VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT
on

Analysis and Design of Algorithms

Submitted by

K.YASASWINI (1BM20CS066)

in partial fulfillment for the award of the degree of
BACHELOR OF ENGINEERING
in
COMPUTER SCIENCE AND ENGINEERING

B. M. S. College of Engineering,

Bull Temple Road, Bangalore 560019
(Affiliated To Visvesvaraya Technological University, Belgaum)

Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the Lab work entitled “**Analysis and Design of Algorithms**” carried out by **K.YASASWINI(1BM20CS066)** who is Bonafede student of **B. M. S. College of Engineering.** It is in partial fulfillment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum during the year 2022. The Lab report has been approved as it satisfies the academic requirements in respect of a **Analysis and Design of Algorithms - (19CS4PCADA)** work prescribed for the said degree.

Name of the Lab-In charge:	**Dr. Jyothi S Nayak**
NAGARATHNA N	Professor and Head
Department of CSE	Department of CSE
BMSCE, Bengaluru	BMSCE, Bengaluru

`

Index Sheet

Sl. No.	Experiment Title	Page No.
1	Write a recursive program to Solve **a)** Towers-of-Hanoi problem **b)** To find GCD	**5**
2	Implement Recursive Binary search and Linear search and determine the time required to search an element. Repeat the experiment for different values of N and plot a graph of the time taken versus N.	**8**
3	Sort a given set of N integer elements using Selection Sort technique and compute its time taken. Run the program for different values of N and record the time taken to sort.	**12**
4	Write program to do the following: **a)** Print all the nodes reachable from a given starting node in a digraph using BFS method. **b)** Check whether a given graph is connected or not using DFS method.	**14**
5	Sort a given set of N integer elements using Insertion Sort technique and compute its time taken.	**18**
6	Write program to obtain the Topological ordering of vertices in a given digraph.	**21**
7	Implement Johnson Trotter algorithm to generate permutations.	**23**
8	Sort a given set of N integer elements using Merge Sort technique and compute its time taken. Run the program for different values of N and record the time taken to sort.	**26**
9	Sort a given set of N integer elements using Quick Sort technique and compute its time taken.	**28**
10	Sort a given set of N integer elements using Heap Sort technique and compute its time taken.	**30**
11	Implement Warshall's algorithm using dynamic programming	**32**
12	Implement 0/1 Knapsack problem using dynamic programming.	**33**
13	Implement All Pair Shortest paths problem using Floyd's algorithm.	**36**
14	Find Minimum Cost Spanning Tree of a given undirected graph using Prim's algorithm.	**38**
15	Find Minimum Cost Spanning Tree of a given undirected graph using Kruskals algorithm.	**40**
16	From a given vertex in a weighted connected graph, find shortest paths to other vertices using Dijkstra's algorithm.	**42**

17	Implement “Sum of Subsets” using Backtracking. “Sum of Subsets” problem: Find a subset of a given set S = {s1,s2,……,sn} of n positive integers whose sum is equal to a given positive integer d. For example, if S = {1,2,5,6,8} and d = 9 there are two solutions {1,2,6} and {1,8}. A suitable message is to be displayed if the given problem instance doesn’t have a solution.	**46**
18	Implement “N-Queens Problem” using Backtracking.	**44**

Course Outcome

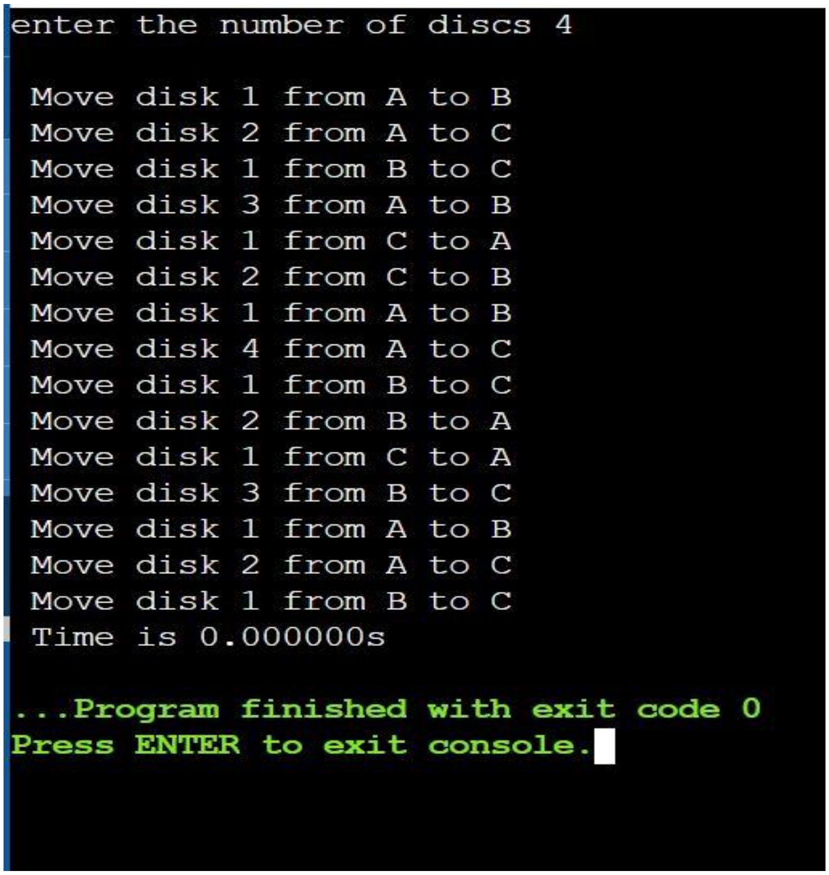
CO1	Ability to **analyze** time complexity of Recursive and Non-Recursive algorithms using asymptotic notations.
CO2	Ability to **design** efficient algorithms using various design techniques.
CO3	Ability to **apply** the knowledge of complexity classes P, NP, and NP-Complete and prove certain problems are NP-Complete
CO4	Ability to **conduct** practical experiments to solve problems using an appropriate designing method and find time efficiency.

Program 1: Write a recursive program to Solve **a)** Towers-of-Hanoi problem **b)** To find GCD

Tower of Hanoi

```
 #include <stdio.h>

#include<time.h>

void towerOfHanoi(int n, char from_rod, char to_rod, char aux_rod)

{

if (n == 1)

    {

    printf("\n Move disk 1 from %c to %c", from_rod, to_rod); return;

    }

    towerOfHanoi(n-1, from_rod, aux_rod, to_rod);

    printf("\n Move disk %d from %c to %c", n, from_rod, to_rod);

    towerOfHanoi(n-1, aux_rod, to_rod, from_rod);

}

int main()

{

    int n;

    time_t start,end;

    printf("enter the number of discs");

    scanf("%d",&n);

    start=time(NULL);

    towerOfHanoi(n,'A','C','B');

    end=time(NULL);

            printf("\n Time is %fs",difftime(end,start));     }
```

OUTPUT:



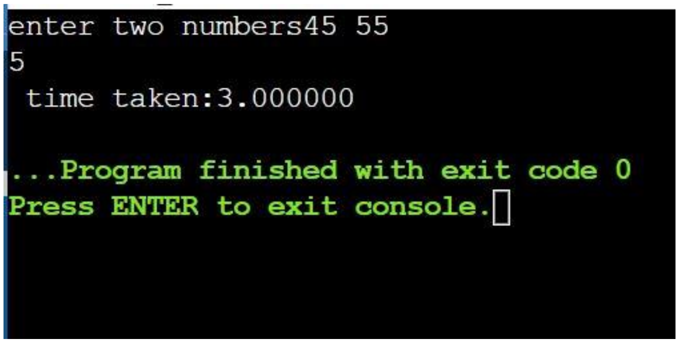
```
enter the number of discs 4

 Move disk 1 from A to B
 Move disk 2 from A to C
 Move disk 1 from B to C
 Move disk 3 from A to B
 Move disk 1 from C to A
 Move disk 2 from C to B
 Move disk 1 from A to B
 Move disk 4 from A to C
 Move disk 1 from B to C
 Move disk 2 from B to A
 Move disk 1 from C to A
 Move disk 3 from B to C
 Move disk 1 from A to B
 Move disk 2 from A to C
 Move disk 1 from B to C
 Time is 0.000000s

...Program finished with exit code 0
Press ENTER to exit console.
```

GCD

```
#include<stdio.h>

#include<time.h>

delay()

{

   int i;

   for(i=8000;i>0;i--);

}

int gcd(int a,int b){

if(b!=0)

   return gcd(b,a%b);

else return a;

}
```

```
void main(){

int m,n,ans;

time_t start,end;

start=time(NULL);

printf("enter two numbers");

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

//delay();

ans=gcd(m,n);

printf("%d",ans);

end=time(NULL);

printf("\n time taken:%f", difftime(end,start));

}
```

OUTPUT:



```
enter two numbers45 55
5
 time taken:3.000000

...Program finished with exit code 0
Press ENTER to exit console.
```

PROGRAM-2 Implement Recursive Binary search and Linear search and determine the time required to search an element. Repeat the experiment for different values of N and plot a graph of the time taken versus N.

LINEAR SEARCH:

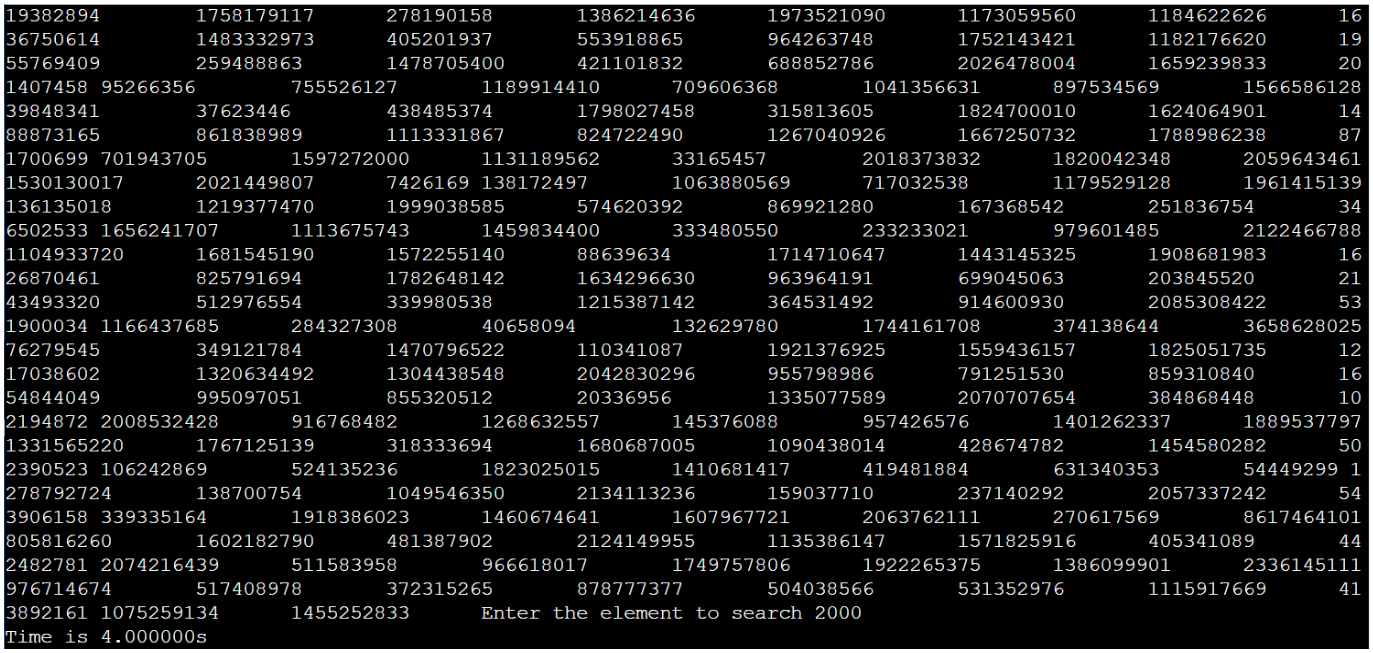
```
#include<stdio.h>
#include<conio.h>
#include<time.h>
void main()
{
  int n,flag,ele,i;
  int arr[1000];
  time_t start,end;
  printf("Enter the number of elements");
  scanf("%d",&n);
  for(i=0;i<n;i++)
  {
     arr[i]=rand();
  }
  printf("Enter the element to search");
  scanf("%d",&ele);
  start=time(NULL);
  for(i=0;i<n;i++)
  {
     if(arr[i]==ele)
     {
        printf("Element found in location %d",(i+1));
```

```
          flag=0;
       }
   }
   if(flag!=0)
   {
       printf("Element not found");
   }
   end=time(NULL);
   printf("Time is %fs",difftime(end,start));
}
```

