**/\*Quick Sort using Divide and Conquer\*/**  
**#include<stdio.h>**  
**#include<conio.h>**  
**int arr[40];**  
**void quicksort(int a[],int p,int r);**  
**int partition(int a[],int p,int r);**  
**void exchange(int i,int j);**  
**void quicksort(int a[],int p,int r){**  
**int q;**  
**if(p<r){**  
**q=partition(a,p,r);**  
**quicksort(a,p,q-1);**  
**quicksort(a,q+1,r);**  
**}**  
**}**  
**int partition(int a[],int p,int r){**  
**int x,j,i;**  
**x=a[r];**  
**i=p-1;**  
**for(j=p;j<=(r-1);j++)**  
**if(a[j]<x){**  
**i=i+1;**  
**exchange(i,j);**  
**}**  
**exchange(i+1,r);**  
**return(i+1);**  
**}**  
**void exchange(int i,int j){**  
**int temp;**  
**temp=arr[i];**  
**arr[i]=arr[j];**  
**arr[j]=temp;**  
**}**  
**int main(){**  
**int n,i;**  
**printf("\nEnter no . elements needed : ");**  
**scanf("%d",&n);**  
**printf("\nEnter elements : ");**  
**for(i=1;i<=n;i++)**  
**scanf("%d",&arr[i]);**  
**quicksort(arr,1,n);**  
**printf("\nSorted Array is : ");**  
**for(i=1;i<=n;i++)**  
**printf("%4d",arr[i]);**  
**getch();**  
**return(0);**  
**}**

3.[Write a program to find maximum and minimum number.](http://thinkbd.com/dcc/algorithm/lab/3-maxmin.cpp" \o "Write a program to find maximum and minimum number." \t "_blank)

#include<stdio.h>

#include<conio.h>

int maxmin(int a[], int left, int right ,int &max,int &min)

{

int max1, min1, mid;

if(left==right)

max=min=a[left];

else if(left==right-1)

{

if(a[left]>a[right])

{

max=a[left];

min=a[right];

}

else

{

max=a[right];

min=a[left];

}

}

else

{

mid=(left+right)/2;

maxmin(a,left,mid,max,min);

maxmin(a,mid+1,right,max1,min1);

if(max1>max)

max=max1;

if(min1<min)

min=min1;

}

return 0;

}

int main()

{

int a[20],i,n,max,min;

printf("Enter how many number: ");

scanf("%d",&n);

printf("Enter one after another.....\n");

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

{

printf("%d ",i);

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

}

maxmin(a,0,n-1,max,min);

printf("\nmaximum=%d",max);

printf("\nMinimum=%d",min);

getch();

return 0;

}

4.[Write a program to find an item using Binary Search](http://thinkbd.com/dcc/algorithm/lab/4-binary-search.cpp" \o "Write a program to find an item using Binary Search" \t "_blank)

#include<stdio.h>

int a[100];

bool bSearch(int a[], int n, int item){

int start,end,mid;

start =0;

end = n-1;

while(start<=end){

mid = (start+end)/2;

if(a[mid] == item)

return true;

else if(a[mid] < item)

start = mid+1;

else

end = mid-1;

}

return false;

}

int main(){

int n,i,item;

printf("Enter number of items: ");

scanf("%d",&n);

printf("\nEnter item list: ");

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

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

}

printf("\nEnter item to search: ");

scanf("%d",&item);

if(bSearch(a,n,item) == true){

printf("%d is found\n",item);

}else{

printf("%d is not found\n",item);

}

return 0;

}

5. [Compare the performance of Bubble sort and Quick sort algorithm.](http://thinkbd.com/dcc/algorithm/lab/5-bubble-quick.cpp)

#include<iostream>

#include<conio.h>

#include<time.h>

#include<stdlib.h>

using namespace std;

void bubble(int a[],int n)

{

int i,j,temp;

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

{

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

{

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

{

temp=a[j+1];

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

a[j]=temp;

}

}

}

}

void quick(int a[],int left,int right)

{

int i=left,j=right,mid;

mid=(a[i]+a[j])/2;

while(i<=j)

{

while(a[i]<mid)

i++;

while(a[j]>mid)

j--;

if(i<=j)

