**Bubble sort**

#include<stdio.h>

#include<conio.h>

#include<time.h>

void bubblesort(int n,int a[50])

{

int i,j,temp;

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

{

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

{

if(a[i]>a[i+1])

{

temp=a[i];

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

a[i+1]=temp;

}

}

}

}

/\*

void main()

{

int n=7,i;

int a[10000];

clrscr();

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

{

a[i]=rand();

}

printf("unsorted array is\n");

for(i=0;i<=7;i++)

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

bubblesort(n,a);

printf("sorted array\n");

for(i=0;i<=7;i++)

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

getch();

}

\*/ void main()

{

int i,n,k,low,high;

float a[1000],result;

clock\_t start,end;

clrscr();

n=100;

for(k=1;k<=10;k++)

{

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

{

a[i]=rand();

}

start=clock();

bubblesort(n,a);

end=clock();

result=(end-start)/CLK\_TCK;

printf("time taken %d is %f\n",n,result);

n=n+100;

}

getch();

}

**Merge sort**

#include<stdio.h>

#include<conio.h>

#include<time.h>

#include<stdlib.h>

void simple\_merge(float a[1000],int low,int mid,int high)

{

int i,j,k;

float c[1000];

i=low;

j=mid+1;

k=low;

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

{

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

{

c[k]=a[i];

k++;

i++ ;

}

else

{

c[k]=a[j];

k++;

j++;

}

}

while(i<=mid)

{

c[k]=a[i];

k++;

i++;

}

while(j<=high)

{

c[k]=a[i];

k++;

j++;

}

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

{

a[i]=c[i];

}

}

void mergesort(float a[1000],int low,int high)

{

int mid;

if(low<high)

{

mid=(low+high)/2;

mergesort(a,low,mid);

mergesort(a,mid+1,high);

simple\_merge(a,low,mid,high);

}

delay(1);

} /\*

void main()

{

float a[1000];

int i,n,k;

clock\_t start,end;

float result;

clrscr();

//printf("enter the size of the array\n");

//scanf("%d",&n);

n=100;

for(k=1;k<=6;k++)

{

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

a[i]=rand();

start=clock();

//delay(3000);

mergesort(a,0,n-1);

end=clock();

//printf("sorted array elements are\n");

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

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

result=(end-start)/CLK\_TCK;

printf("\n time for %d %f",n,result);

n=n+100;

}

getch();

}

\*/

void main()

{

float a[1000];

int n,i;

clrscr();

n=5;

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

a[i]=rand();

printf("the unsorted array is \n");

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

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

mergesort(a,0,n-1);

printf("\n\n the sorted array is ");

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

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

getch();

}

**Quick sort**

#include<stdio.h>

#include<math.h>

#include<time.h>

#include<conio.h>

#include<stdlib.h>

#include<stdio.h>

#include<dos.h>

//algorithm for Quicksort(a,low,high)

int partition(float a[10000],int low,int high);

void quicksort(float a[10000],int low,int high)

{

int k;

if(low>high)

return;

k=partition(a,low,high);

quicksort(a,low,k-1);

quicksort(a,k+1,high);

delay(1);

}

//algorithm for Partition(A,low,high)

int partition(float a[10000],int low,int high)

{

int i,j,key,t;

key=a[low];

i=low;

j=high+1;

while(i<=j)

do

{

i=i+1;

}

while(key>=a[i]);

do

{

j=j-1;

}

while(key<a[j]);

{

if(i<j)

{

t=a[i];

a[i]=a[j];

a[j]=t;

}

}

t=a[low];

a[low]=a[j];

a[j]=t;

return j;

}/\*

void main()

{

float a[10000];

int i,n,k;

clock\_t start,end;

float result;

clrscr();

n=100;

for(k=1;k<=10;k++)

{

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

a[i]=rand();

start=clock();

quicksort(a,0,n-1);

end=clock();

result=(end-start)/CLK\_TCK;

printf("\ntime taken to sort %d elements=%f\n",n,result);

n=n+100;

}

getch();

}

\*/

void main()