BINARY SEARCH:

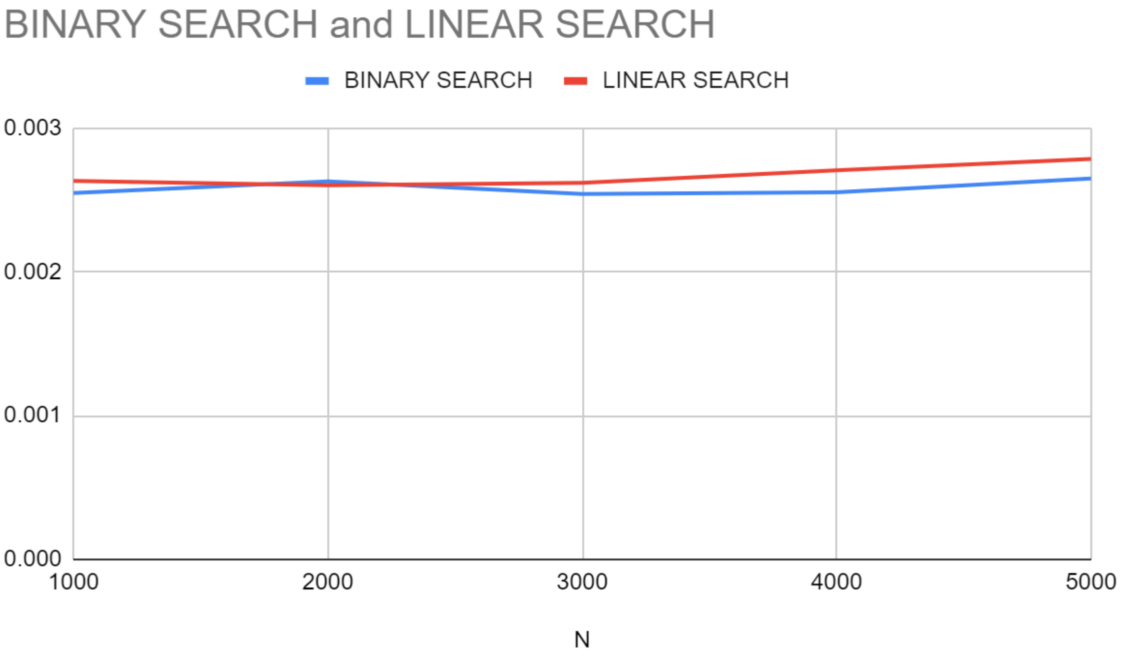
```
#include<stdio.h>
#include<conio.h>
#include<time.h>
void main()
{
   int n,flag,ele,i;
   int arr[1000];
   int beg=0,end=n-1,mid;
   time_t start,endt;
   printf("Enter the number of elements");
   scanf("%d",&n);
   for(i=0;i<n;i++)
   {
       arr[i]=rand();
   }
   printf("Enter element to search");
   scanf("%d",&ele);
   mid=(end-beg)/2;
   start=time(NULL);
   while(beg<=end)
   {
       if(arr[mid]<ele)
       {
           beg=mid+1;
       }
       else if(arr[mid]>ele)
       {
           end=mid-1;
       }
       else
       {
           printf("element found in %d",mid);
           flag=0;
       }
   }
   if(flag!=0)
   {
       printf("Element not found");
   }
   endt=time(NULL);
   printf("Time is %fs",difftime(endt,start));      }
```

OUTPUT:



```
19382894        1758179117      278190158       1386214636      1973521090      1173059560      1184622626      16
36750614        1483332973      405201937       553918865       964263748       1752143421      1182176620      19
55769409        259488863       1478705400      421101832       688852786       2026478004      1659239833      20
1407458 95266356        755526127       1189914410      709606368       1041356631      897534569       1566586128
39848341        37623446        438485374       1798027458      315813605       1824700010      1624064901      14
88873165        861838989       1113331867      824722490       1267040926      1667250732      1788986238      87
1700699 701943705       1597272000      1131189562      33165457        2018373832      1820042348      2059643461
1530130017      2021449807      7426169 138172497       1063880569      717032538       1179529128      1961415139
136135018       1219377470      1999038585      574620392       869921280       167368542       251836754       34
6502533 1656241707      1113675743      1459834400      333480550       233233021       979601485       2122466788
1104933720      1681545190      1572255140      88639634        1714710647      1443145325      1908681983      16
26870461        825791694       1782648142      1634296630      963964191       699045063       203845520       21
43493320        512976554       339980538       1215387142      364531492       914600930       2085308422      53
1900034 1166437685      284327308       40658094        132629780       1744161708      374138644       3658628025
76279545        349121784       1470796522      110341087       1921376925      1559436157      1825051735      12
17038602        1320634492      1304438548      2042830296      955798986       791251530       859310840       16
54844049        995097051       855320512       20336956        1335077589      2070707654      384868448       10
2194872 2008532428      916768482       1268632557      145376088       957426576       1401262337      1889537797
1331565220      1767125139      318333694       1680687005      1090438014      428674782       1454580282      50
2390523 106242869       524135236       1823025015      1410681417      419481884       631340353       54449299 1
278792724       138700754       1049546350      2134113236      159037710       237140292       2057337242      54
3906158 339335164       1918386023      1460674641      1607967721      2063762111      270617569       8617464101
805816260       1602182790      481387902       2124149955      1135386147      1571825916      405341089       44
2482781 2074216439      511583958       966618017       1749757806      1922265375      1386099901      2336145111
976714674       517408978       372315265       878777377       504038566       531352976       1115917669      41
3892161 1075259134      1455252833      Enter the element to search 2000
Time is 4.000000s
```

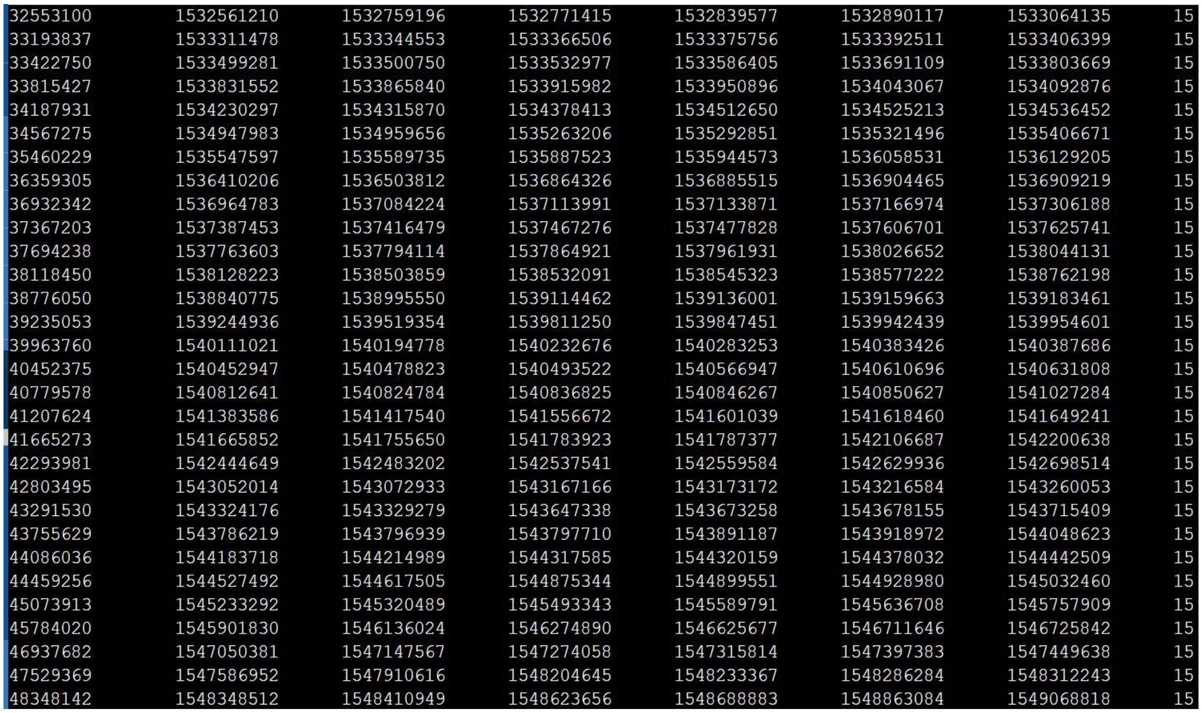
GRAPH:



PROGRAM 3 Sort a given set of N integer elements using Selection Sort technique and compute its time taken. Run the program for different values of N and record the time taken to sort.

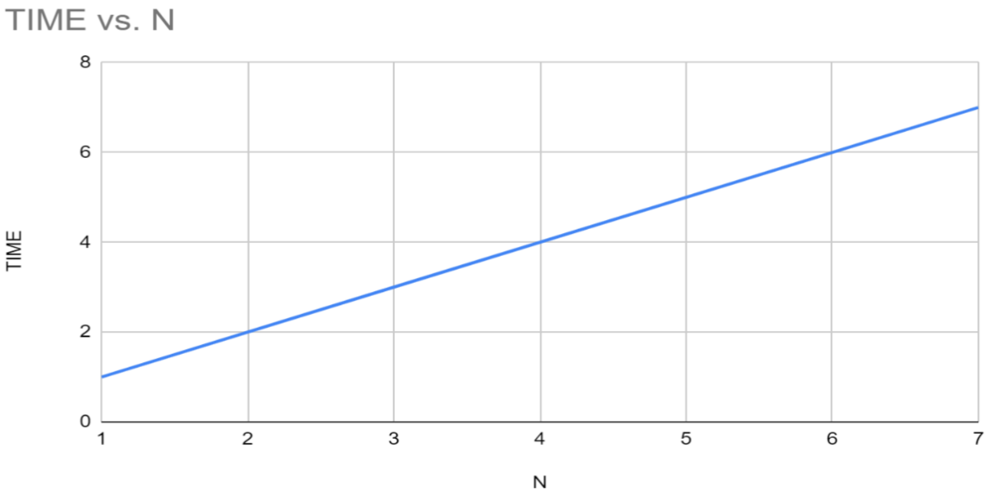
```
#include <stdio.h>

#include<time.h>

int main()

{

int a[100], n, i, j, position, swap;

time_t start,end;

printf("Enter number of elements");

scanf("%d", &n);

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

a[i]=rand(); }

start=time(NULL);

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

{

position=i;

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

{

if(a[position] > a[j])

position=j;

}

if(position != i)

{

swap=a[i];

a[i]=a[position];

a[position]=swap;    }     }
end=time(NULL);

printf("Sorted Array:");

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

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

printf("Time is %fs",difftime(end,start));

return 0;  }
```