{

int temp;

temp=a[i];

a[i]=a[j];

a[j]=temp;

i++;

j--;

}

}

if(left<j)

quick(a,left,j);

if(i<right)

quick(a,i,right);

}

int main()

{

int n = 4500;

int i; int b[n];

clock\_t sb,eb,sq,eq;

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

b[i]=rand() % n;

sb=clock();

bubble(b,n);

eb=clock();

cout<<"Buble"<<((eb-sb)/CLK\_TCK)<<"\n";

sq=clock();

quick(b,0,n-1);

eq=clock();

cout<<"Quick"<<((eq-sq)/CLK\_TCK);

getch();

return 0;

}

6. [Compare the performance of Quick sort and Merge sort algorithm.](http://thinkbd.com/dcc/algorithm/lab/6-quick-merge.cpp)

#include<iostream>

#include<conio.h>

#include<time.h>

#include<stdlib.h>

using namespace std;

void quick(int a[],int left,int right)

{

int i=left,j=right,mid;

mid=(a[i]+a[j])/2;

while(i<=j)

{

while(a[i]<mid)

i++;

while(a[j]>mid)

j--;

if(i<=j)

{

int temp;

temp=a[i];

a[i]=a[j];

a[j]=temp;

i++;

j--;

}

}

if(left<j)

quick(a,left,j);

if(i<right)

quick(a,i,right);

}

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

{

int c = low+high;

int b[c];

int h=low;

int i=low;

int j=mid+1;

int k;

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

{

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

b[i++]=a[h++];

else

b[i++]=a[j++];

}

if(h>mid)

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

b[i++]=a[k];

else

for(k=h;k<=mid;k++)

b[i++]=a[k];

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

a[k]=b[k];

}

void Mergesort(int a[],int low,int high)

{

if (low<high)

{

int mid=(low+high)/2;

Mergesort(a,low,mid);

Mergesort(a,mid+1,high);

Merge(a,low,mid,high);

}

}

int main()

{

int n = 4500;

int i; int b[n];

clock\_t sb,eb,sq,eq;

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

b[i]=rand()%n;

sb=clock();

quick(b,0,n-1);

eb=clock();

cout<<"Quick"<<((eb-sb)/CLK\_TCK)<<"\n";

sq=clock();

Mergesort(b,0,n-1);

eq=clock();

cout<<"Merge"<<((eq-sq)/CLK\_TCK);

getch();

return 0;

}

7. [Write a program to find all pairs shortest path](http://thinkbd.com/dcc/algorithm/lab/7-shortestpathall.cpp)

#include<iostream>

#include<conio.h>

#define max 100

#define inf 1000

using namespace std;

int n;

int m[max][max],W[max][max];

void input\_data()

{

int i,j,x,y,w;

cout<<"Enter the number of vertices: ";

cin>>n;

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

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

m[i][j]=inf;

cout<<"input -1,-1,-1,-1 to stop inputting edges.\n";

cout<<"Enter the value of x: "; cin>>x;

cout<<"Enter the value of y: "; cin>>y;

cout<<"Enter the value of w: "; cin>>w;

while(!((x==-1)&&(y==-1)&&(w==-1)))

{

if((x<0)||(x>n)||(y<0)||(y>n))

cout<<"Invalid node numbers given.Must be between 0 and n-1\n";

else

m[x][y]=w;

cout<<"Enter the value of x: "; cin>>x;

cout<<"Enter the value of y: "; cin>>y;

cout<<"Enter the value of w: "; cin>>w;

}

}

int min(int x,int y)

{

if(x<y)

return x;

else

return y;

}

void shortestpath()

{

int i,j,k,x,y;

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

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

W[i][j]=m[i][j];

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

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

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

W[i][j]=min(W[i][j],W[i][k]+W[k][j]);

cout<<"Enter source & destination node numders: (-1 -1 to exit)\n";

cin>>x>>y;

while((x!=-1)&&(y!=-1))

{

if(W[x][y]==inf)

cout<<"No path.\n";

else

cout<<"Path length is: "<<W[x][y];

cin>>x>>y;

}

}

int main()

{

//clrscr();

int option;

do

{

cout<<"Graph manipulation program\n";

cout<<"----------------------------\n";

cout<<"Enter option: ";

cin>>option;

cout<<"----------------------------\n";

switch(option)