{

int i,n;

float a[10000];

clrscr();

n=5;

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

a[i]=rand();

printf("the unsorted array is \n");

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

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

quicksort(a,0,n-1);

printf("\n\n the sorted array is ");

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

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

getch();

}

**Selection sort**

#include<stdio.h>

#include<conio.h>

#include<time.h>

#include<math.h>

void selectionsort(int n,int a[])

{

int i,j,min,temp,pos;

for(j=0;j<n-1;j++)

{

min=a[j];

pos=j;

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

{

if(a[i]<min)

{

min=a[i];

pos=i;

}

}

temp=a[j];

a[j]=a[pos];

a[pos]=temp;

}

}/\*

void main()

{

int n,i,a[10000];

float res;

clock\_t end,start;

clrscr();

printf("enter the no. of elements\n");

scanf("%d",&n);

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

a[i]=rand();

start=clock();

delay(10);

selectionsort(n,a);

end=clock();

res=(end-start)/CLK\_TCK;

printf("the sorted array is\n");

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

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

printf("time for sorting %f",res);

getch();

} \*/

void main()

{

int n=7,i;

int a[10000];

clrscr();

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

{

a[i]=rand();

}

printf("unsorted array is\n");

for(i=0;i<=7;i++)

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

selectionsort(n,a);

printf("sorted array\n");

for(i=0;i<=7;i++)

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

getch();

}

**Dijikstras algorithm:**

#include<stdio.h>

#include<conio.h>

#define infinity 999

void dijkstras(int cost[10][10],int n,int source,int distance[])

{

int visited[10],min,u;

int i,j;

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

{

distance[i]=cost[source][i];

visited[i]=0;

}

visited[source]=1;

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

{

min=infinity;

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

{

if(visited[j]==0 && distance[j]<min)

{

min=distance[j];

u=j;

}

}

visited[u]=1;

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

{

if(visited[j]==0 &&(distance[u]+cost[u][j])<distance[j])

{

distance[j]=distance[u]+cost[u][j];

}

}

}

}

void main()

{

int n,cost[10][10],distance[10];

int i,j,source,sum;

clrscr();

printf("enter the no. of nodes\n");

scanf("%d",&n);

printf("\nenter cost matrix\nenter 999 for no edge\n");

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

{

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

{

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

}

}

printf("enter the source node\n");

scanf("%d",&source);

dijkstras(cost,n,source,distance);

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

{

printf("\nshortest distance from %d to %d is %d\n",source,i,distance[i]);

}

getch();

}

**Kruskal’s algorithm:**

#include<stdio.h>

#include<conio.h>

#define INFINITY 999

int find(int v,int p[100])

{

while(p[v]!=v)

{

v=p[v];

}

return v;

}

void kruskal(int cost[100][100],int n)

{

int i,j,k,u,v,sum,t[100][2],p[100], mincost,count;

count=0;

k=0;

sum=0;

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

{

p[i]=i;

}

while(count<n-1)

{

mincost=INFINITY;

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

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

if((cost[i][j]!=0)&&(cost[i][j]<mincost))

{

mincost=cost[i][j];

u=i;

v=j;

}

if(mincost==INFINITY)

break;

i=find(u,p);

j=find(v,p);

if(i!=j)

{

t[k][0]=u;

t[k][1]=v;

k++;

count++;

sum=sum+mincost;

if(i<j)

p[i]=i;

else

p[i]=j;

}

cost[u][v]=cost[v][u]=INFINITY;

}

if(count==n-1)

{

printf("edges of spanning tree are\n");

for(i=0;i<n-1;i++)

printf("%d->%d\n",t[i][0],t[i][1]);

printf("sum=%d",sum);

}

else

printf("spanning tree does not exit\n");

}

void main()

{

int i,j,cost[100][100],n;

clrscr();

printf("enter no. of vertices\n");

scanf("%d",&n);

printf("enter cost adjacency matrix\n");

printf("enter 999 for no edge\n");

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

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

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

kruskal(cost,n);

getch();

