

# VISVESVARAYA TECHNOLOGICAL UNIVERSITY

“JnanaSangama”, Belgaum -590014, Karnataka.



## LAB REPORT on

## Analysis and Design of Algorithms

*Submitted by*

**LOKESH R (1BM20CS078)**

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



**B.M.S. COLLEGE OF ENGINEERING**  
(Autonomous Institution under VTU)  
**BENGALURU-560019**  
**May-2022 to July-2022**

**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 **LOKESH R(1BM20CS078)**, who is bonafide 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.

**Dr.Rajeshwari BS**  
Assistant Professor  
Department of CSE  
BMSCE, Bengaluru

**Dr. Jyothi S Nayak**  
Professor and Head  
Department of CSE  
BMSCE, Bengaluru

## Index Sheet

Sl. No.	Experiment Title	Page No.
<b>1</b>	Write a recursive program to Solve <b>a)</b> Towers-of-Hanoi problem <b>b)</b> To find GCD	<b>5-7</b>
<b>2</b>	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.	<b>8-13</b>
<b>3</b>	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.	<b>14-18</b>
<b>4</b>	Write program to do the following: <b>a)</b> Print all the nodes reachable from a given starting node in a digraph using BFS method. <b>b)</b> Check whether a given graph is connected or not using DFS method.	<b>19-24</b>
<b>5</b>	Sort a given set of N integer elements using Insertion Sort technique and compute its time taken.	<b>25-27</b>
<b>6</b>	Write program to obtain the Topological ordering of vertices in a given digraph.	<b>28-30</b>
<b>7</b>	Implement Johnson Trotter algorithm to generate permutations.	<b>31-37</b>
<b>8</b>	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.	<b>38-41</b>
<b>9</b>	Sort a given set of N integer elements using Quick Sort technique and compute its time taken.	<b>42-44</b>
<b>10</b>	Sort a given set of N integer elements using Heap Sort technique and compute its time taken.	<b>45-50</b>
<b>11</b>	Implement Warshall's algorithm using dynamic programming	<b>51-52</b>
<b>12</b>	Implement 0/1 Knapsack problem using dynamic programming.	<b>53-57</b>
<b>13</b>	Implement All Pair Shortest paths problem using Floyd's algorithm.	<b>58-60</b>
<b>14</b>	Find Minimum Cost Spanning Tree of a given undirected graph using Prim's algorithm.	<b>61-63</b>
<b>15</b>	Find Minimum Cost Spanning Tree of a given undirected graph using Kruskals algorithm.	<b>64-67</b>
<b>16</b>	From a given vertex in a weighted connected graph, find shortest paths to other vertices using Dijkstra's algorithm.	<b>68-71</b>
<b>17</b>	Implement "Sum of Subsets" using Backtracking. "Sum of Subsets" problem: Find a subset of a given set $S = \{s_1, s_2, \dots, s_n\}$ of n	<b>72-74</b>

	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.	
<b>18</b>	Implement "N-Queens Problem" using Backtracking.	<b>75-77</b>

## Course Outcome

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

## 1 .Write a recursive program to

### a. Solve Towers-of-Hanoi problem

a) #include<stdio.h>

    #include<conio.h>

    #include<math.h>

void hanoi(int x, char from, char to, char aux)

{

if(x==1)

printf("Move Disk From %c to %c\n",from,to);

else

{

hanoi(x-1,from,aux,to);

printf("Move Disk From %c to %c\n",from,to);

hanoi(x-1,aux,to,from);

}}

int main()

{

int disk;

int moves;