{

case 1:

input\_data();

break;

case 2:

if(n!=0)

shortestpath();

break;

}

}while(option!=3);

getch();

return 0;

}

8. [Write a program to find the minimum cost spanning tree using prim's algorithm](http://thinkbd.com/dcc/algorithm/lab/8-prims-algorithm.cpp)

#include<stdio.h>

#include<iostream>

#include<conio.h>

#define INF 25000

#define R 50

using namespace std;

int total, n,t[R][3],cost[R][R];

int input();

void display(int mincost);

int prim();

int main()

{

int q,mincost;

do

{

cout<<"1.input data\n";

cout<<"2.Evalute minimum-cost spanning tree\n";

cout<<"3.Quit\n";

cout<<"Enter choice:";

cin>>q;

if(q==1) n=input();

else if(q==2)

{

mincost=prim();

display(mincost);

}

}

while(q!=3);

return 0;

}

void display(int mincost)

{

int i;

cout<<"\nprim's minimum-cost spanning tree is"<<mincost<<"\n";

cout<<"The spanning tree is:\n";

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

cout<<t[i][1]<<"->"<<t[i][2]<<"\n";

cout<<"\n\n";

}

int input()

{

int V,x,y,kost;

cout<<"Enter number of vertices:";

cin>>V;

for(x=0;x<=V;x++)

for(y=0;y<=V;y++)

{

cost[x][y]=INF;

cost[y][x]=INF;

}

do

{

cout<<"Enter edge(-1,-1 to quit):";

cin>>x>>y;

if(x==-1||y==-1)break;

cout<<"Enter cost:";

cin>>kost;

cost[x][y]=kost;

cost[y][x]=kost;

}

while(1);

return V;

}

int prim()

{

int i,j,k,l,mincost,adjacent[R];

k=l=1;

total=0;

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

{

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

if(cost[i][j]<cost[k][l])

{

k=i;

l=j;

}

}

mincost=cost[k][l];

t[1][1]=k;t[1][2]=l;total++;

for(i=2;i<=n;i++)

{

if(cost[i][l]<cost[i][k])adjacent[i]=l;

else

adjacent[i]=k;

}

adjacent[k]=adjacent[l]=0;

for(i=2;i<=n;i++)

{

l=INF;

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

{

if(cost[k][adjacent[k]]<l)

{

l=cost[k][adjacent[k]];

j=k;

}

}

t[i][1]=adjacent[j];

t[i][2]=j;

total++;

mincost=mincost+cost[j][adjacent[j]];

adjacent[j]=0;

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

if(adjacent[k]!=0&&cost[k][adjacent[k]]>cost[k][j])

adjacent[k]=j;

}

return mincost;

}

9. [Write a program to solve 0/1 knapsack problem using dynamic programming](http://thinkbd.com/dcc/algorithm/lab/9-0-1knapsack.cpp)

#include<iostream>

#include<conio.h>

#include<stdio.h>

using namespace std;

int max(int a ,int b)

{

if(a>b)

return a;

else

return b;

}

int main()

{

int textcolor[3];

int clrscr();

int a[100][100],w[100],p[100],m,n,i,j;

printf("How many element:");

cin>>n;

printf("Bag size:");

cin>>m;

printf("...........\n");

printf("Enter price & weight:\n");

printf("...........\n");

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

{

printf("%d:",i);

textcolor[95];

printf("Enter weight:");

cin>>w[i];

printf("Enter price:");

cin>>p[i];

}

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

a[0][j]=0;

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

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

{

if(w[i]>j)

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

else

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

}

cout<<"\n\n";

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

{

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

{

textcolor[28];

printf("%5d",a[i][j]);

}

cout<<"\n";

}

textcolor[128];

printf("\n Total profit:%d ",a[n][m]);

getch();

return 0;

}

10. [Write a program to solve the n-queens Problem](http://thinkbd.com/dcc/algorithm/lab/10-n-queens.cpp)

#include<iostream>

#include<math.h>

#include<stdlib.h>

#define R 50

using namespace std;

int x[R],count;

int place(int k,int i)

{

int j;