OUTPUT



```
32553100     1532561210     1532759196     1532771415     1532839577     1532890117     1533064135     15
33193837     1533311478     1533344553     1533366506     1533375756     1533392511     1533406399     15
33422750     1533499281     1533500750     1533532977     1533586405     1533691109     1533803669     15
33815427     1533831552     1533865840     1533915982     1533950896     1534043067     1534092876     15
34187931     1534230297     1534315870     1534378413     1534512650     1534525213     1534536452     15
34567275     1534947983     1534959656     1535263206     1535292851     1535321496     1535406671     15
35460229     1535547597     1535589735     1535887523     1535944573     1536058531     1536129205     15
36359305     1536410206     1536503812     1536864326     1536885515     1536904465     1536909219     15
36932342     1536964783     1537084224     1537113991     1537133871     1537166974     1537306188     15
37367203     1537387453     1537416479     1537467276     1537477828     1537606701     1537625741     15
37694238     1537763603     1537794114     1537864921     1537961931     1538026652     1538044131     15
38118450     1538128223     1538503859     1538532091     1538545323     1538577222     1538762198     15
38776050     1538840775     1538995550     1539114462     1539136001     1539159663     1539183461     15
39235053     1539244936     1539519354     1539811250     1539847451     1539942439     1539954601     15
39963760     1540111021     1540194778     1540232676     1540283253     1540383426     1540387686     15
40452375     1540452947     1540478823     1540493522     1540566947     1540610696     1540631808     15
40779578     1540812641     1540824784     1540836825     1540846267     1540850627     1541027284     15
41207624     1541383586     1541417540     1541556672     1541601039     1541618460     1541649241     15
41665273     1541665852     1541755650     1541783923     1541787377     1542106687     1542200638     15
42293981     1542444649     1542483202     1542537541     1542559584     1542629936     1542698514     15
42803495     1543052014     1543072933     1543167166     1543173172     1543216584     1543260053     15
43291530     1543324176     1543329279     1543647338     1543673258     1543678155     1543715409     15
43755629     1543786219     1543796939     1543797710     1543891187     1543918972     1544048623     15
44086036     1544183718     1544214989     1544317585     1544320159     1544378032     1544442509     15
44459256     1544527492     1544617505     1544875344     1544899551     1544928980     1545032460     15
45073913     1545233292     1545320489     1545493343     1545589791     1545636708     1545757909     15
45784020     1545901830     1546136024     1546274890     1546625677     1546711646     1546725842     15
46937682     1547050381     1547147567     1547274058     1547315814     1547397383     1547449638     15
47529369     1547586952     1547910616     1548204645     1548233367     1548286284     1548312243     15
48348142     1548348512     1548410949     1548623656     1548688883     1548863084     1549068818     15
```

GRAPH



PROGRAM 4 : Write program to do the following:

a) Print all the nodes reachable from a given starting node in a digraph using BFS method.

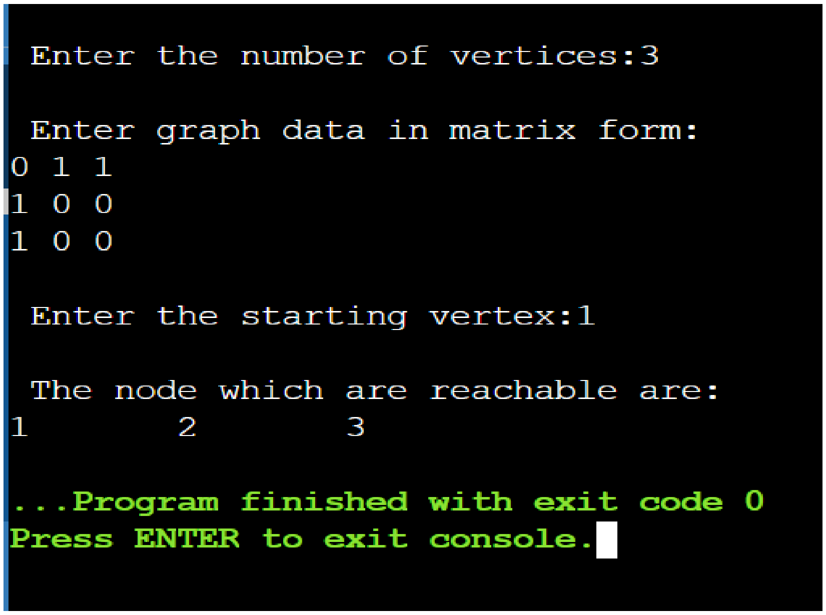
b) Check whether a given graph is connected or not using DFS method.

A:

```
#include<stdio.h>

#include<conio.h>

int a[20][20],q[20],visited[20],n,i,j,f=0,r=-1;

void bfs(int v)

{

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

if(a[v][i] && !visited[i])

q[++r]=i;

if(f<=r)

{

visited[q[f]]=1;

bfs(q[f++]);

}

}

void main()

{

int v;


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

scanf("%d",&n);

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

{
```

```
q[i]=0;
visited[i]=0;
}
printf("\n Enter graph data in matrix form:\n");
for(i=1;i<=n;i++)
for(j=1;j<=n;j++)
scanf("%d",&a[i][j]);
printf("\n Enter the starting vertex:");
scanf("%d",&v);
bfs(v);
printf("\n The node which are reachable are:\n");
for(i=1;i<=n;i++)
if(visited[i])
printf("%d\t",i);
getch();
}
```

OUTPUT

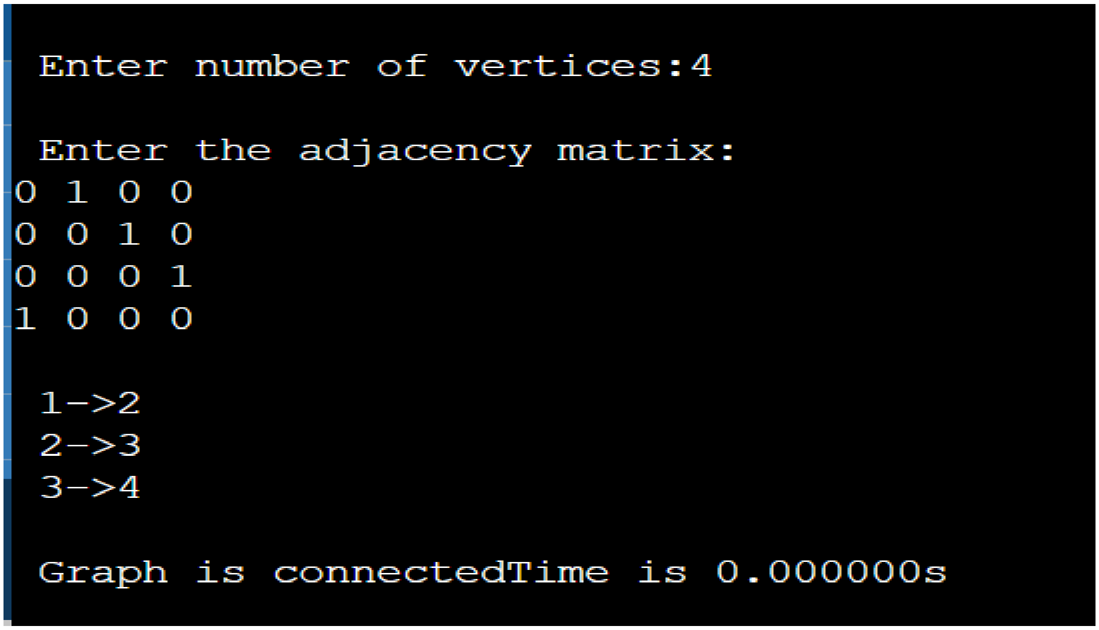


```
 Enter the number of vertices:3

 Enter graph data in matrix form:
0 1 1
1 0 0
1 0 0

 Enter the starting vertex:1

 The node which are reachable are:
1       2       3

...Program finished with exit code 0
Press ENTER to exit console.
```

B.

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
#include<time.h>
int a[20][20],reach[20],n;
time_t start,end;
void dfs(int v)   {
int i;
reach[v]=1;
for(i=1;i<=n;i++)
if(a[v][i] && !reach[i])   {
printf("\n %d->%d",v,i);
dfs(i);  }    }
void main()  {
int i,j,count=0;
printf("\n Enter number of vertices:");
scanf("%d",&n);
for(i=1;i<=n;i++)  {
reach[i]=0;
for(j=1;j<=n;j++)
a[i][j]=0;
}
printf("\n Enter the adjacency matrix:\n");
for(i=1;i<=n;i++)
for(j=1;j<=n;j++)
scanf("%d",&a[i][j]);
start=time(NULL);
dfs(1);
end=time(NULL);
printf("\n");
for(i=1;i<=n;i++)
{
if(reach[i])
count++;
}
if(count==n)
printf("\n Graph is connected");
else
printf("\n Graph is not connected");
printf("Time is %fs",difftime(end,start));
getch();
}
```

OUTPUT

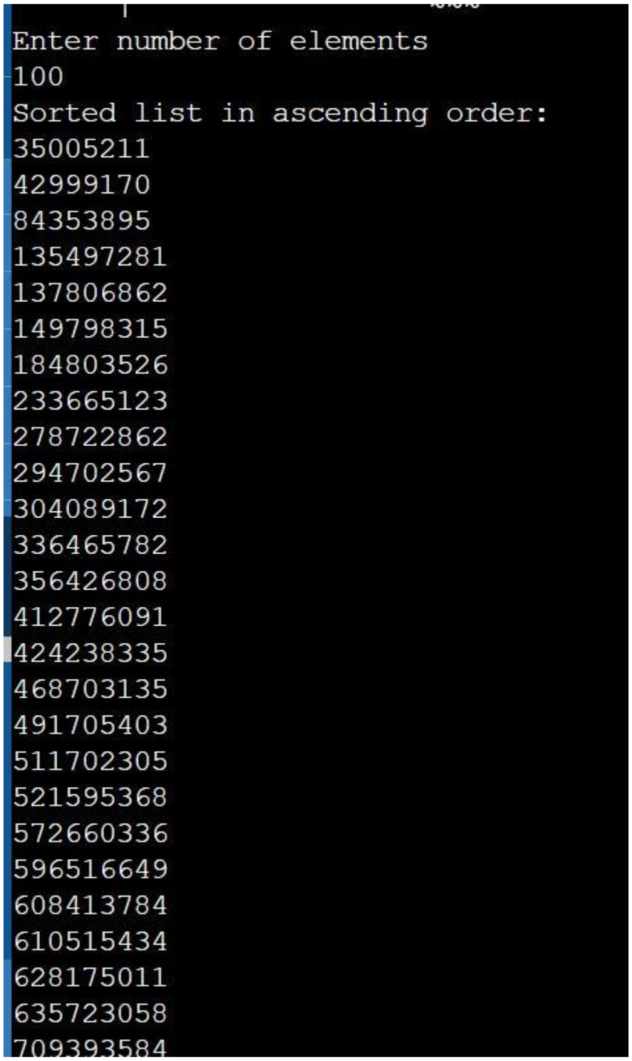


```
 Enter number of vertices:4

 Enter the adjacency matrix:
0 1 0 0
0 0 1 0
0 0 0 1
1 0 0 0

 1->2
 2->3
 3->4

 Graph is connectedTime is 0.000000s
```

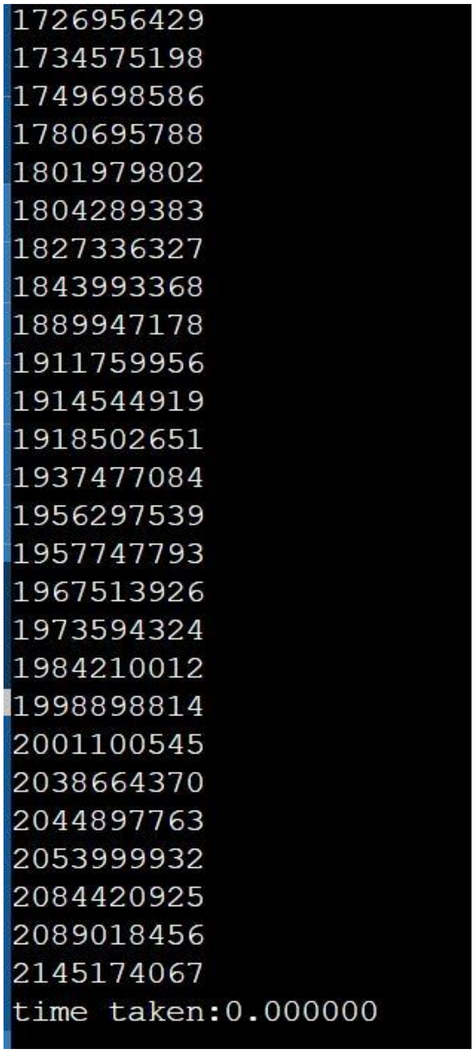
PROGRAM 5 Sort a given set of N integer elements using Insertion Sort technique and compute its time taken.