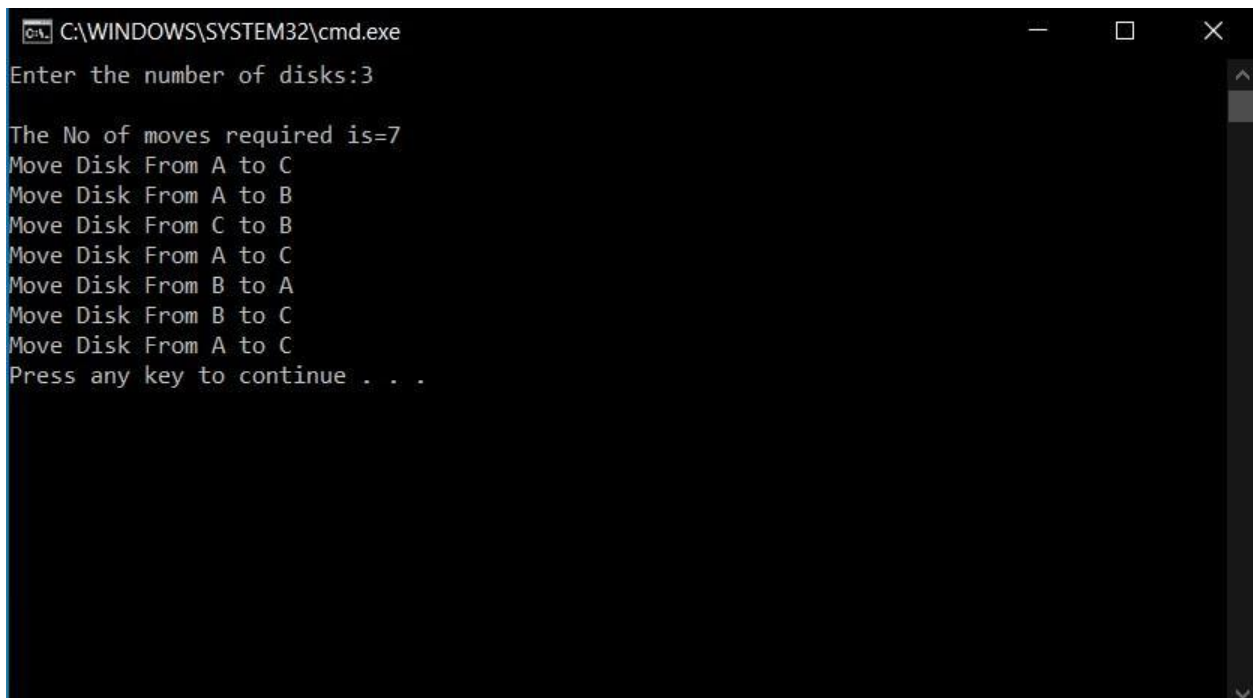
printf("Enter the number of disks:");

scanf("%d",&disk);

moves=pow(2,disk)-1;

```
printf("\nThe No of moves required is=%d \n",moves);  
hanoi(disk,'A','C','B');  
return 0; }
```

## OUTPUT-



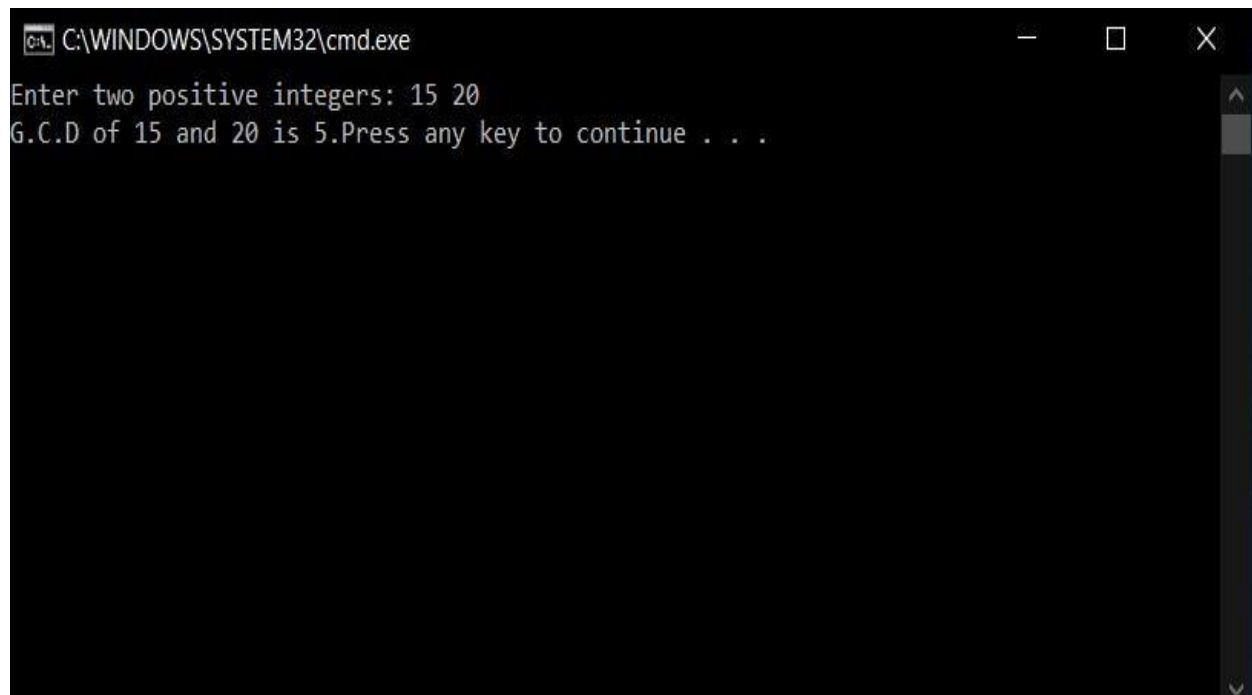
```
C:\WINDOWS\SYSTEM32\cmd.exe  
Enter the number of disks:3  
The No of moves required is=7  
Move Disk From A to C  
Move Disk From A to B  
Move Disk From C to B  
Move Disk From A to C  
Move Disk From B to A  
Move Disk From B to C  
Move Disk From A to C  
Press any key to continue . . .
```

