ADA Lab Programs-2024

1. Selection Sort:

```
#include<stdio.h>
#include<time.h>
#include<stdlib.h>
#include<conio.h>
void selection(int arr[], int n)
{
   int i, j, small;

   for (i = 0; i < n-1; i++)    // One by one move boundary of unsorted subarray
   {
      small = i; //minimum element in unsorted array

      for (j = i+1; j < n; j++)
      if (arr[j] < arr[small])
         small = j;
// Swap the minimum element with the first element
   int temp = arr[small];
   arr[small] = arr[i];
   arr[i] = temp;
   }
}

void main()
 {
  int a[200],i,n;
 clock_t s,e;
 printf("enter the size of unsorted list\n");
 scanf("%d",&n);

 for(i=0;i<n;i++)
  {
   a[i]=rand();
  }
 printf("an usorted list is\n");
 for(i=0;i<n;i++)
  {
   printf("%d ",a[i]);
  }
s=clock();
       selection(a, n);
e=clock();
printf("the sorted list is\n");
 for(i=0;i<n;i++)
  {
   printf("%d ",a[i]);
  }
printf("\n total time taken=%f\n",(double)(e-s)/CLOCKS_PER_SEC);
}
```

2. Merge Sort :

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

int a[100000],b[100000];

void merge(int l,int m,int h)
{
        int i,j,k;

        i=l; j=m+1; k=l;

        while(i<=m && j<=h)
                {

                        if( a[i] <a[j])
                                { b[k]=a[i]; i++; }
                        else
                                { b[k]=a[j]; j++; }
                        k++;
                }

        while ( i<=m)
                {
                        b[k]=a[i]; k++; i++;
                }

        while(j<=h)
                {
                        b[k]=a[j]; k++; j++;
                }

for(i=l;i<=h;i++)
                a[i]=b[i];
 }

void   mergesort(int l,int h)
{
 if(l<h)
  {
        int m=(l+h)/2;
        mergesort(l,m);
        mergesort(m+1,h);
        merge(l,m,h);
  }
}

 main()
{
        int i,j,n,k;
        clock_t s,e;
        double TT;
        printf("\n Enter number of elements \n");
        scanf("%d",&n);
```

```
        for(i=0;i<n;i++)
        {
          a[i]=rand();
           //a[i]=i;
           //a[i]=n-i;
        }
        printf("\n UNSORTED ARRAY \n");
        for(i=0;i<n;i++)
        printf("%d\t",a[i]);

        k=n/2;
        s=clock();
        mergesort(1,k);
        mergesort(k+1,n);
        merge(1,k,n);
         e=clock();
        printf("\n SORTED ARRAY \n");
        for(i=0;i<n;i++)
        printf("%d\t",a[i]);
        printf("\n\n");
        TT = ((double)(e - s)) / CLOCKS_PER_SEC;
    printf("Time taken to sort %d elements: %lf seconds\n", n, TT);
}
```

3. Quick Sort:

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
#include<time.h>
int a[100000],n;
int partition(int lower,int upper)
{
int l, i, j, t, key ;
        i = lower + 1 ;
        j = upper;
        key = a[lower] ;
        while ( j >= i )
        {
                while ( a[i] <=key&&i<=upper)
                        i++ ;
                while ( a[j] > key && j>lower)
                        j-- ;
                if ( j >= i )
                {
                        t = a[i] ;
                        a[i] = a[j] ;
                        a[j] = t ;
                }
        }
        t = a[lower] ;
        a[lower] = a[j] ;
        a[j] = t ;
        return j ;
}
```

```
void quick(int low,int high)
{
 int j;
 if(low<high)
  {
   j=partition(low,high);
   quick(low,j-1);
   quick(j+1,high);
  }
}

void main()
{
 int i;
 clock_t s,e;
  printf("enter the size of unsorted list\n");
 scanf("%d",&n);

 for(i=0;i<n;i++)
 {
  a[i]=rand();  //avg case
  //a[i]=i;     //Best Case
  //a[i]=n-i;    //Worst case
 }

 printf("an unsorted list is\n");
 for(i=0;i<n;i++)
 {
  printf("%d ",a[i]);
 }
 printf("\n");
 s=clock();
 for(i=0;i<n;i++)
 quick(0,n-1);
 e=clock();
 printf("a sorted list is:\n");
 for(i=0;i<n;i++)
 {
  printf("%ld ",a[i]);
 }
 printf("\n total time taken=%f\n",((double)(e-s))/CLOCKS_PER_SEC);
}
```

