conquer strategy. Eg: Quick sort algorithm for sorting for sorting list of integers in ascending order.

```
#include<stdio.h>
void quicksort (int number [25], int first, int last)
{
  int i, j, pivot, temp;
  if(first<last)
        {
                pivot=first;
                i=first;
                j=last;
                while(i<j)
                {
                         while(number[i]<=number[pivot]&&i<last)
                         i++;
                        while(number[j]>number[pivot])
                         j--;
                        if(i<j)
                        {
                                temp=number[i];
                                number[i]=number[j];
                                number[j]=temp;
                         }
                }
        temp=number[pivot];
         number[pivot]=number[j];
         number[j]=temp;
        quicksort (number, first, j-1);
         quicksort (number, j+1, last);
        }
}
int main ()
{
  int i, count, number [25];
  printf ("How many elements are u going to enter: ");
  scanf ("%d”, &count);
  printf ("Enter %d elements: ", count);
  for (i=0; i<count; i++)
  scanf ("%d", &number[i]);
  quicksort (number,0, count-1);
  printf ("Order of Sorted elements: ");
  for (i=0; i<count; i++)
  printf (" %d”, number[i]);
  return 0;
}
```

Output:

```
How many elements are u going to enter: 5
Enter 5 elements:
 22 3 4 6 88
Order of Sorted elements:
  3 4 6 22 88
```

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

```
#include<stdio.h>
#include<conio.h>
void mergesort (int a [], int lb, int ub);
void merge (int a [], int lb, int mid, int ub);
void main ()
{
        int a [30], n, i;
        printf ("Enter number of elements: \n");
        scanf ("%d", &n);
        printf ("Enter array elements: \n");
        for (i=0; i<n; i++)
        scanf ("%d", &a[i]);
        mergesort (a, 0, n-1);
        printf ("\nSorted array is:");
        for (i=0; i<n; i++)
        printf ("%d ", a[i]);
        getch ();
}
 void mergesort (int a [], int lb, int ub);
{
        int mid;
        if (lb <ub)
        {
                mid=(lb+ub)/2;
                mergesort (a, lb, mid);
                mergesort (a, mid+1, ub);
                merge (a, lb, mid, ub);
        }
}
void merge (int a [], int lb, int mid, int ub);
{
int temp 50];
int i, j, k;
i=lb;
j=mid+1;
k=lb;
        while (i<=mid && j<=ub)
        {
                if(a[i]<=a[j])
                {
                        temp[k++] =a[i++];
                }
                else
                {
                        temp[k++] =a[j++];
                }
        }
        If(i>mid)
        {
                while(j<=ub)
                {
                        temp[k++] =a[j++];
                }
        }
        else
        {
                while(i<=mid)
                temp[k++] =a[i++];
        }
}
for (k=lb; k<=ub; k++)
{
        a[k]=temp{k];
}
}
```

Output:

```
Enter no of elements:5
Enter array elements:20 30 55 2 3
Sorted array is :2 3 20 30 55
```

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

```
#include<stdio.h>
int max, min;
int a [100];
void minmax (int i, int j)
{
        int max1, min1, mid;
        if(i==j)
        {
                max = min = a[i];
        }
        else
        {
                if (i == j-1)
                 {
                        if(a[i]  <a[j])
                        {
                                max = a[j];
                                min = a[i];
                        }
                        else
                        {
                                max = a[i];
                                min = a[j];
                        }
                 }
                 else
                 {
                        mid = (i+j)/2;
                        minmax (i, mid);
                        max1 = max;
                        min1 = min;
                        minmax (mid+1, j);
                        if (max <max1)
                        max = max1;
                        if (min > min1)
                         min = min1;
                 }
        }
}
void main ()
{
        int i, num;
        printf ("\n Enter the total number of numbers: ");
        scanf ("%d", &num);
        printf ("Enter the numbers: \n");
        for (i=1; i<=num; i++)
        scanf ("%d", &a[i]);
        max =max=a [0];
```

```
        minmax (1, num);
        printf ("Minimum element in an array:
        %d\n", min);
         printf ("Maximum element in an array:
        %d\n", max);
getch ();
}
```

