**INSERTION SORT**

#include<stdio.h>

void insertion\_sort(int a[],int n)

{

int i,value,hole;

for(i=1;i<n;i++)

{

value=a[i];

hole=i;

while(i>0 && a[hole-1]>value)

{

a[hole]=a[hole-1];

hole=hole-1;

}

a[hole]=value;

}

printf("\nsorted array is :\t");

for(i=0;i<n;i++)

{

printf("%d\n ",a[i]);

}

}

void main()

{

int a[10],i,n,j=100000,k=99999, b[10];

printf("Enter number of elements\n");

scanf("%d",&n);

printf("Enter elements to sort\n");

for(i=0;i<n;i++)

{

scanf("%d",&a[i]);

}

insertion\_sort(a,n);

}

**SELECTION SORT**

#include<stdio.h>

void selection\_sort(int a[],int n)

{

int i,j,imin,temp;

for(i=0;i<n;i++)

{

imin=i;

for(j=i+1;j<n;j++)

{

if(a[j]<a[imin])

{

imin=j;

}

}

temp=a[i];

a[i]=a[imin];

a[imin]=temp;

}

printf("\nsorted array is :\t");

for(i=0;i<n;i++)

{

printf("%d\n ",a[i]);

}

}

void main()

{

int a[100],i,n;

printf("Enter number of elements\n");

scanf("%d",&n);

printf("Enter elements to sort\n");

for(i=0;i<n;i++)

{

scanf("%d",&a[i]);

}

selection\_sort(a,n);

}

**BINARY SEARCH**

#include<stdio.h>

int binary(int a[],int low,int high, int val)

{

int mid;

if(high==low)

{

if(val==a[low])

return low;

else

return (-1);

}

else

{

mid=(high+low)/2;

if(val==a[mid])

return mid;

else if(val<a[mid])

return binary(a,low,mid-1,val);

else if(val>a[mid])

return binary(a,mid+1,high,val);

}

}

void insertion\_sort(int a[],int n)

{

int i,j,temp;

for(i=1;i<n;i++)

{

temp=a[i];

for(j=i-1;j>=0 && a[j]>temp;j--)

{

a[j+1]=a[j];

}

a[j+1]=temp;

}

}

void main()

{

int a[100],mid,low,high,n,i,val,index;

printf("\nEnter number of elements\n");

scanf("%d",&n);

printf("\nEnter elements of array\n");

for(i=0;i<n;i++)

{

scanf("%d",&a[i]);

}

insertion\_sort(a,n);

printf("\nEnter value to search\n");

scanf("%d",&val);

low=0;

high=n-1;

index=binary(a,low,high,val);

if(index==(-1))

printf("\nValue not found!\n");

else

printf("\nValue found at %d\n",(index+1));

}

**FINDING MINIMUM AND MAXIMUM**

#include<stdio.h>

int max,min;

void maxmin(int a[],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;

maxmin(a,i,mid);

max1=max;

min1=min;

maxmin(a,mid+1,j);

if(max<max1)

max=max1;

if(min>min1)

min=min1;

}

}

void main()

{

int a[100],n,i;

printf("Enter number of elements\n");

scanf("%d",&n);

printf("Enter elements of array\n");

for(i=0;i<n;i++)

scanf("%d",&a[i]);

maxmin(a,0,n-1);

printf("\nMax element is %d and Min element is %d",max,min);

printf("\nSelect option\n1.Start new search for max & min\n2.Exit\n");

scanf("%d",&op);

}

**MERGE SORT**

#include<stdio.h>

void merge(int a[],int low,int mid,int high)

{

int i,j,k,temp[100];

k=low;

i=low;

j=mid+1;

while(i<=mid && j<=high)

{

if(a[i]<=a[j])

{

temp[k]=a[i];

i++;

k++;

}

else

{

temp[k]=a[j];

j++;

k++;

}

}

while(i<=mid)

{

temp[k]=a[i];

i++;

k++;

}

while(j<=high)

{

temp[k]=a[j];

j++;

k++;

}

for(k=low;k<=high;k++)

{

a[k]=temp[k];

}

}

void MergeSort(int a[],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[100],i,n;

printf("Enter number of elements\n");

scanf("%d",&n);

printf("Enter elements to sort\n");

for(i=0;i<n;i++)

{

scanf("%d",&a[i]);

}

MergeSort(a,0,n-1);

printf("\nsorted array is :\t");

for(i=0;i<n;i++)

{

printf("%d ",a[i]);

}

}

**QUICK SORT**

#include<stdio.h>

void quicksort(int a[],int beg,int end)

{

int pivot;

if(beg<end)

{

pivot=partition(a,beg,end);

quicksort(a,beg,pivot);

quicksort(a,pivot+1,end);

}

}

void partition(int a[],int beg,int end)

{

int pivot,temp;

pivot=beg;

while(beg<end)

{

while(a[pivot]<a[end] && end>pivot)

{

end--;

}

if(end>pivot)