4. Prim's Algorithm:

```
#include<stdio.h>
#include<conio.h>
#include<process.h>
void prims();
int c[10][10],n;
void main()
{
int i,j;
printf("\n enter the no. of vertices:\t");
scanf("%d",&n);
printf("\n enter the cost matrix:\n");
for(i=1;i<=n;i++)
{
for(j=1;j<=n;j++)
```

```
{
scanf("%d",&c[i][j]);
}
}
prims();
}

void prims()
{
int i,j,u,v,min;
int ne=0,mincost=0;
int elec[10];
for(i=1;i<=n;i++)
{
        elec[i]=0;
}
elec[1]=1;
while(ne!=n-1)
{
        min=9999;
        for(i=1;i<=n;i++)
        {
                for(j=1;j<=n;j++)
                {
                        if(elec[i]==1)
                        {
                                if(c[i][j]<min)
                                  {
                                    min=c[i][j];
                                     u=i;
                                     v=j;
                                  }
                        }
                 }
         }
   if(elec[v]!=1)
    {
      printf("\n%d----->%d=%d\n",u,v,min);
       elec[v]=1;
      ne=ne+1;
      mincost=mincost+min;
    }
c[u][v]=c[v][u]=9999;
}
printf("\n mincost=%d",mincost);
}
```

5. **Kruskal Algorithm:**

```
#include<stdio.h>
#include<conio.h>
void kruskals();
int c[10][10],n;
void main()
{
int i,j;

printf("\n enter the no. of vertices:\t");
scanf("%d",&n);
printf("\n enter the cost matrix:\n");
```

```
for(i=1;i<=n;i++)
{
for(j=1;j<=n;j++)
{
scanf("%d",&c[i][j]);
}
}
kruskals();

}
void kruskals()
{
int i,j,u,v,a,b,min;
int count=0,mincost=0;
int parent[10];
for(i=1;i<=n;i++)
{
parent[i]=0;
}
while(count!=n-1)
{
        min=9999;
          for(i=1;i<=n;i++)
          {
          for(j=1;j<=n;j++)
           {
            if(c[i][j]<min)
              {
                    min=c[i][j];
                    u=a=i;
                    v=b=j;
              }
            }
         }
while(parent[u]!=0)
{
u=parent[u];
}
while(parent[v]!=0)
{
v=parent[v];
}
          if(u!=v)
          {
          printf("\n%d----->%d=%d\n",a,b,min);
          parent[v]=u;
          mincost=mincost+min;
          }
c[a][b]=c[b][a]=9999;
count++;
}
printf("\n mincost=%d",mincost);
}
```

6. **Topological Sorting Algorithm:**

```
#include<stdio.h>
#include<conio.h>
int indegree[20],t[20],a[20][20],n;
void t_sort()
{
int top=-1,k=0,i,j,sum=0,s[20],u,v;
for(j=1;j<=n;j++)
 {
        sum=0;
        for(i=1;i<=n;i++)
        sum=sum+a[i][j];
        indegree[j]=sum;
 }
for(i=1;i<=n;i++)
{
 if(indegree[i]==0)
 {
  top=top+1;
  s[top]=i;
 }
}
while(top!=-1)
{
 u=s[top];
 top=top-1;
 t[k++]=u;
         for(v=1;v<=n;v++)
         {
                 if(a[u][v]==1)
                 {
                 indegree[v]--;
                 if(indegree[v]==0)
                 {
                 top=top+1;
                 s[top]=v;
                 }
                 }
         }
 }
printf("\n Topological sequence is: \n");
for(i=0;i<k;i++)
printf("%d\t",t[i]);
}

void main()
{
int i,j;
printf("enter the no. of vertices\n");
scanf("%d",&n);
printf("enter the adjacency matrix\n");
        for(i=1;i<=n;i++)
        {
        for(j=1;j<=n;j++)
        scanf("%d",&a[i][j]);
        }
t_sort();
}
```

7. **Dijkstra Algorithm:**

```
#include<stdio.h>
#include<conio.h>
void dijkstras();
int c[10][10],n,src;
void main()
{
int i,j;
printf("\nenter the no of vertices:\t");
scanf("%d",&n);
printf("\nenter the cost matrix:\n");
for(i=1;i<=n;i++)
{
for(j=1;j<=n;j++)
{
scanf("%d",&c[i][j]);
}
}
printf("\nenter the source node:\t");
scanf("%d",&src);
dijkstras();
}

void dijkstras()
{
int vis[10],dist[10],u,j,count,min;
for(j=1;j<=n;j++)
{
dist[j]=c[src][j];
}
for(j=1;j<=n;j++)
{
vis[j]=0;
}
dist[src]=0;
vis[src]=1;
count=1;
while(count!=n)
{
        min=9999;
        for(j=1;j<=n;j++)
        {
                if(dist[j]<min&&vis[j]!=1)
                {
                min=dist[j];
                u=j;
                }
        }
vis[u]=1;
count++;
        for(j=1;j<=n;j++)
        {
        if(min+c[u][j]<dist[j]&&vis[j]!=1)
        {
        dist[j]=min+c[u][j];
        }
        }
}
```

```
printf("\n the shortest distance is:\n");
        for(j=1;j<=n;j++)
        {
        printf("\n%d----->%d=%d",src,j,dist[j]);
        }
}
```