}

**Topological sorting:**

#include<stdio.h>

#include<conio.h>

void findindegree(int a[10][10],int n,int indegree[10])

{

int i,j,sum;

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

{

sum=0;

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

{

sum+=a[i][j];

}

indegree[j]=sum;

}

}

void topologicalsort(int a[10][10],int n,int indegree[10])

{

int s[10],u,v,t[10],i;

int top=-1;

int k=0;

findindegree(a,n,indegree);

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

{

if(indegree[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)

{

indegree[v]--;

if(indegree[v]==0)

{

s[++top]=v;

}

}

}

}

printf("the topological sequence is \t");

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

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

printf("\n");

}

void main()

{

int n,i,j,a[10][10],indegree[10];

clrscr();

printf("enter the size:\n");

scanf("%d",&n);

printf("enter the adjacency mATRIX:\n");

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

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

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

topologicalsort(a,n,indegree) ;

getch();

}

**BFS:**

#include<stdio.h>

#include<conio.h>

void bfs(int a[10][10],int n,int src,int t[10][2])

{

int k,s[10],i,f,r,u,v,q[10] ;

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

s[i] = 0 ;

f = 0 ;

r = 0 ;

k = 0 ;

q[r] = src ;

s[src] = 1 ;

while(f<=r)

{

u = q[f] ;

f = f + 1;

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

{

if(a[u][v] == 1 && s[v]==0)

{

s[v] = 1 ;

r = r+1 ;

q[r] = v ;

t[k][0] = u ;

t[k][1] = rfx ;

k = k+1 ;

}

}

}

}

void main()

{

int n,i,j,a[10][10],src,t[10][2] ;

clrscr() ;

printf("Enter the number of vertices\n");

scanf("%d",&n);

printf("Enter the adjacency matrix\n");

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

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

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

printf("Enter source node\n");

scanf("%d",&src);

bfs(a,n,src,t);

printf("Nodes which are reachable are\n");

for(i=0;i<n-1;i++)

printf("(%d,%d)\n",t[i][0],t[i][1]);

getch() ;

}

**Prims:**

#include<stdio.h>

#include<conio.h>

#define INFINITY 999

void prims(int cost[100][100],int n,int visited[100],int p[100],int d[100])

{

int mincost,count,i,j,u,v,source,t[100][2],k,sum;

count=0;

k=0;

sum=0;

mincost=INFINITY;

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

{

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

{

if(cost[i][j]<mincost)

{

mincost=cost[i][j];

source=i;

}

}

}

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

{

visited[i]=0;

d[i]=cost[source][i];

p[i]=source;

}

visited[source]=i;

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

{

u=-1;

mincost=INFINITY;

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

{

if((visited[j]==0) && (d[j]<mincost))

{

mincost=d[j];

u=j;

}

}

if(u==-1)

{

break;

}

visited[u]=1;

t[k][0]=p[u];

t[k][1]=u;

count++;

k++;

sum=sum+mincost;

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

{

if((visited[v]==0) && (cost[u][v]<d[v]))

{

d[v]=cost[u][v];

p[v]=u;

}

}

}

if(count==n-1)

{

printf("spamming tree exists\n edges of the spamming tree is\n");

for(i=0;i<n-1;i++)

{

printf("%d->%d\n",t[i][0],t[i][1]);

}

printf("\nsum=%d",sum);

}

else

printf("spamming tree doesn't exist\n");

}

void main()

{

int n,i,j,cost[100][100],visited[100],p[100],d[100];

clrscr();

printf("enter the no. of vertices in the graph\n");

scanf("%d",&n);

printf("enter the cost adjacency matrix\n");

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

{

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

{

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

}

}

prims(cost,n,visited,p,d);

getch();

}

**Travelling salesmen problem:**

#include<stdio.h>

#include<conio.h>

int x[15],used[15];

int adj[15][15]={0};

int path[15][15],wt[15];

int c,min;

int p\_ok(int k,int n)

{

if(used[x[k]])

return 0;

if(k<n-1)

return(adj[x[k-1]][x[k]]);

else

return(adj[x[k-1]][x[k]] && adj[x[k]][x[0]]);