```
#include <stdio.h>

#include<time.h>

int main()

{

    int n, i, j, temp;

    int arr[64];

  time_t start,end;

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

    scanf("%d", &n);

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

        arr[i]=rand();   }

    start=time(NULL);

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

    {     j = i;

            while ( j > 0 && arr[j-1] > arr[j])

            {

                temp     = arr[j];

                arr[j]   = arr[j-1];

                arr[j-1] = temp;

                j--;

            }

    }

    end=time(NULL);

    printf("Sorted list in ascending order:\n");
    for (i = 0; i <= n - 1; i++)

    {

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

    }

    printf("time taken:%f",difftime(end,start));

    return 0;

}
```

OUTPUT

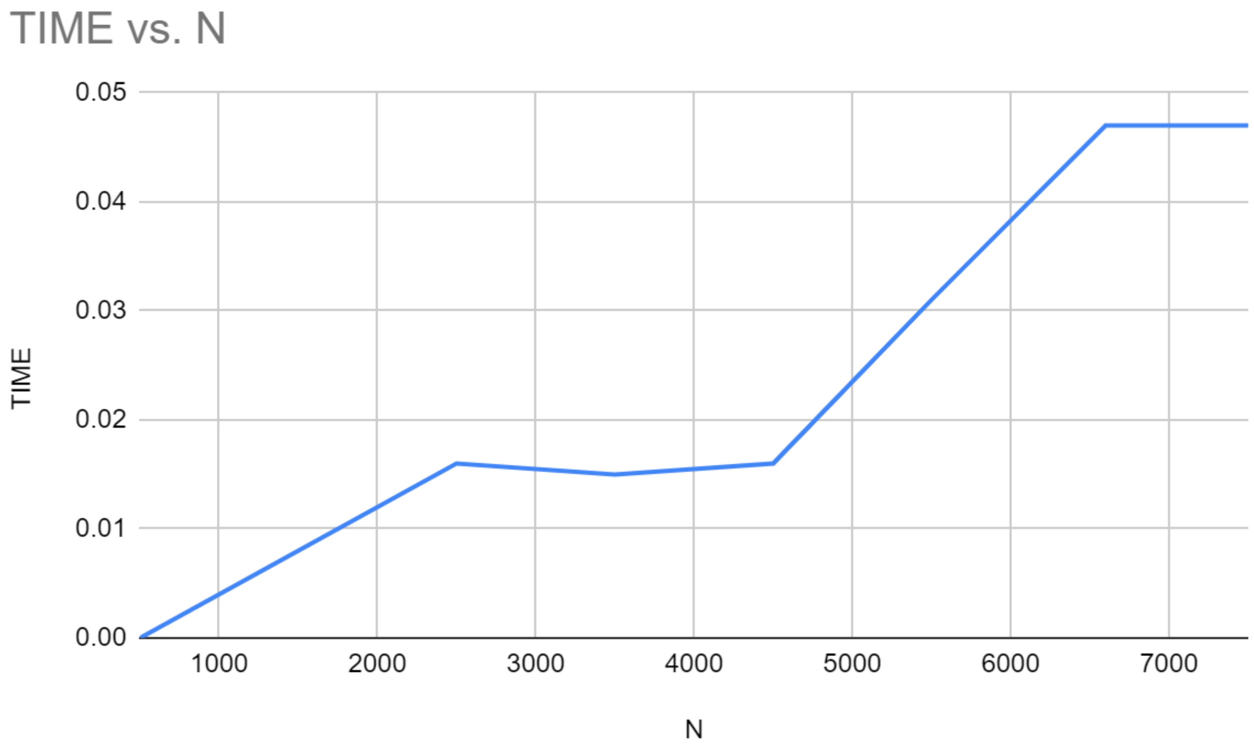


```
Enter number of elements
100
Sorted list in ascending order:
35005211
42999170
84353895
135497281
137806862
149798315
184803526
233665123
278722862
294702567
304089172
336465782
356426808
412776091
424238335
468703135
491705403
511702305
521595368
572660336
596516649
608413784
610515434
628175011
635723058
709393584
```



```
1726956429
1734575198
1749698586
1780695788
1801979802
1804289383
1827336327
1843993368
1889947178
1911759956
1914544919
1918502651
1937477084
1956297539
1957747793
1967513926
1973594324
1984210012
1998898814
2001100545
2038664370
2044897763
2053999932
2084420925
2089018456
2145174067
time taken:0.000000
```

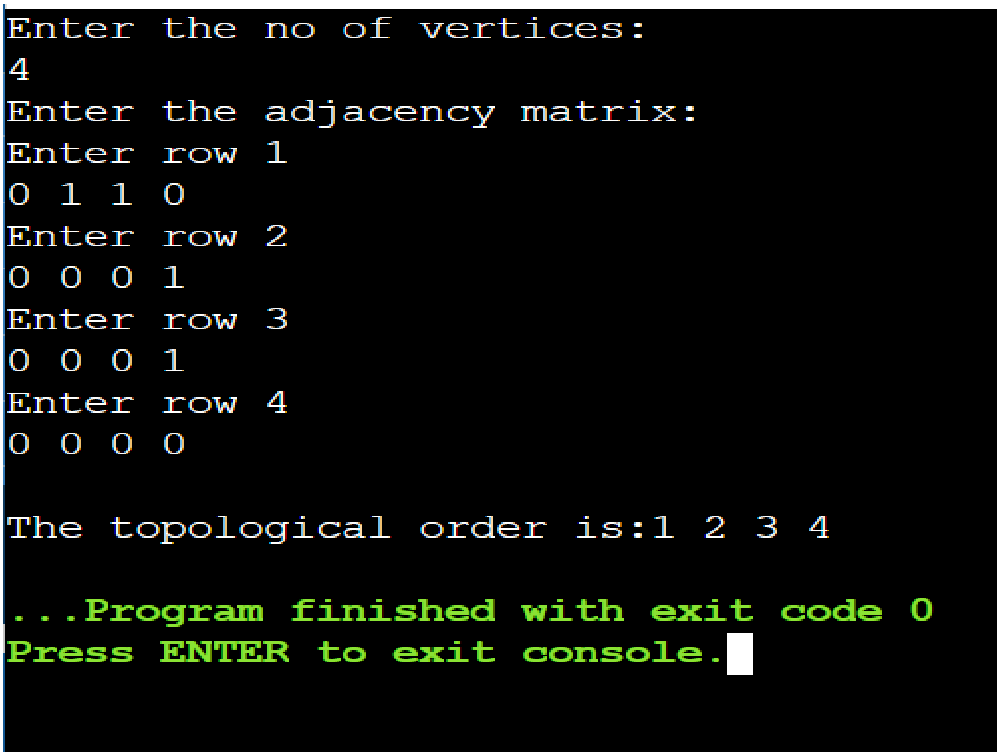
GRAPH


TIME vs. N
0.05
0.04
0.03
0.02
0.01
0.00
TIME
1000
2000
3000
4000
5000
6000
7000
N

PROGRAM 6 Write program to obtain the Topological ordering of vertices in a given digraph

```
#include<stdio.h>
#include<conio.h>
int main()
{
int i,j,k,n,a[10][10],indeg[10],flag[10],count=0;
printf("Enter the no of vertices:\n");
scanf("%d",&n);
printf("Enter the adjacency matrix:\n");
for(i=0;i<n;i++)
{
printf("Enter row %d\n",i+1);
for(j=0;j<n;j++)
scanf("%d",&a[i][j]);
}
for(i=0;i<n;i++)
{
        indeg[i]=0;
        flag[i]=0;   }
    for(i=0;i<n;i++)
        for(j=0;j<n;j++)
            indeg[i]=indeg[i]+a[j][i];
    printf("\nThe topological order is:");
    while(count<n)
{
        for(k=0;k<n;k++){
            if((indeg[k]==0) && (flag[k]==0)){
              printf("%d ",(k+1));
              flag [k]=1;
            }
for(i=0;i<n;i++)
{
              if(a[i][k]==1)
                indeg[k]--;
            }
 }
   count++;
  }
return 0;
}
```

OUTPUT

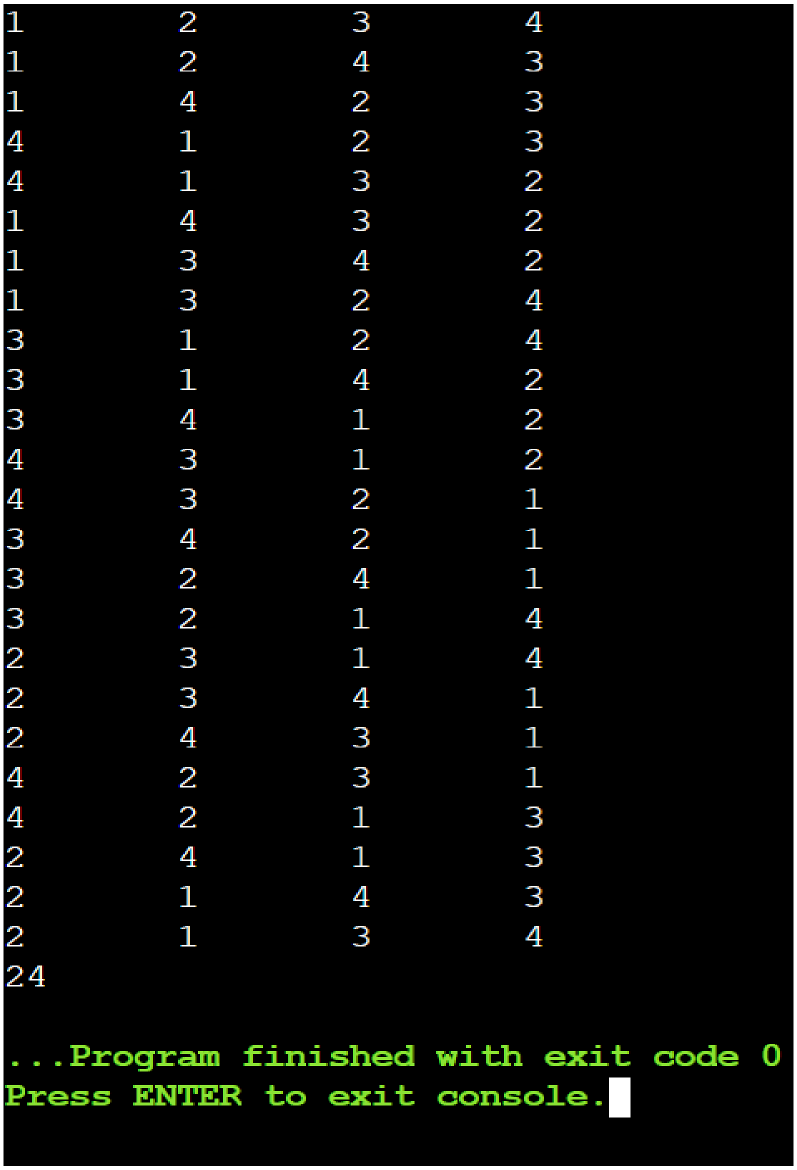


```
Enter the no of vertices:
4
Enter the adjacency matrix:
Enter row 1
0 1 1 0
Enter row 2
0 0 0 1
Enter row 3
0 0 0 1
Enter row 4
0 0 0 0

The topological order is:1 2 3 4

...Program finished with exit code 0
Press ENTER to exit console.
```

PROGRAM 7: Implement Johnson Trotter algorithm to generate permutations.