8. Warshall's Algorithm:

```
#include<conio.h>
#include<stdio.h>
int a[10][10],n;
void warshalls();
void main()
{
int i,j;

printf("\n Enter the no. of vertices:\t");
scanf("%d",&n);
printf("\n Enter the adjacency matrix:\n");
for(i=1;i<=n;i++)
{
for(j=1;j<=n;j++)
{
scanf("%d",&a[i][j]);
}
}
warshalls();
}

void warshalls()
{
int i,j,k;
        for(k=1;k<=n;k++)
        {
        for(i=1;i<=n;i++)
        {
        for(j=1;j<=n;j++)
        {
                if(a[i][j]!=1)
                {
                if(a[i][k]==1&&a[k][j]==1)
                {
                a[i][j]=1;
                }
        }
        }
        }
}
printf("\n Path matrix is:\n");
for(i=1;i<=n;i++)
{
for(j=1;j<=n;j++)
{
printf("%d\t",a[i][j]);
}
printf("\n\n");
}
}
```

9. Floyd's Algorithm:

```
#include<stdio.h>
#include<sys/time.h>
#include<stdlib.h>
#include<unistd.h>

int n,c[10][10],d[10][10];
int min(int,int);
void read_data();
void write_data();
void floyds();

void write_data()
{
int i,j;
printf("the least distance matrix is\n ");
for(i=0;i<n;i++)
{
for(j=0;j<n;j++)
{
printf("%d\t",d[i][j]);
}
printf("\n");
}
}
void floyds()
{
int i,j,k;
for(i=0;i<n;i++)
for(j=0;j<n;j++)
d[i][j]=c[i][j];
        for(k=0;k<n;k++)
        {
        for(i=0;i<n;i++)
        {
        for(j=0;j<n;j++)
        {
                d[i][j]=min(d[i][j],d[i][k]+d[k][j]);
        }
        }
        }
}

int min(int a,int b)
{
if(a<b)
return a;
return b;
}
void read_data()
{
int i,j;
printf("enter the number of vertices\n");
scanf("%d",&n);
printf("enter the cost matrix\n");
for(i=0;i<n;i++)
{
for(j=0;j<n;j++)
```

```
{
scanf("%d",&c[i][j]);
}
}
}
int main()
{
read_data();
floyds();
write_data();
}
```

10. Greedy Knapsack:

```
#include<stdio.h>
#include<conio.h>
void Gknapsack();
int max(int,int);
int i,j,n,M1,p[10],w[10], sol[10];
float ratio[10];
void main()
{

printf("\n enter the no. of items:\t");
scanf("%d",&n);
printf("\n enter the weight of the each item:\n");
for(i=1;i<=n;i++)
{
scanf("%d",&w[i]);
}
printf("\nenter the profit of each item:\n");
for(i=1;i<=n;i++)
{
scanf("%d",&p[i]);
}
printf("\nenter the knapsack's capacity:\t");
scanf("%d",&M1);
Gknapsack();
}

void Gknapsack()
{
int sum=0,q;

for(i=1;i<=n;i++)
{
ratio[i]=(float)p[i]/w[i];
//profit to weight ratio
}

for (q=1;q<=n;q++)
{
        float max=0.0;
        int k = 0; //to store max ratio index
        for(i=1;i<=n;i++)
        {
```

```
                    if((ratio[i] > max) && (sol[i] == 0))
                    {
                      max=ratio[i];
                     k = i;
                    }
            }

     if(M1>= w[k])
    {
      sol[k] = 1; //selected
     M1 = M1 - w[k];
      sum = sum + p[k];
    }
else
   {
    sol[k] = -1; //cannot select
   }
}

for(i=1;i<=n;i++)
{
if(sol[i] == -1)
sol[i] = 0;
}
printf("\nSolution with Greedy Method:%d ",sum);
printf("The solution vector is:");
for(i=1;i<=n;i++)
printf(" %d\t",sol[i]);
}
```