}

void tsp(int k , int n)

{

int i,sum;

for(x[k]=1;x[k]<n;x[k]++)

{

if(p\_ok(k,n))

{

used[x[k]]=1;

if(k==n-1)

{

sum=0;

printf("\npossible path %d : ",c+1);

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

{

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

path[c][i]=x[i];

sum+=adj[x[i]][x[i+1]];

}

printf("sum =%d\n",sum);

wt[c]=sum;

if(c==0||sum<min)

min=sum;

c++;

used[x[k]]=0;

getch();

}

else

tsp(k+1,n);

used[x[k]]=0;

}

}

}

void find(int n)

{

int i,j;

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

{

if(wt[i]==min)

{

printf("min path ");

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

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

}

}

}

void main()

{

int i,n,j,ed;

clrscr();

printf("\ntsp\n");

printf("enter the number of cites");

scanf("%d",&n);

printf("enter the cost if path exists,else 0 \n");

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

{

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

{

scanf("%d",&ed);

if(ed)

adj[i][j]=adj[i][j]=ed;

}

}

used[0]=1;

tsp(1,n);

if(!c)

printf("no path to cover all cities");

else

{

printf("min cost is %d",min);

find(n);

}

getch();

}

**Warshall’s**

#include<stdio.h>

#include<conio.h>

void warshall(int n,int a[10][10],int p[10][10])

{

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 read\_adjacency\_matrix(int n,int a[10][10])

{

int i,j;

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

{

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

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

}

}

void write\_matrix(int n,int a[10][10])

{

int i,j;

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

{

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

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

printf("\n");

}

}

void main()

{

int n,a[10][10],p[10][10];

clrscr();

printf("enter number of nodes");

scanf("%d",&n);

printf("enter adjacency matrix\n");

read\_adjacency\_matrix(n,a);

warshall(n,a,p);

printf("the transitive closure shown below\n");

write\_matrix(n,p);

getch();

}

**Floyd’s**

#include<stdio.h>

#include<conio.h>

#define infinity 999

int min(int a, int b)

{

return (a<b)?a:b;

}

void floyd(int n, int a[10][10], int d[10][10])

{

int i,j,k;

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

{

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

{

d[i][j]=a[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]);

}

}

}

}

void main()

{

int i,j,k,a[10][10],d[10][10],n;

clrscr();

printf("enter no. of nodes\n");

scanf("%d",&n);

printf("enter the adjacency matrix\n");

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

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

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

printf("distance matrix:\n");

floyd(n,a,d);

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

{

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

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

printf("\n");

}

getch();

}

**Horsepool:**

#include<stdio.h>

#include<conio.h>

#include<string.h>

void shifttable(char p[100],int s[100])

{

int i,m;

m=strlen(p);

for(i=0;i<=127;i++)

s[i]=m;

for(i=0;i<=m-2;i++)

{

s[p[i]]=m-1-i;

}

}

int horspool(char p[50], char t[50], int s[50])

{

int m,n,i,k;

n=strlen(t);

m=strlen(p);

i=m-1;

while(i<=n-1)

{

k=0;

while(k<=m-1 && t[i-k]==p[m-1-k])

k=k+1;

if(k==m) return(i-m+1);

i=i+s[t[i]];

}

return -1;

}

void main()

{

int s[50],res;

char p[50],t[50];

clrscr();

printf("enter text:\n");

scanf("%s",t);

printf("enter the pattern to match:\n");

scanf("%s",p);

shifttable(p,s);

res=horspool(p,t,s);

if(res==-1)

printf("pattern not found in the given text\n");

else

printf("pattern found at position %d\n",res);

getch();

}

**Knapsack**

#include<stdio.h>

#include<conio.h>

int max(int a,int b)

{

return a>b?a:b;

}

void knapsack(int n,int w[],int m,int v[10][10],int p[])

{

int i,j;

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])

v[i][j]=v[i-1][j];

else

v[i][j]=max(v[i-1][j],v[i-1][j-w[i]]+p[i]);