```
#include <stdio.h>
int fact(int n)  {
   int f=1;
   for(int i=1;i<=n;i++)  {
      f=f*i;   }
   return f; }
int search(int a[],int mobile,int n)  {
   for(int i=0;i<n;i++)   {
      if(a[i]==mobile)  {
         return i;   }  }
   return -1;   }
int getMobile(int a[],int dir[],int n)
{   int mobile=0;
   for(int i=0;i<n;i++)  {
      if(dir[a[i]-1]==0 && i!=0)  {
         if(a[i]>a[i-1] && a[i]>mobile)  {
            mobile=a[i];  }    }
      else if(dir[a[i]-1]==1 && i!=n-1)        {
         if(a[i]>a[i+1] && a[i]>mobile)  {
            mobile=a[i];     }       }      }
   return mobile;
}
void Permutations(int a[],int dir[],int n)  {
   int mobile=getMobile(a,dir,n);
   int pos=search(a,mobile,n);
   if(dir[a[pos]-1]==0 )  {
      int temp=a[pos];
      a[pos]=a[pos-1];
      a[pos-1]=temp;  }
   else if(dir[a[pos]-1]==1 )  {
        int temp=a[pos];
        a[pos]=a[pos+1];
        a[pos+1]=temp;  }
    for(int i=0;i<n;i++)  {
        if(a[i]>mobile) {
            if(dir[a[i]-1]==0) {
                dir[a[i]-1]=1; }
            else if(dir[a[i]-1]==1 ){
                dir[a[i]-1]=0; }   }
for(int i=0;i<n;i++)   {
        printf("%d\t",a[i]);    }
    printf("\n");  }
int main()
{
    int n=4;  int a[]={1,2,3,4};
    for(int i=0;i<n;i++)
    {
        printf("%d\t",a[i]);
    }
    printf("\n");

    int dir[]={0,0,0,0};

    int total=fact(n);
    int count=1;
    // printf("%d",total);
    for(int i=1;i<total;i++)
    {
        Permutations(a,dir,n);
        count++;
    }
    printf("%d",count);


    return 0;
}
```

OUTPUT

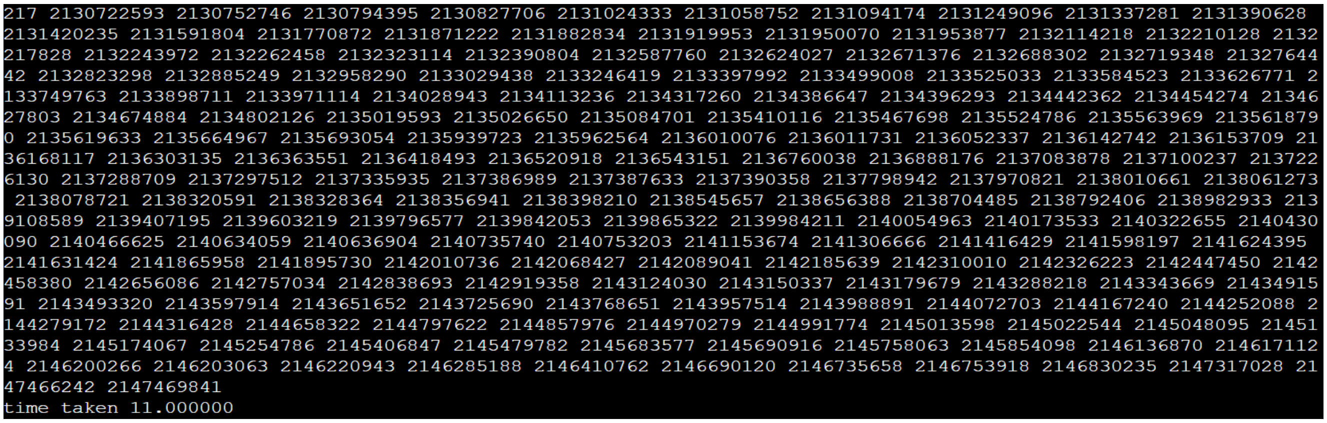


```
1       2       3       4
1       2       4       3
1       4       2       3
4       1       2       3
4       1       3       2
1       4       3       2
1       3       4       2
1       3       2       4
3       1       2       4
3       1       4       2
3       4       1       2
4       3       1       2
4       3       2       1
3       4       2       1
3       2       4       1
3       2       1       4
2       3       1       4
2       3       4       1
2       4       3       1
4       2       3       1
4       2       1       3
2       4       1       3
2       1       4       3
2       1       3       4
24

...Program finished with exit code 0
Press ENTER to exit console.
```

PROGRAM 8: Sort a given set of N integer elements using Merge Sort technique and compute its time taken. Run the program for different values of N and record the time taken to sort.

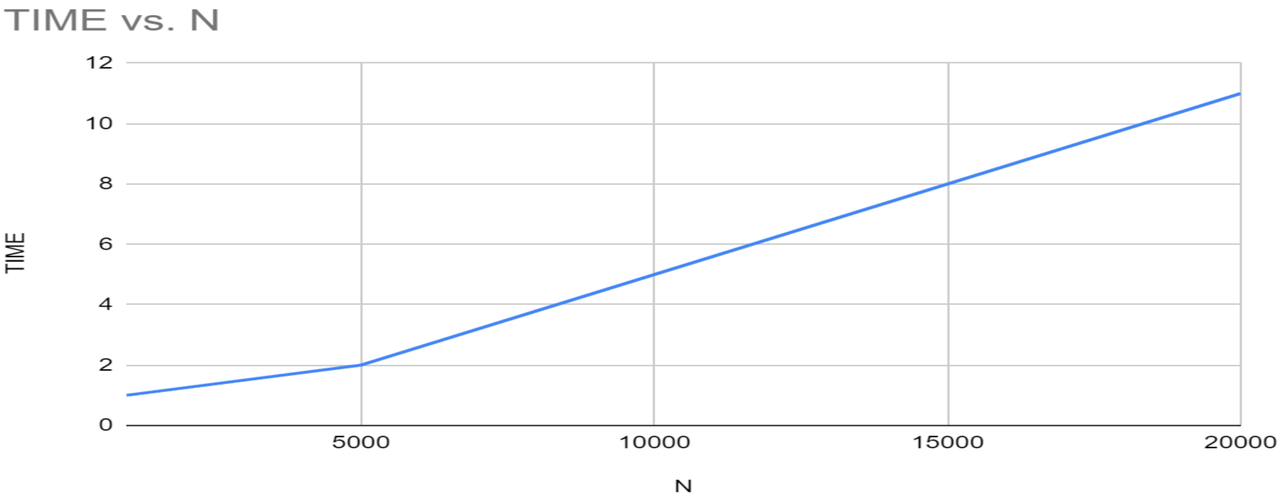
```
#include <stdio.h>
#include <stdlib.h>
 #include<time.h>
void merge(int arr[], int l, int m, int r)
{
    int i, j, k;
    int n1 = m - l + 1;
    int n2 = r - m;
    int L[n1], R[n2];
    for (i = 0; i < n1; i++)
        L[i] = arr[l + i];
    for (j = 0; j < n2; j++)
        R[j] = arr[m + 1 + j];
    i = 0;
    j = 0;
    k = l;
    while (i < n1 && j < n2) {
        if (L[i] <= R[j]) {
            arr[k] = L[i];
            i++;  }
        else {
            arr[k] = R[j];
            j++;  }
        k++;   }
    while (i < n1) {
        arr[k] = L[i];
        i++;
        k++;   }
    while (j < n2) {
        arr[k] = R[j];
        j++;
        k++;  }   }
void mergeSort(int arr[], int l, int r)
{
    if (l < r) {
int m = l + (r - l) / 2;
        mergeSort(arr, l, m);
        mergeSort(arr, m + 1, r);
  merge(arr, l, m, r);   }   }
int main()
{
    int i,n;
    int arr[1000];
      time_t start, end;
    printf("enter the number of elements");
    scanf("%d",&n);
    for(i=0;i<=n;i++)
    {
        arr[i]=rand();
    }
    start=time(NULL);
    int arr_size = sizeof(arr) / sizeof(arr[0]);
    printf("Given array is \n");
    for (i = 0; i < n ;i++)
        printf("%d ", arr[i]);
    printf("\n");
    mergeSort(arr, 0, arr_size - 1);
    for (i = 0; i < n; i++)
        printf("%d ", arr [i]);
    printf("\n");
    end=time(NULL);
    printf("time taken %f", difftime(end,start));
    return 0;
}
```

OUTPUT:



```
217 2130722593 2130752746 2130794395 2130827706 2131024333 2131058752 2131094174 2131249096 2131337281 2131390628
2131420235 2131591804 2131770872 2131871222 2131882834 2131919953 2131950070 2131953877 2132114218 2132210128 2132
217828 2132243972 2132262458 2132323114 2132390804 2132587760 2132624027 2132671376 2132688302 2132719348 21327644
42 2132823298 2132885249 2132958290 2133029438 2133246419 2133397992 2133499008 2133525033 2133584523 2133626771 2
133749763 2133898711 2133971114 2134028943 2134113236 2134317260 2134386647 2134396293 2134442362 2134454274 21346
27803 2134674884 2134802126 2135019593 2135026650 2135084701 2135410116 2135467698 2135524786 2135563969 213561879
0 2135619633 2135664967 2135693054 2135939723 2135962564 2136010076 2136011731 2136052337 2136142742 2136153709 21
36168117 2136303135 2136363551 2136418493 2136520918 2136543151 2136760038 2136888176 2137083878 2137100237 213722
6130 2137288709 2137297512 2137335935 2137386989 2137387633 2137390358 2137798942 2137970821 2138010661 2138061273
 2138078721 2138320591 2138328364 2138356941 2138398210 2138545657 2138656388 2138704485 2138792406 2138982933 213
9108589 2139407195 2139603219 2139796577 2139842053 2139865322 2139984211 2140054963 2140173533 2140322655 2140430
090 2140466625 2140634059 2140636904 2140735740 2140753203 2141153674 2141306666 2141416429 2141598197 2141624395
2141631424 2141865958 2141895730 2142010736 2142068427 2142089041 2142185639 2142310010 2142326223 2142447450 2142
458380 2142656086 2142757034 2142838693 2142919358 2143124030 2143150337 2143179679 2143288218 2143343669 21434915
91 2143493320 2143597914 2143651652 2143725690 2143768651 2143957514 2143988891 2144072703 2144167240 2144252088 2
144279172 2144316428 2144658322 2144797622 2144857976 2144970279 2144991774 2145013598 2145022544 2145048095 21451
33984 2145174067 2145254786 2145406847 2145479782 2145683577 2145690916 2145758063 2145854098 2146136870 214617112
4 2146200266 2146203063 2146220943 2146285188 2146410762 2146690120 2146735658 2146753918 2146830235 2147317028 21
47466242 2147469841
time taken 11.000000
```

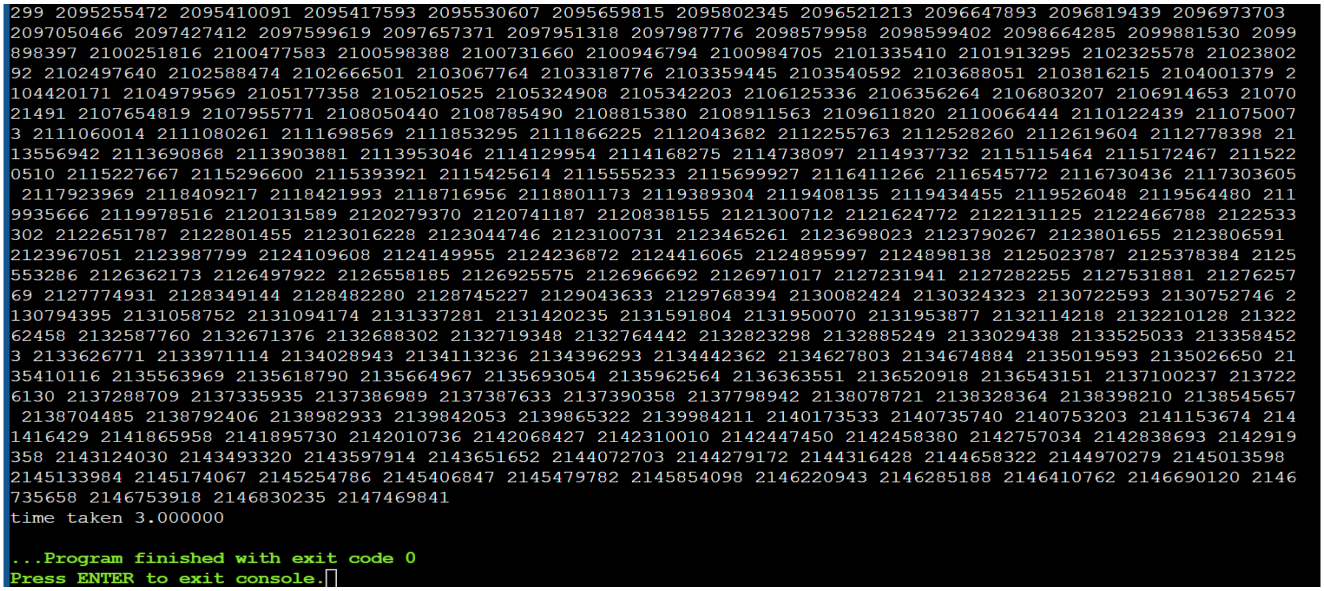
GRAPH:



PROGRAM 9: Sort a given set of N integer elements using Quick Sort technique and compute its time taken.

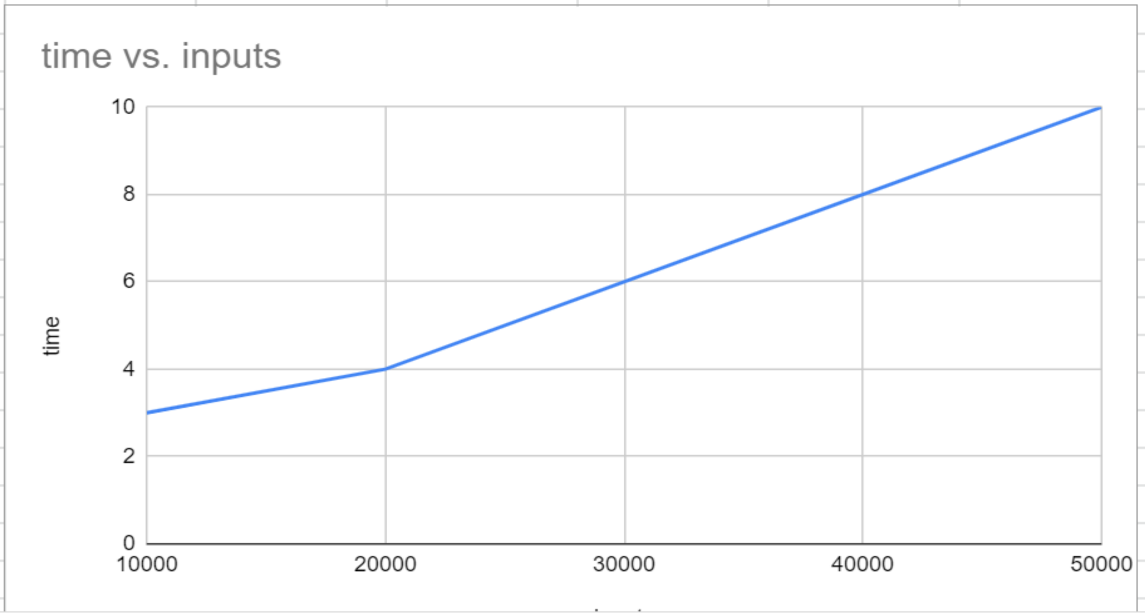
```
#include<stdio.h>
#include <stdlib.h>
 #include<time.h>
void quicksort(int number[25],int first,int last){
  int i, j, pivot, temp;
  if(first<last){
    pivot=first;
    i=first;
    j=last;
    while(i<j){
      while(number[i]<=number[pivot]&&i<last)
      i++;
      while(number[j]>number[pivot])
      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(){
   time_t start, end;
  int i, count, number[10000];
  printf("How many elements are u going to enter?: ");