11. Dynamic Knapsack:

```
#include<stdio.h>
#include<conio.h>
void Dknapsack();
int max(int,int);
int i,j,n,m,p[10],w[10],v[10][10];
void main()
{
printf("\n enter the no. of items:\t");
scanf("%d",&n);
printf("\n enter the weight of the each item:\n");
for(i=1;i<=n;i++)
{
scanf("%d",&w[i]);
}
printf("\n enter the profit of each item:\n");
for(i=1;i<=n;i++)
{
scanf("%d",&p[i]);
}
printf("\n enter the knapsack's capacity:\t");
scanf("%d",&m);
Dknapsack();
}

void Dknapsack()
{
int x[10];
```

```
            for(i=0;i<=n;i++)
            {
             for(j=0;j<=m;j++)
             {
                        if(i==0||j==0)
                       {
                         v[i][j]=0;
                       }
                        else if(j-w[i]<0)
                       {
                       v[i][j]=v[i-1][j];
                       }
                      else
                       {
                          v[i][j]=max(v[i-1][j],v[i-1][j-w[i]]+p[i]);
                       }
             }
            }
printf("\n the output is:\n");
for(i=0;i<=n;i++)
{
 for(j=0;j<=m;j++)
 {
   printf("%d\t",v[i][j]);
 }
 printf("\n\n");
}
printf("\n the optimal solution is %d",v[n][m]);
printf("\n the solution vector is:\n");
for(i=n;i>=1;i--)
{
   if(v[i][m]!=v[i-1][m])
   {
    x[i]=1;
    m=m-w[i];
   }
   else
   {
   x[i]=0;
   }
}
for(i=1;i<=n;i++)
 {
 printf("%d\t",x[i]);
 }
}

int max(int x,int y)
{
            if(x>y)
            {
            return x;
            }
            else
            {
            return y;
            }
}
```

12. Subset Sum Algorithm

```
#include<stdio.h>
#include<conio.h>
void subset(int,int,int);
int count=0,d,s[10],x[10];
void main()
{
int sum=0, i,n;
printf("\n enter no. of elements:\t");
scanf("%d",&n);
printf("\n enter the elements in ascending order:\n");
for(i=0;i<=n-1;i++)
{
scanf("%d",&s[i]);
}
printf("\n enter the required sum:\t");
scanf("%d",&d);
for(i=0;i<=n-1;i++)
{
sum=sum+s[i];
}
        if(sum<d||s[0]>d)
        {
        printf("no solution exists\n");
        }
        else
        {
        subset(0,0,sum);
        }

}

void subset(int m,int k,int sum)
{
int i;
x[k]=1;
if(m+s[k]==d)
{
        printf("\n subset solution %d is\n",++count);
        for(i=0;i<=k;i++)
        {
        if(x[i]==1)
        {
        printf("%d\t",s[i]);
        }
        }
}
else if(m+s[k]+s[k+1]<=d)
        {
        subset(m+s[k],k+1,sum-s[k]);
        }
if((m+sum-s[k]>=d)&&(m+s[k+1]<=d))
{
x[k]=0;
subset(m,k+1,sum-s[k]);
}
if(count==0)
printf("no solution exists\n");
}
```

13. NQueens Algorithm:

```
#include<stdio.h>
#include<conio.h>
void nqueens(int);
int place(int[],int);
void prin(int n,int x[])
{
char c[10][10];
int i,j;
        for(i=1;i<=n;i++)
        {
        for(j=1;j<=n;j++)
        {
        c[i][j]='X';
        }
        }
for(i=1;i<=n;i++)
{
c[i][x[i]]='Q';
}
        for(i=1;i<=n;i++)
        {
        for(j=1;j<=n;j++)
        {
        printf("%c",c[i][j]);
        }
        printf("\n");
        }
}

void main()
{
int n;
printf("\nenter the no of queens:\t");
scanf("%d",&n);
if(n==2 || n==3)
printf("no solution for %d queens\n",n);
else
nqueens(n);
}
void nqueens(int n)
{
int k,x[10],count=0;
k=1;
x[k]=0;
while(k!=0)
{
x[k]++;
        while(place(x,k)==1&&x[k]<=n)
        {
        x[k]++;
        }
if(x[k]<=n)
{
        if(k==n)
        {
        printf("\nsolution %d is\n",++count);
        for(k=1;k<=n;k++)
        printf("%d----->%d\n",k,x[k]);
```

```
            printf("\n solution in the form of chess board\n");
            prin(n,x);
            }
            else
            {
            k++;
            x[k]=0;
            }
}
else
{
     k--;
}
}
}
int place(int x[],int k)
{
int i;
            for(i=1;i<=k-1;i++)
            {
            if(i-x[i]==k-x[k]||i+x[i]==k+x[k]||x[i]==x[k])
            {
            return 1;
            }
           }
return 0;
}
```