{

temp=a[pivot];

a[pivot]=a[end];

a[end]=temp;

pivot=end;

}

while(a[pivot]>a[beg] && beg<pivot)

{

beg++;

}

if(beg<pivot)

{

temp=a[pivot];

a[pivot]=a[beg];

a[beg]=temp;

pivot=beg;

}

}

return pivot;

}

void main()

{

int a[20],i,pivot,n;

printf("Enter number of elements\n");

scanf("%d",&n);

printf("Enter elements\n");

for(i=0;i<n;i++)

scanf("%d",&a[i]);

quicksort(a,0,n);

printf("Sorted array is:\n");

for(i=0;i<n;i++)

printf("%d ",a[i]);

}

**TENNIS TOURNAMENT**

#include<stdio.h>

void tour(int a[][50], int r, int n)

{

int i,j,k;

for(i=0;i<n;i=i+r)

{

for(j=i;j<i+r/2;j++)

{

for(k=0;k<r/2;k++)

{

a[j+r/2][k+r/2]=a[j][k];

}

}

for(j=i+r/2;j<i+r;j++)

{

for(k=0;k<r/2;k++)

{

a[j-r/2][k+r/2]=a[j][k];

}

}

}

}

void main()

{

int n,a[50][50],i,j,r=2;

printf("Enter number of players\n");

scanf("%d",&n);

for(i=0;i<n;i++)

{

a[i][0]=i+1;

}

while(r<=n)

{

tour(a,r,n);

r=r\*2;

}

printf("Tournament table :\n");

printf("\n");

for(i=0;i<n;i++)

{

if(i==0)

{

printf("Players\t");

}

else

{

printf("Day %d\t",i);

}

}

printf("\n");

for(i=0;i<n;i++)

{

for(j=0;j<n;j++)

{

printf("%d\t",a[i][j]);

}

printf("\n");

}

}

**KNAPSACK PROBLEM**

#include<stdio.h>

void print(float a[][4], int n)

{

int i,j;

for(i=0;i<=3;i++)

{

for(j=0;j<n;j++)

{

printf("%.2f\t",a[j][i]);

}

printf("\n");

}

}

void swap(float a[][4], int n)

{

int i;

float temp;

for(i=0;i<3;i++)

{

temp=a[n][i];

a[n][i]=a[n-1][i];

a[n-1][i]=temp;

}

}

void insertion\_sort(float a[][4],int n)

{

int i,j;

for(i=1;i<n;i++)

{

//temp=a[i];

j=i;

while(j>0 && a[j-1][2]<a[j][2])

{

swap(a,j);

j=j-1;

}

//a[j]=temp;

}

}

void main()

{

int i,j=0,n;

float a[50][4],w,profit;

printf("Enter number of terms :");

scanf("%d",&n);

printf("Enter the weight of sack\n");

scanf("%f",&w);

printf("Enter profit of elements :\n");

for(i=0;i<n;i++)

{

printf("%d : ",i+1);

scanf("%f",&a[i][0]);

}

printf("Enter weight of elements :\n");

for(i=0;i<n;i++)

{

printf("%d : ",i+1);

scanf("%f",&a[i][1]);

}

for(i=0;i<n;i++)

{

a[i][2]=a[i][0]/a[i][1];

}

insertion\_sort(a,n);

while(w>0&&j<n)

{

if(a[j][1]<=w)

{

w=w-a[j][1];

a[j][3]=1;

}

else if(a[j][1]>w)

{

a[j][3]=w/a[j][1];

w=0;

}

j++;

}

while(j<n)

{

a[j][3]=0;

j++;

}

for(i=0;i<n;i++)

{

profit+=a[i][0]\*a[i][3];

}

print(a,n);

printf("Profit is %f",profit);

}

**KRUSKAL’S ALGO**

#include<stdio.h>

int i,j,k,a,b,u,v,n,ne=1,min,mincost=0,cost[9][9],parent[9];

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;

}

void main()

{

printf("Enter 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);

}

**PRIM’S ALGO**

#include<stdio.h>

int a,b,u,v,n,i,j,ne=1,visited[10]={0},min,mincost=0,cost[10][10];

void main()

{

printf("Enter the number of nodes:");

scanf("%d",&n);

printf("\nEnter the adjacency matrix:\n");

for(i=1;i<=n;i++)

for(j=1;j<=n;j++)

{

printf("%d,%d:",i,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);

}

**OPTIMAL BINARY SEARCH TREE**

#include<stdio.h>

void main()

{

int w[10][10], c[10][10], r[10][10], data[10], prob[10];

int temp=0, root, min, min1, n;

int i,j,k,b;

printf("Enter the number of elements:");

scanf("%d",&n);

printf("\n");

for(i=1; i <= n; i++)

{

printf("Enter the Element of %d:",i);

scanf("%d",&data[i]);

}

printf("\n");

for(i=0; i <= n; i++)

{

printf("Enter the Probability of %d:",i);

scanf("%d",&prob[i]);