  scanf("%d",&count);
  for(i=0;i<count;i++)
  {
     number[i]=rand();
  }
  start=time(NULL);
  quicksort(number,0,count-1);
  end=time(NULL);
  printf("Order of Sorted elements: ");
  for(i=0;i<count;i++)
  printf(" %d",number[i]);
  printf("\ntime taken %f", difftime(end,start));
  return 0;
}
```

OUTPUT:



```
299 2095255472 2095410091 2095417593 2095530607 2095659815 2095802345 2096521213 2096647893 2096819439 2096973703
2097050466 2097427412 2097599619 2097657371 2097951318 2097987776 2098579958 2098599402 2098664285 2099881530 2099
898397 2100251816 2100477583 2100598388 2100731660 2100946794 2100984705 2101335410 2101913295 2102325578 21023802
92 2102497640 2102588474 2102666501 2103067764 2103318776 2103359445 2103540592 2103688051 2103816215 2104001379 2
104420171 2104979569 2105177358 2105210525 2105324908 2105342203 2106125336 2106356264 2106803207 2106914653 21070
21491 2107654819 2107955771 2108050440 2108785490 2108815380 2108911563 2109611820 2110066444 2110122439 211075007
3 2111060014 2111080261 2111698569 2111853295 2111866225 2112043682 2112255763 2112528260 2112619604 2112778398 21
13556942 2113690868 2113903881 2113953046 2114129954 2114168275 2114738097 2114937732 2115115464 2115172467 211522
0510 2115227667 2115296600 2115393921 2115425614 2115555233 2115699927 2116411266 2116545772 2116730436 2117303605
 2117923969 2118409217 2118421993 2118716956 2118801173 2119389304 2119408135 2119434455 2119526048 2119564480 211
9935666 2119978516 2120131589 2120279370 2120741187 2120838155 2121300712 2121624772 2122131125 2122466788 2122533
302 2122651787 2122801455 2123016228 2123044746 2123100731 2123465261 2123698023 2123790267 2123801655 2123806591
2123967051 2123987799 2124109608 2124149955 2124236872 2124416065 2124895997 2124898138 2125023787 2125378384 2125
553286 2126362173 2126497922 2126558185 2126925575 2126966692 2126971017 2127231941 2127282255 2127531881 21276257
69 2127774931 2128349144 2128482280 2128745227 2129043633 2129768394 2130082424 2130324323 2130722593 2130752746 2
130794395 2131058752 2131094174 2131337281 2131420235 2131591804 2131950070 2131953877 2132114218 2132210128 21322
62458 2132587760 2132671376 2132688302 2132719348 2132764442 2132823298 2132885249 2133029438 2133525033 213358452
3 2133626771 2133971114 2134028943 2134113236 2134396293 2134442362 2134627803 2134674884 2135019593 2135026650 21
35410116 2135563969 2135618790 2135664967 2135693054 2135962564 2136363551 2136520918 2136543151 2137100237 213722
6130 2137288709 2137335935 2137386989 2137387633 2137390358 2137798942 2138078721 2138328364 2138398210 2138545657
 2138704485 2138792406 2138982933 2139842053 2139865322 2139984211 2140173533 2140735740 2140753203 2141153674 214
1416429 2141865958 2141895730 2142010736 2142068427 2142310010 2142447450 2142458380 2142757034 2142838693 2142919
358 2143124030 2143493320 2143597914 2143651652 2144072703 2144279172 2144316428 2144658322 2144970279 2145013598
2145133984 2145174067 2145254786 2145406847 2145479782 2145854098 2146220943 2146285188 2146410762 2146690120 2146
735658 2146753918 2146830235 2147469841
time taken 3.000000

...Program finished with exit code 0
Press ENTER to exit console.
```

GRAPH:

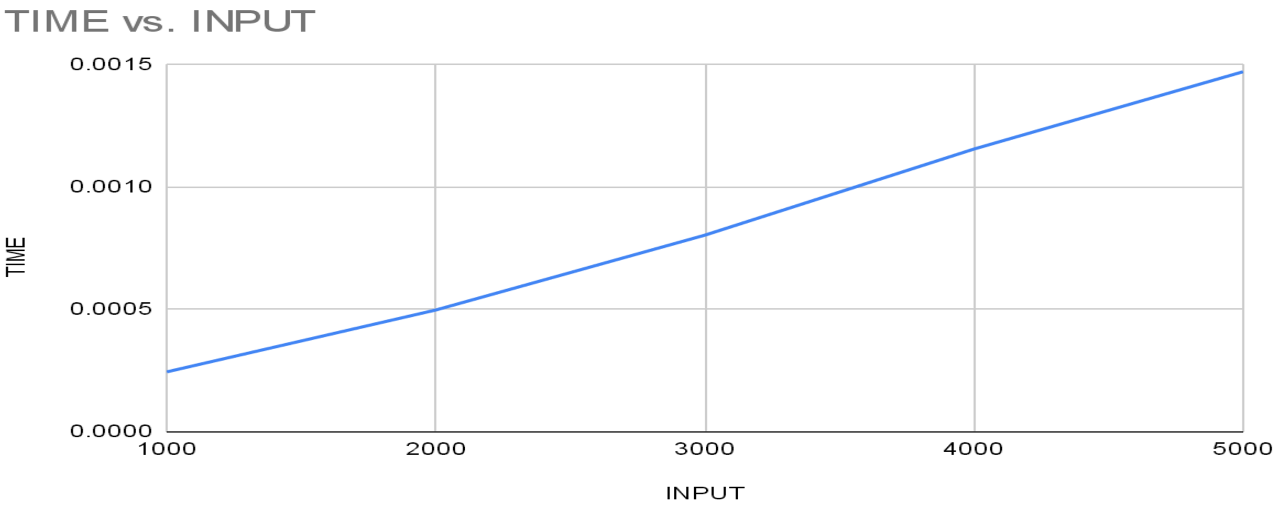


PROGRAM 10: Sort a given set of N integer elements using Heap Sort technique and compute its time taken.

```
#include <stdio.h>
#include<conio.h>
void heapify(int a[], int n, int i)  {
    int largest = i;
    int left = 2 * i + 1;
    int right = 2 * i + 2;
    if (left < n && a[left] > a[largest])
        largest = left;
    if (right < n && a[right] > a[largest])
        largest = right;
    if (largest != i) {
        int temp = a[i];
        a[i] = a[largest];
        a[largest] = temp;
        heapify(a, n, largest);    }        }
void heapSort(int a[], int n)
{       int i, temp;
    for ( i = n / 2 - 1; i >= 0; i--)
          heapify(a, n, i);
    for ( i = n - 1; i >= 0; i--) {
          temp = a[0];
          a[0] = a[i];
          a[i] = temp;
          heapify(a, i, 0);     }      }
void printArr(int arr[], int n)    {       int i;
    for ( i = 0; i < n; ++i)  {
          printf("%d", arr[i]);
          printf(" "); }    }
```

```
int main()
{
int i,n;
  int  a[100];
  clrscr();
    printf("enter the no.of elements");
    scanf("%d",&n);
    for(i=0;i<n;i++)
{
 a[i]=rand()%10;
}
   printf("Before sorting array elements are - \n");
   printArr(a, n);
   heapSort(a, n);
   printf("\nAfter sorting array elements are - \n");
   printArr(a, n);
   getch();
   return 0;
}
```

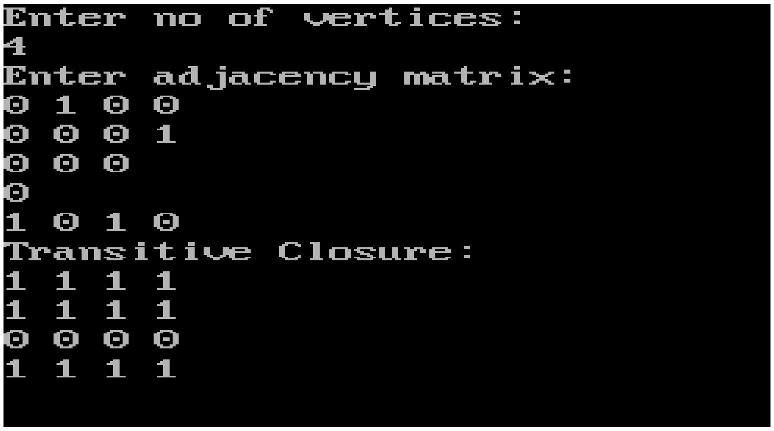
GRAPH:



PROGRAM 11: Implement Warshall’s algorithm using dynamic programming

```
#include<stdio.h>

int a[30][30];

void warshall(int n){

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

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

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

                a[i][j]=a[i][j]|| (a[i][k] && a[k][j]);     }

int main(){

    int n;

    printf("Enter no of vertices: \n");

    scanf("%d",&n);

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

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

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

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

    warshall(n);

    printf("Transitive Closure: \n");