Output:

```
Enter the total number of
numbers: 6Enter the numbers:
10 60 4 67 89 100
Minimum element in an
array: 4 Maximum
element in an array: 100
```

5. Write a program to perform Knapsack Problem using Greedy Solution

```
#include<stdio.h>
int main ()
{
        float weight [50], profit [50], ratio [50], Totalvalue, temp, capacity, amount;
        int n, i, j;
        printf ("Enter the number of items:");
        scanf ("%d", &n);
        for (i = 0; i < n; i++)
        {
                Printf ("Enter Weight and Profit for item[%d]: \n", i);
                Scanf ("%f %f", &weight[i], &profit[i]);
        }
        Printf ("Enter the capacity of knapsack:\n");
        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;
        }
        Printf ("Knapsack problems using Greedy Algorithm:\n");
        for (i = 0; i < n; i++)
        {
                if (weight[i] > capacity)
                        break;
                else
                {
                        Totalvalue = Totalvalue + profit[i];
                        capacity = capacity - weight[i];
                }
        }
if (i < n)
Totalvalue = Totalvalue + (ratio[i]*capacity);
Printf ("\n The maximum value is: %f\n”, Totalvalue);
return 0;
```

output: -

```
Enter the number of items :4
Enter Weight and Profit for item [0]: 2
12
Enter Weight and Profit for item [1]:1
10
Enter Weight and Profit for item [2]:3
20
Enter Weight and Profit for item [3]:2
15
Enter the capacity of knapsack:5
Knapsack problems using Greedy Algorithm:
The maximum value is :38.333332
```

6. Write a program to perform Travelling Salesman problem

```
#include<stdio.h>
#include<conio.h>
int cm [10][10], complete [10], n, cost=0;
void mincost (int city)
{
        int i, ncity;
        Completed[city]=1;
        Printf(“%d---- >”, city+1);
        ncity=least(city);
        if(ncity==999)
        {
                ncity=0;
                printf (“%d”, ncity+1);
                cost+=cm[city][ncity];
                return;
        }
        mincost(ncity);
}
int least (int c)
{
        int i, nc=999;
        int kmin, min=999;
        for (i=0; i<n; i++)
        {
                if((cm[c][i]! =0) &&(complete[i]==0))
                if(cm[c][i] +cm[i][c] <min)
                {
                        min=cm[i][0] +cm[c][i];
                        kmin=cm[c][i];
                        nc=i;
                }
        }
        if (min! =999)
        cost=cost+kmin;
        return nc;
}
void main ()
{
        int i, j;
        clrscr ();
        printf (“enter the number of vertices is:”);
        scanf (“%d”, &n);
        printf (“enter the cost matrix \n”);
        for (i=0; i<n; i++)
        {
                for (j=0; j<n; j++)
                scanf (“%d”, &cm[i][j]);
                complete[i]=0;
        }
printf (“\n minimum path is:\n”);
mincost (0);
printf (“\n minimum cost is %d \n”, cost);
getch ();
 }
```

Output:

```
enter the number of vertices is: 4
enter the cost matrix
0 4 1 3
4 0 2 1
1 2 0 5
3 1 5 0
minimum path is: 1→3→2→4→1
minimum cost is 7
```