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

{

if(x[j]==i||abs(x[j]-i)==abs(j-k))

return 0;

}

return 1;

}

void NQueens(int k,int n)

{

int i,j;

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

{

if(place(k,i)==1)

{

x[k]=i;

if(k==n)

{

count++;

cout<<"Solution"<<count<<".\n";

cout<<"Row:Column=";

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

cout<<j<<":"<<x[j]<<" ";

cout<<endl;

}

else NQueens(k+1,n);

}

}

}

int main()

{

int n;

cout<<"\*\*\* 8-Queens Chess Problem\*\*\*\n";

do

{

cout<<"\nEnter number of Queens(0 to quit):";

cin>>n;

count=0;

NQueens(1,n);

}while(n);

return 0;

}

12. [Write a program to find the single Source Shortest Path using greedy method (Dijkstra's Algorithm)](http://thinkbd.com/dcc/algorithm/lab/12-single-source-shortest-path.cpp)

#include<iostream>

#define INF 30000

#define R 100

using namespace std;

int n,v,cost[R][R],dist[R];

void input()

{

int x,y,length;

cout<<"Enter number of vertices:";

cin>>n;

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

{

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

cost[x][y]=INF;

cost[x][x]=0;

}

do

{

cout<<"Enter edge(-1,-1 to quit):";

cin>>x>>y;

if(x==-1||y==-1)break;

cout<<"Enter length:";

cin>>length;

cost[x][y]=length;

}

while(1);

cout<<"Enter source:";

cin>>v;

}

void shortestpath()

{

int i,j,temp,u,w;

int s[R];

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

{

s[i]=0;

dist[i]=cost[v][i];

}

s[v]=1;

dist[v]=0;

for(i=2;i<=n-1;i++)

{

temp=INF;

u=1;

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

if(s[j]==0&&dist[j]<temp)

{

u=j;

temp=dist[j];

}

s[v]=1;

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

if(s[w]==0&&dist[w]>dist[u]+cost[u][w])

dist[w]=dist[u]+cost[u][w];

}

}

void display()

{

int i;

cout<<"\nshortest paths from source"<<v<<":\n";

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

cout<<i<<":"<<dist[i]<<" ";

cout<<"\n\n";

}

int main()

{

//freopen("single source.txt","r",stdin);

int q;

do

{

cout<<"1.input data\n";

cout<<"2.Evalute single source shortest path\n";

cout<<"3.quit\n";

cin>>q;

if(q==1)

input();

else

if(q==2)

{

shortestpath();

display();

}

}while(q!=3);

return 0;

}

Gredy KnapsaK:

#include<iostream>

using namespace std;

void quick(int a[],int left,int right){

int i=left,j=right,mid;

mid=(a[i]+a[j])/2;

while(i<=j){

while(a[i]<mid)

i++;

while(a[j]>mid)

j--;

if(i<=j){

int temp;

temp=a[i];

a[i]=a[j];

a[j]=temp;

i++;

j--;

}

}

if(left<j)

quick(a,left,j);

if(i<right)

quick(a,i,right);

}

int main(){

int n,a[100];

cout<<"Enter number of items: ";

cin>>n;

cout<<endl<<"Enter items: ";

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

cin>>a[i];

}

quick(a,0,n-1);

cout<<endl <<"After Quick Sort:"<<endl;;

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

cout<<a[i]<<" ";

}

cout<<endl;

return 0;

}

Using devide and conquer method:

#include<iostream>

using namespace std;

void quick(int a[],int left,int right){

int i=left,j=right,mid;

mid=(a[i]+a[j])/2;

while(i<=j){

while(a[i]<mid)

i++;

while(a[j]>mid)

j--;

if(i<=j){

int temp;

temp=a[i];

a[i]=a[j];

a[j]=temp;

i++;

j--;

}

}

if(left<j)

quick(a,left,j);

if(i<right)

quick(a,i,right);

}

int main(){

int n,a[100];

cout<<"Enter number of items: ";

cin>>n;

cout<<endl<<"Enter items: ";

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

cin>>a[i];

}

quick(a,0,n-1);

cout<<endl <<"After Quick Sort:"<<endl;;

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

cout<<a[i]<<" ";

}

cout<<endl;

return 0;

}