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

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

        printf("\n");         }          }
```

OUTPUT:

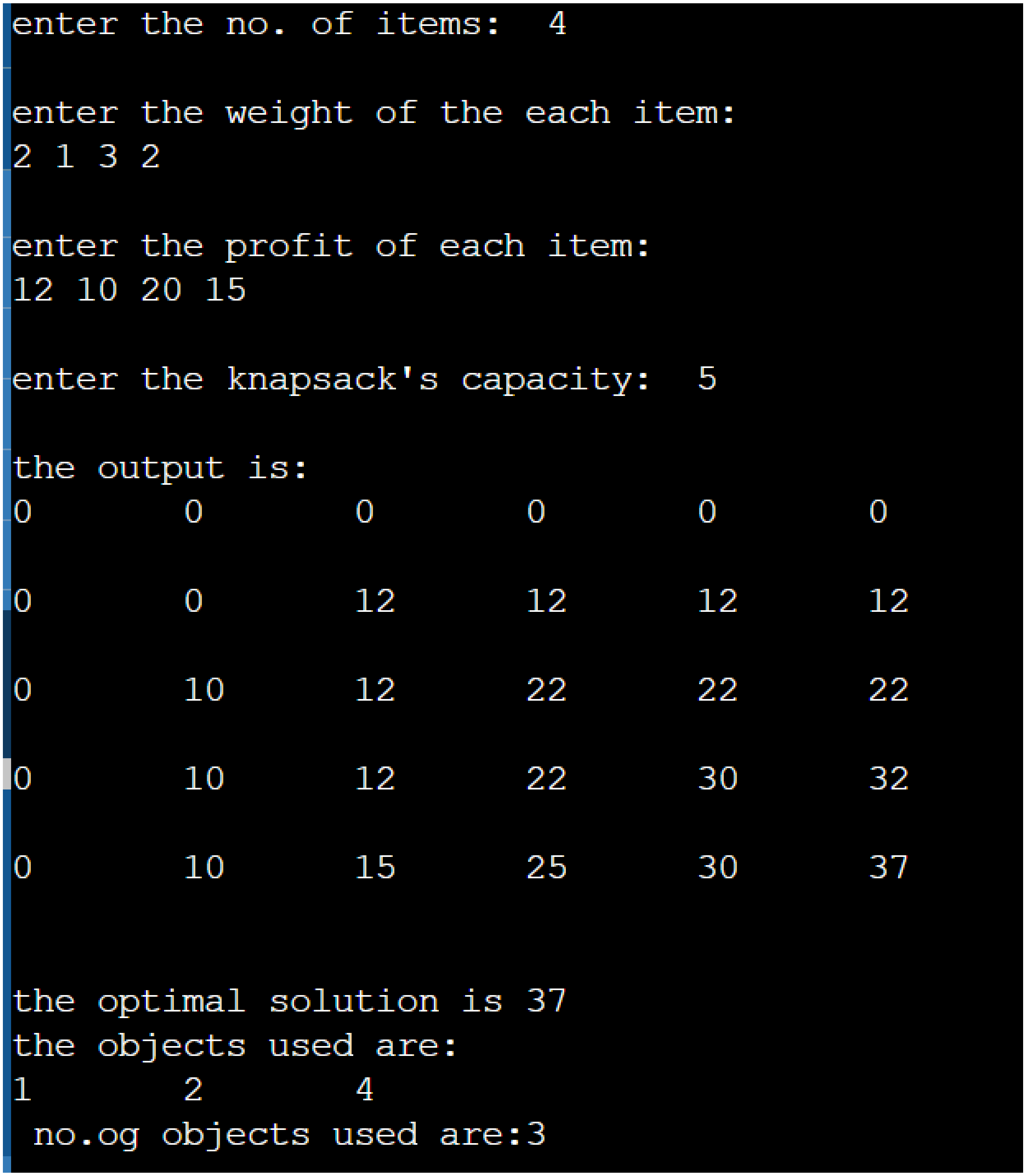


```
Enter no of vertices:
4
Enter adjacency matrix:
0 1 0 0
0 0 0 1
0 0 0
0
1 0 1 0
Transitive Closure:
1 1 1 1
1 1 1 1
0 0 0 0
1 1 1 1
```

PROGRAM 12: Implement 0/1 Knapsack problem using dynamic programming

```
#include<stdio.h>
void knapsack();
int max(int,int);
int i,j,n,m,p[10],w[10],v[10][10];
void main()
{
printf("\nenter the no. of items:\t");
 scanf("%d",&n);
 printf("\nenter 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",&m);
 knapsack();   }
void knapsack()
  {
    int count=0;
 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("\nthe 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("\nthe optimal solution is %d",v[n][m]);
 printf("\nthe objects used are:\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++)  {
 if(x[i]==1)  {
count++;
  printf("%d\t",i);  } }
 printf("\n no.og objects used are:%d", count);    }
int max(int x,int y)    {
if(x>y)    {
  return x;   }
else   {
  return y;   }
}
```

OUTPUT:

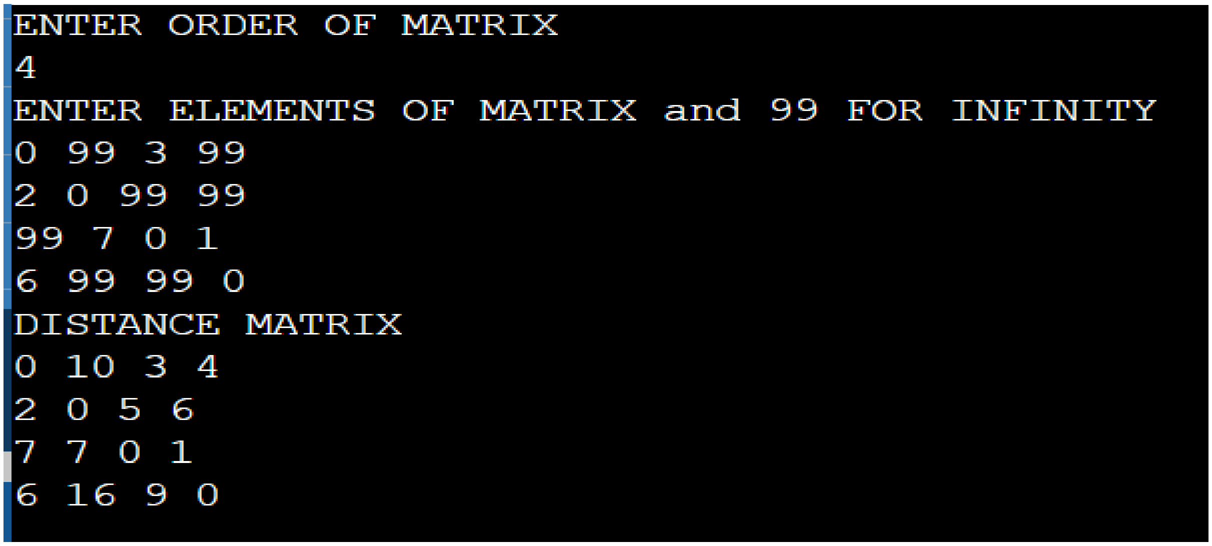


```
enter the no. of items:   4

enter the weight of the each item:
2 1 3 2

enter the profit of each item:
12 10 20 15

enter the knapsack's capacity:   5

the output is:
0       0       0       0       0       0

0       0       12      12      12      12

0       10      12      22      22      22

0       10      12      22      30      32

0       10      15      25      30      37


the optimal solution is 37
the objects used are:
1       2       4
 no.og objects used are:3
```

PROGRAM 13: Implement All Pair Shortest paths problem using Floyd’s algorithm.

```
#include<stdio.h>
int n;
void display(int dist[][n]);
void floyd (int graph[][n])
{
        int dist[n][n], i, j, k;
        for (i = 0; i < n; i++)
                for (j = 0; j < n; j++)
                        dist[i][j] = graph[i][j];
        for (k = 0; k < n; k++)
        {
                for (i = 0; i < n; i++)
                {
                        for (j = 0; j < n; j++)
                        {
                        if (dist[i][k] + dist[k][j] < dist[i][j])
                        dist[i][j] = dist[i][k] + dist[k][j];    }    }    }
        display(dist);
}
void display(int dist[][n])
{           printf ("DISTANCE MATRIX \n");
        for (int i = 0; i < n; i++)
        {       for (int j = 0; j < n; j++)
                {      if (dist[i][j] == 99)
                                printf("99 ");
                        else
                                printf ("%d ", dist[i][j]);       }
                printf("\n");       }       }
int main()
{
   printf("ENTER ORDER OF MATRIX \n");
   scanf("%d",&n);
   int graph[n][n];
   printf("ENTER ELEMENTS OF MATRIX and 99 FOR INFINITY\n");
   for(int i = 0;i < n;i++)
   {
      for(int j = 0;j < n; j++)
      {
         scanf("%d",&graph[i][j]);       }    }
       floyd(graph);
       return 0;
}
```

OUTPUT:

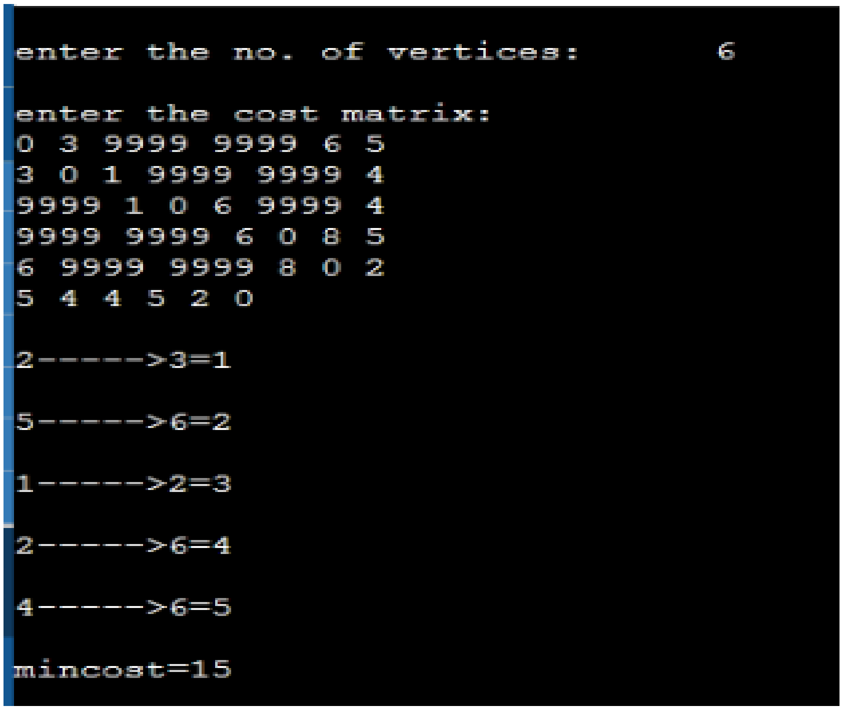


```
ENTER ORDER OF MATRIX
4
ENTER ELEMENTS OF MATRIX and 99 FOR INFINITY
0 99 3 99
2 0 99 99
99 7 0 1
6 99 99 0
DISTANCE MATRIX
0 10 3 4
2 0 5 6
7 7 0 1
6 16 9 0
```

PROGRAM 14: Find Minimum Cost Spanning Tree of a given undirected graph using Prim’s algorithm.