7 Write program to implement Dynamic Programming algorithm for the 0/1 Knapsackproblem.

```
#include<stdio.h>
int max(int a, int b){return(a>b)?a:b;}
int knapsack (int n,int c,int w[],int p[])
 {
int i,j;
int k[20][20];
for(i =0;i<=n;i++)
{
for(j =0;j<=c;j++)
{
if(i==0||j==0)
k[i][j]=0;
else if(w[i]<=j)
k[i][j]=max(p[i]+k[i-1][j-w[i]],k[i-1][j]);
else
k[i][j]=k[i-1][j];
}
}
return k[n][c];
}
int main()
{
int n,c,p[20],w[20],i;
printf("\n Enter the number of objects:");
scanf("%d",&n);
printf("Enter the capacity of knapsack:\n");
scanf("%d",&c);
printf("Enter profit and Weight of objects:\n");
for(i=0;i<n;i++)
scanf("%d%d",&p[i],&w[i]);
printf("maximum Profit %d\n”,knapsack(n,c,w,p));
return 0;
}
```

```
Enter the number of
objects:4
Enter the capacity of
knapsack:
8
Enter profit and Weight of
objects:
1 2
2 3
5 4
6 5
maximum Profit 8
```

8.Write a program that implement Prim’s algorithm to generate minimum cost spanning tree.

```
#include <stdio.h>
#include<conio.h>
void main()
{

   int ne=1,n,min,vis[10]={0},cost[10][10],i,j,u,v,mincost=0;
   printf("enter the number of edges in the graph\n");
   scanf("%d",&n);
   printf("enter the cost adjacency matrix\n");
   for(i=1;i<=n;i++)
   for(j=1;j<=n;j++)
   {
      scanf("%d",&cost[i][j]);

   }
   vis[1]=1;
   while (ne<n)
   {
      min=999;
      for(i=1;i<=n;i++)
      for(j=1;j<=n;j++)
      if((vis[i]==1)&&(cost[i][j]<min))
      {
         min=cost[i][j];
         v=i;
         u=j;
      }
      if(vis[u]==0)
      {
         printf("%d-->%d=%d\n",v,u,min);
         ne++;
         mincost=mincost+min;
         vis[u]=1;
      }
      cost[u][v]=cost[v][u]=999;
   }
      printf("Minimum Cost=%d\n",mincost);
}
```

```
enter the number of edges in the graph
4
enter the cost adjacency matrix
999 20 10 15
20 999 14 16
10 14 999 18
5 16 18 999
1-->3=10
3-->2=14
1-->4=15
Minimum Cost=39
```

9.Write a program that implement Kruskal’s algorithm to generate minimum cost spanning tree.

```
#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:4
Enter the cost adjacency matrix:
0 1 2 3
4 5 6 7
4 8 9 7
4 5 6 7
The edges of Minimum Cost Spanning
Tree are
1 edge (1,2) =1
2 edge (1,3) =2
3 edge (1,4) =3
Minimum cost = 6
```

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

```
#include <stdio.h>
#include <stdlib.h>
void main() {
    int n, i, j, u, v;
    int adj_matrix[50][50];
     int e;
     clrscr();
    printf("Enter the number of vertices: ");
    scanf("%d", &n);
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            adj_matrix[i][j] = 0;
        }
    }
    printf("Enter the number of edges: ");
    scanf("%d", &e);
    printf("Enter the edges (u, v):\n");
    for (i = 0; i < e; i++) {
        scanf("%d %d", &u, &v);
        adj_matrix[u - 1][v - 1] = 1;
        adj_matrix[v - 1][u - 1] = 1;
    }
    printf("The adjacency matrix is:\n");
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            printf("%d ", adj_matrix[i][j]);
        }
        printf("\n");
        getch();
    }
}
```

Output:

```
Enter the number of vertices: 3
Enter the number of edges: 3
Enter the edges (u, v):
1 3
1 2
2 3
The adjacency matrix is:
0 1 1
1 0 1
1 1 0
```