}

printf("W\t\tC\t\tR\n");

for(i=0; i <= n; i++)

{

for(j=0; j <= n; j++)

{

if(i == j)

{

w[i][j] = prob[i];

c[i][j] = 0;

r[i][j] = 0;

printf("W[%d][%d]: %d\tC[%d][%d]: %d\tR[%d][%d]: %d\n",i,j,w[i][j],i,j,c[i][j],i,j,r[i][j]);

}

}

}

printf("\n");

for(b=0; b < n; b++)

{

for(i=0,j=b+1; j < n+1 && i < n+1; j++,i++)

{

if(i!=j && i < j)

{

w[i][j] = data[j] + prob[j] + w[i][j-1];

min = 30000;

for(k = i+1; k <= j; k++)

{

min1 = c[i][k-1] + c[k][j] + w[i][j];

if(min > min1)

{

min = min1;

temp = k;

}

}

c[i][j] = min;

r[i][j] = temp;

}

printf("W[%d][%d]: %d\tC[%d][%d]: %d\tR[%d][%d]: %d\n",i,j,w[i][j],i,j,c[i][j],i,j,r[i][j]);

}

printf("\n");

}

printf("Minimum cost = %d\n",c[0][n]);

root = r[0][n];

printf("Root = %d \n",root);

}

**TRAVELLING SALESMAN PROBLEM**

#include<stdio.h>

int matrix[25][25], visited\_cities[10], limit, cost = 0;

int tsp(int c)

{

int count, nearest\_city = 999;

int minimum = 999, temp;

for(count = 0; count < limit; count++)

{

if((matrix[c][count] != 0) && (visited\_cities[count] == 0))

{

if(matrix[c][count] < minimum)

{

minimum = matrix[count][0] + matrix[c][count];

}

temp = matrix[c][count];

nearest\_city = count;

}

}

if(minimum != 999)

{

cost = cost + temp;

}

return nearest\_city;

}

void minimum\_cost(int city)

{

int nearest\_city;

visited\_cities[city] = 1;

printf("%c ", city + 'A');

nearest\_city = tsp(city);

if(nearest\_city == 999)

{

nearest\_city = 0;

printf("%c", nearest\_city + 'A');

cost = cost + matrix[city][nearest\_city];

return;

}

minimum\_cost(nearest\_city);

}

void main()

{

int i, j;

printf("Enter Total Number of Cities:\t");

scanf("%d", &limit);

printf("\nEnter Cost Adjacency Matrix\n");

for(i = 0; i < limit; i++)

{

for(j = 0; j < limit; j++)

{

scanf("%d", &matrix[i][j]);

}

visited\_cities[i] = 0;

}

printf("\nEntered Cost Matrix\n");

for(i = 0; i < limit; i++)

{

printf("\n");

for(j = 0; j < limit; j++)

{

printf("%d ", matrix[i][j]);

}

}

printf("\n\nPath:\t");

minimum\_cost(0);

printf("\n\nMinimum Cost: \t");

printf("%d\n", cost);

}

**8 QUEEN PROBLEM**

#include<stdio.h>

int x[100];

void nqueen(int k,int n)

{

int i,z,w;

for(i=1;i<=n;i++)

{

if (place(k,i))

{

x[k]=i;

if(k==n)

{

printf("\nSolution:\n");

printf("1\t2\t3\t4\n");

for(z=1;z<=n;z++)

{

for(w=1;w<=n;w++)

{

if(w==x[z])

printf("Q\t");

else

printf(".\t");

}

printf("\n");

}

}

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()

{

int n;

printf("Enter the number of Queens: ");

scanf("%d",&n);

printf("Possible solutions are:");

nqueen(1,n);

}

**LONGEST COMMON SUBSEQUENCE**

#include<stdio.h>

#include<string.h>

void main()

{

int i,j,n1,n2,lcs[10][10];

char a[10],b[10];

printf("Enter first and second strings:\n");

gets(a);

gets(b);

n1=strlen(a);

n2=strlen(b);

for(i=0;i<=n2;i++)

{

for(j=0;j<=n1;j++)

{

if(i==0 || j==0)

lcs[i][j]=0;

}

}

for(i=0;i<=n2;i++)

{

for(j=0;j<=n1;j++)

{

if(b[i]==a[j])

{

lcs[i+1][j+1]=lcs[i][j]+1;

}

else

{

if(lcs[i+1][j]>lcs[i][j+1])

lcs[i+1][j+1]=lcs[i+1][j];

else

lcs[i+1][j+1]=lcs[i][j+1];

}

}

}

printf("\t");

for(i=0;i<=n1;i++)

printf("\t%c",a[i]);

printf("\n");

for(i=0;i<=n2;i++)

{

if(i!=0)

printf("%c\t",b[i-1]);

else

printf("\t");

for(j=0;j<=n1;j++)

{

printf("%d\t",lcs[i][j]);

}

printf("\n");

}

printf("Longest common substring is %d",lcs[i-1][j-1]);

}