```
#include<stdio.h>
#include<conio.h>
void prims();
int c[10][10],n;
void main()  {
 int i,j;
clrscr();
 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]); } }
prims();
getch(); }
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("\nmincost=%d",mincost);   }
```

OUTPUT:

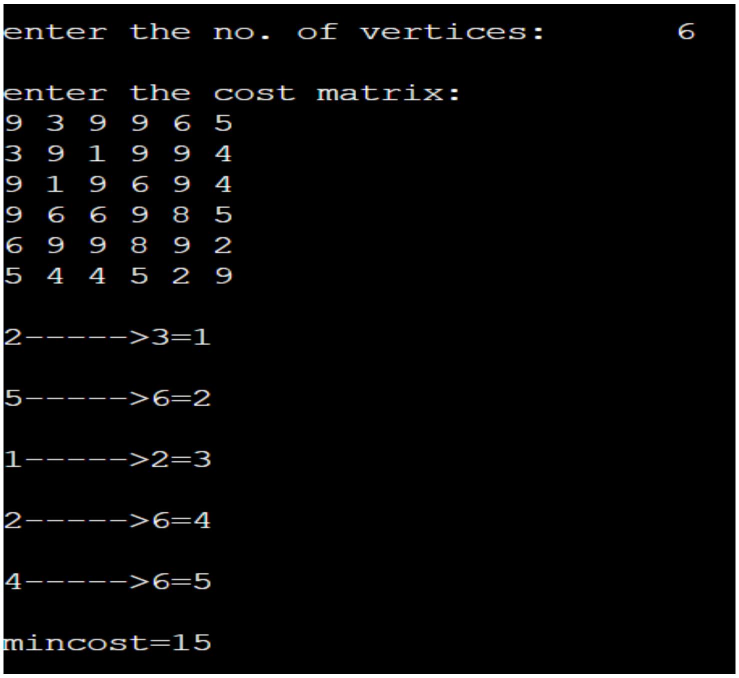


```
enter the no. of vertices:      6

enter the cost matrix:
0 3 9999 9999 6 5
3 0 1 9999 9999 4
9999 1 0 6 9999 4
9999 9999 6 0 8 5
6 9999 9999 8 0 2
5 4 4 5 2 0

2----->3=1

5----->6=2

1----->2=3

2----->6=4

4----->6=5

mincost=15
```

PROGRAM 15: Find Minimum Cost Spanning Tree of a given undirected graph using Kruskals algorithm.

```
#include<stdio.h>
#include<conio.h>
void kruskals();
int c[10][10],n;
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]); } }
kruskals();
 getch();  }
void kruskals()    {
 int i,j,u,v,a,b,min;
 int ne=0,mincost=0;
 int parent[10];
  parent[i]=0;   }
 while(ne!=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;
   ne=ne+1;
   mincost=mincost+min;    }
   c[a][b]=c[b][a]=9999;  }
printf("\nmincost=%d",mincost);   }
```

OUTPUT:



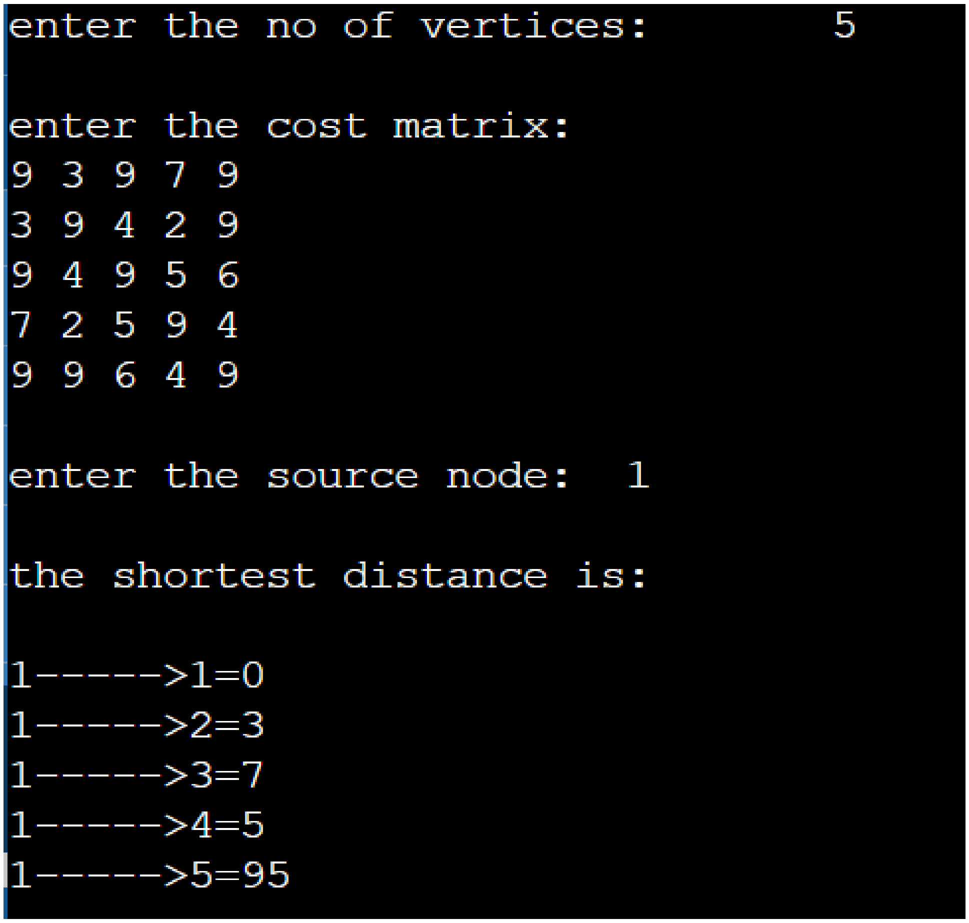
```
enter the no. of vertices:      6

enter the cost matrix:
9 3 9 9 6 5
3 9 1 9 9 4
9 1 9 6 9 4
9 6 6 9 8 5
6 9 9 8 9 2
5 4 4 5 2 9

2----->3=1

5----->6=2

1----->2=3

2----->6=4

4----->6=5

mincost=15
```

PROGRAM 16: From a given vertex in a weighted connected graph, find shortest paths to other vertices using Dijkstra’s 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();
 getch();   }
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;
```

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

OUTPUT:

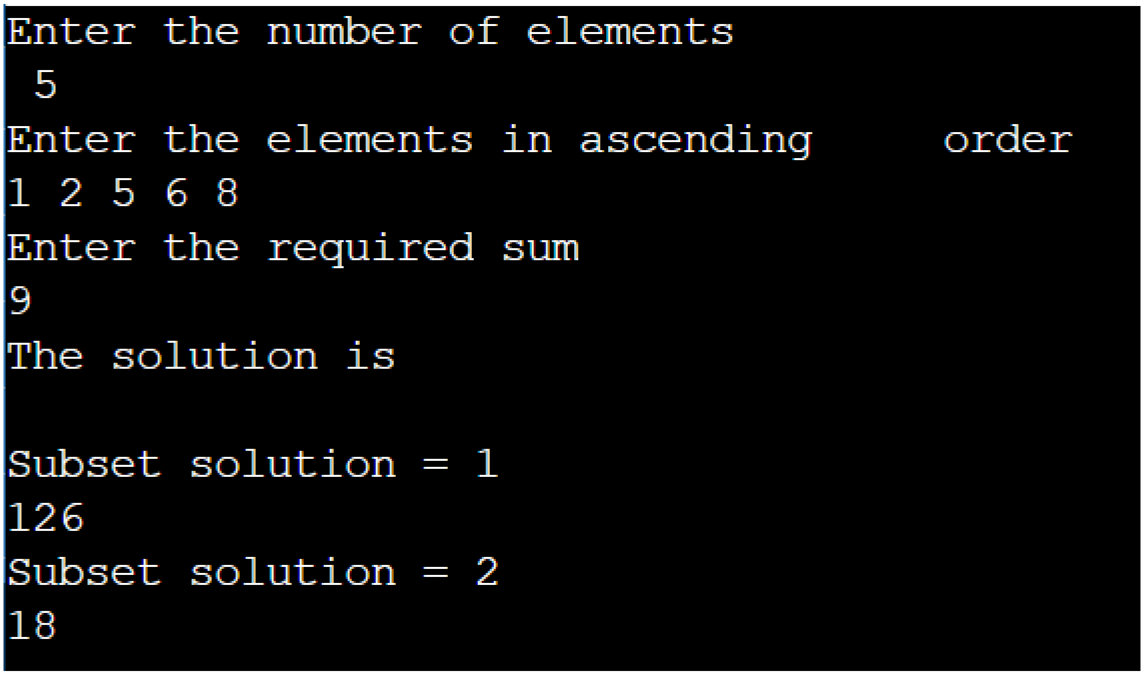


```
enter the no of vertices:      5

enter the cost matrix:
9 3 9 7 9
3 9 4 2 9
9 4 9 5 6
7 2 5 9 4
9 9 6 4 9

enter the source node:   1

the shortest distance is:

1----->1=0
1----->2=3
1----->3=7
1----->4=5
1----->5=95
```

PROGRAM 17: Implement “Sum of Subsets” using Backtracking. “Sum of Subsets” problem: Find a subset of a given set S = {s1,s2,……,sn} of n positive integers whose sum is equal to a given positive integer d. For example, if S = {1,2,5,6,8} and d = 9 there are two solutions {1,2,6} and {1,8}. A suitable message is to be displayed if the given problem instance doesn’t have a solution. #include<stdio.h>

```
#include<conio.h>
intcount,w[10],d,x[10];
void subset(intcs, int k, int r)
{
int i;
x[k]=1;
if(cs+w[k]==d)
{
printf("\nSubset solution = %d\n", ++count);
for(i=0;i<=k;i++)
{
if(x[i]==1)
printf("%d", 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;
printf("Enter the number 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");
scanf("%d", &d);
for(i=0;i<n;i++)
sum+=w[i];
if(sum<d)
{
printf("No solution exists\n");
return;
}
printf("The solution is\n");
count=0;
subset(0,0,sum);
getch();
}
```

OUTPUT:



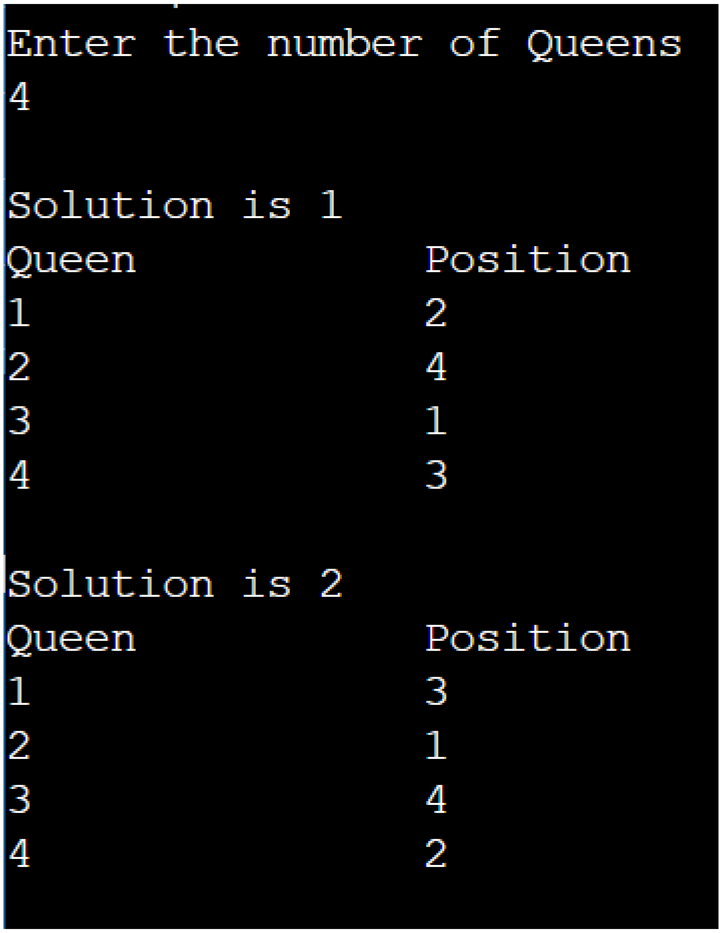
```
Enter the number of elements
 5
Enter the elements in ascending     order
1 2 5 6 8
Enter the required sum
9
The solution is

Subset solution = 1
126
Subset solution = 2
18
```

PROGRAM 18: Implement “N-Queens Problem” using Backtracking.

```
#include<stdio.h>
#include<conio.h>
void nqueens(int n)
{
        Int k,x[20],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 is %d\n", ++count);
                                printf("Queen\t\tPosition\n");
                                for(k=1;k<=n;k++)
                                        printf("%d\t\t%d\n", k,x[k]);
                        }
                        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 0;
        }
        return 1;
}
```

```
void main()
{
        int n;
        clrscr();
        printf("Enter the number of Queens\n");
        scanf("%d", &n);
        nqueens(n);
        getch();
}
```

OUTPUT:



```
Enter the number of Queens
4

Solution is 1
Queen           Position
1               2
2               4
3               1
4               3

Solution is 2
Queen           Position
1               3
2               1
3               4
4               2
```