11. Implement function to print the In-degree, Out-degree and to display that adjacency matrix.

```
#include <stdio.h>
#define MAX 100
// A function to print the in-degree, out-degree and the adjacency matrix of a graph
void print_graph(int adj[MAX][MAX], int n) {
    int i, j, in, out;
    printf("Vertex\tIn\tOut\n");
    for (i = 0; i < n; i++) {
        in = 0; // Initialize the in-degree of vertex i
        out = 0; // Initialize the out-degree of vertex i
        for (j = 0; j < n; j++) {
            if (adj[i][j] == 1) // If there is an edge from i to j
                out++; // Increment the out-degree of i
            if (adj[j][i] == 1) // If there is an edge from j to i
                in++; // Increment the in-degree of i
        }
        printf("%d\t%d\t%d\n", i, in, out); // Print the vertex, in-degree and out-degree
    }
    printf("\nThe adjacency matrix is:\n");
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            printf("%d ", adj[i][j]); // Print the adjacency matrix element
        }
        printf("\n");
    }
}

int main() {
    int adj[MAX][MAX]; // Declare an adjacency matrix
    int n, i, j; // Declare the number of vertices and loop variables
    printf("Enter the number of vertices: ");
    scanf("%d", &n); // Read the number of vertices
    printf("Enter the adjacency matrix:\n");
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            scanf("%d", &adj[i][j]); // Read the adjacency matrix element
        }
    }
    print_graph(adj, n); // Call the function to print the graph
    return 0;
}
```

Output:

```
Enter the number of vertices: 3
Enter the adjacency matrix:
0 1 2
1 2 3
2 3 4
Vertex In       Out
0       1       1
1       1       1
2       0       0

The adjacency matrix is:
0 1 2
1 2 3
2 3 4
```

12.Write A Program to Implement Job Sequencing Algorithm Using Greedy Approach.

```
#include<stdio.h>
#include<conio.h>
int jobsequence(int d[6],int j[6],int n)
{
int q,i,r,k;
d[0]=0;
j[0]=0;
j[1]=1; k=1;
for(i=2;i<=n;i++)
{
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 main()
{
 int d[6],j[6],p[6],k,i;
clrscr();
printf("Enter the deadlines :");
for(i=1;i<=5;i++)
 scanf("%d",&d[i]);
 printf("Enter the profits :");
 for(i=1;i<=5;i++)
 scanf("%d",&p[i]);
 for(i=1;i<=5;i++)
j[i]=i;
 k=jobsequence(d,j,5);
printf("\nThe solution job sequence is ");
 for(i=1;i<=k;i++)
printf("\n%d",j[i]);
getch();
}
```

Output:

```
Enter the deadlines :1 2 5 6 7
Enter the profits :2 3 5 4 7
The solution job sequence is
1
2
3
4
5
```

13. C Program to implement Optimal Binary Search using Dynamic Programming

```
#include <stdio.h>
#include <limits.h>
#include<conio.h>
int cost[10][10];
int sum(int freq[], int i, int j);
int optimalSearchTree(int keys[],int freq[], int n)
{
int i,L,r,j; for(i=0;i<n;i++)
cost[i][i] = freq[i]; for (L=2;L<=n;L++)
{
for (i=0;i<=n-L+1;i++)
{
int j=i+L-1;
int off_set_sum=sum(freq,i,j);
 cost[i][j]=INT_MAX;
for (r=i;r<=j;r++)
{
int c=((r>i)?cost[i][r-1]:0)+((r < j)? cost[r+1][j]:0)+off_set_sum;
if (c < cost[i][j])
cost[i][j] = c;
}
}
}
return cost[0][n-1];
}
int sum(int freq[],int i,int j)
{
int k,s=0;
for(k=i;k<=j;k++)
s+=freq[k];
return s;
}
int main()
{
int keys[] = {10, 12, 20};
int freq[] = {34, 8, 50};
int n=sizeof(keys)/sizeof(keys[0]);
clrscr();
printf("Cost of Optimal BST is %d ",optimalSearchTree(keys,freq,n));
getch();
return 0;
}
```

Output:

```
Cost of Optimal BST is 142
```

14. Write program to implement backtracking algorithm for the solving problems like N queens.

15.Write program to implement the DFS and BFS algorithm for a graph.