}

}

}

void print\_opimal\_solution(int n,int m,int w[],int v[10][10])

{

int i,j,x[10];

printf("the optimal solution is %d\n",v[n][m]);

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

x[i]=0;

i=n;

j=m;

while(i!=0 &&j!=0)

{

if(v[i][j]!=v[i-1][j])

{

x[i]=1;

j=j-w[i];

}

i=i-1;

}

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

{

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

}

printf("=");

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

{

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

}

}

void main()

{

int m,i,j,n,p[10],w[10],v[10][10];

clrscr();

printf("enter the number of objects\n");

scanf("%d",&n);

printf("enter the weight of n objects\n");

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

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

printf("enter the profits of n objects\n");

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

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

printf("enter the capacity of knapsack\n");

scanf("%d",&m);

knapsack(n,w,m,v,p);

printf("the output is \n");

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

{

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

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

printf("\n");

}

print\_opimal\_solution(n,m,w,v);

getch();

}

**DFS**

#include<stdio.h>

#include<conio.h>

void read(int n, int cost[10][10])

{

int i,j;

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

{

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

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

}

}

void dfs(int n, int cost[20][20], int u, int s[])

{

int v;

s[u]=1;

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

{

if(cost[u][v]==1 && s[v]==0)

dfs(n,cost,v,s);

}

}

int connectivity(int n, int cost[20][20])

{

int i,j,flag,s[10];

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

{

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

s[i]=0;

dfs(n,cost,j,s);

flag=0;

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

{

if(s[i]==0)

flag=1;

}

if(flag==0)

return 1;

}

return 0;

}

void main()

{

int n, cost[20][20],flag,i,j;

clrscr();

printf("enter the no. of vertices\n");

scanf("%d",&n);

printf("enter the adjacency matrix\n");

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

{

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

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

}

flag = connectivity(n,cost);

if(flag==1)

printf("graph is connected\n");

else

printf("graph is disconnected\n");

getch();

}

Queen

#include<stdio.h>

#include<conio.h>

void print\_sol(int x[],int n)

{

static int count=0;

int i,j;

char a[10][10];

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

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

a[i][j]='X';

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

a[i][x[i]]='Q';

printf("%d solution is\n",++count);

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

{

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

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

printf("\n");

}

}

int place(int x[],int k)

{

int i;

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

if(x[i]==x[k] || (i-x[i])==(k-x[k]) || (i+x[i])==(k+x[k]))

return 0;

return 1;

}

void nqueen(int n)

{

int x[10],k,flag=0;

k=1;

x[k]=0;

while(k!=0)

{

x[k]=x[k]+1;

while(x[k]<=n && !place(x,k))

x[k]=x[k]+1;

if(x[k]<=n)

{

if(k==n)

{

print\_sol(x,n);

flag=1;

}

else

{

k++;

x[k]=0;

}

}

else

k--;

}

if(flag==0)

printf("No solution\n");

}

void main()

{

int n;

clrscr();

printf("Enter the number of queens\t");

scanf("%d",&n);

nqueen(n);

getch();

}

**Subset prgm**

#include<stdio.h>

#include<conio.h>

int count,w[10],d,x[10];

void subset(int cs,int k,int r)

{

int i;

x[k]=1;

if(cs+w[k]==d)

{

printf("\n subset solution=%d\n",++count);

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

{

if(x[i]==1)

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

}

}

else if(cs+w[k]+w[k+1]<=d)

subset(cs+w[k],k+1,r-w[k]);

if(cs+r-w[k]>=d && cs+w[k+1]<=d)

{

x[k]=0;

subset(cs,k+1,r-w[k]);

}

}

void main()

{

int sum=0,i,n;

clrscr();

printf("enter no of elements\n");

scanf("%d",&n);

printf("enter the elements in ascending order\n");

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

{

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

}

printf("enter the required sum\n");

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

{

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

}

sum+=w[i];

if(sum<d)

{

printf("no solution exist\n");

return;

}

printf("solution is\n");

count=0;

subset(0,0,sum);

getch();

}