## b)To find GCD

```
#include <stdio.h>  
  
int hcf(int x, int y);  
int main()  
{  
    int x;  
    int y;
```

```
printf("Enter two positive integers: ");  
scanf("%d %d", &x, &y);  
printf("G.C.D of %d and %d is %d.", x, y, hcf(x,y));  
return 0;  
}  
  
int hcf(int x, int y)  
{  
    if (y != 0)  
        return hcf(y,x%y);  
    else  
        return x;  
}
```

## OUTPUT -



The screenshot shows a Windows command prompt window with the title bar "C:\WINDOWS\SYSTEM32\cmd.exe". The prompt displays the output of a C program: "Enter two positive integers: 15 20" followed by "G.C.D of 15 and 20 is 5.Press any key to continue . . .". The window has standard Windows window controls (minimize, maximize, close) in the top right corner.

**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.**

```
#include<stdio.h>

#include<time.h>

#include<stdlib.h> /* To recognise exit function when compiling with
gcc*/

int bin_srch(int [],int,int,int);
int lin_srch(int [],int,int,int);
int n,a[10000];
int main()
{
    int ch,key,search_status,temp;
    clock_t end,start;
    unsigned long int i, j;
    while(1)
    {
        printf("\n1: Binary search\t 2: Linear search\t 3: Exit\n");
        printf("\nEnter your choice:\t");
        scanf("%d",&ch);

        switch(ch)
```



```

{
case 1:
    n=1000;
while(n<=5000)
{
    for(i=0;i<n;i++)
    {
        //a[i]=random(1000);
        a[i]=i; //Inserting numbers in Ascending order
    }
    key=a[n-1]; //Last element of the array
    start=clock();

        search_status=bin_srch(a,0,n-1,key);
    if(search_status==-1)
        printf("\nKey Not Found");
    else
        printf("\n Key found at position %d",search_status);
        //Dummy loop to create delay
    for(j=0;j<500000;j++){ temp=38/600;}

    end=clock();

```

```

    printf("\nTime for n=%d is %f
Secs",n,(((double)(end-start))/CLOCKS_PER_SEC));
    n=n+1000;
}
break;
case 2:
    n=1000;
while(n<=5000)
{
    for(i=0;i<n;i++)
    {
        //a[i]=random(10000);
        a[i]=i;
    }
    key=a[n-1]; //Last element of the array
    start=clock();
    search_status=lin_srch(a,0,n-1,key);
    if(search_status==-1)
        printf("\nKey Not Found");
    else
        printf("\n Key found at position %d",search_status);
    //Dummy loop to create delay

```

```

for(j=0;j<500000;j++){ temp=38/600;}

    end=clock();

    printf("\nTime for n=%d is %f
Secs",n,(((double)(end-start))/CLOCKS_PER_SEC));

    n=n+1000;

}

break;

default:

    exit(0);

}

getchar();

}

}

int bin_srch(int a[],int low,int high,int key) {
    int mid;
    if(low>high)
        return -1;
    mid=(low+high)/2;
    if(key==a[mid])
        return mid;
    if(key<a[mid])
        return bin_srch(a,low,mid-1,key);

```

```

else

    return bin_srch(a,mid+1,high,key);
}

int lin_srch(int a[],int i,int high,int key) {

    if(i>high)

        return -1;

    if(key==a[i])

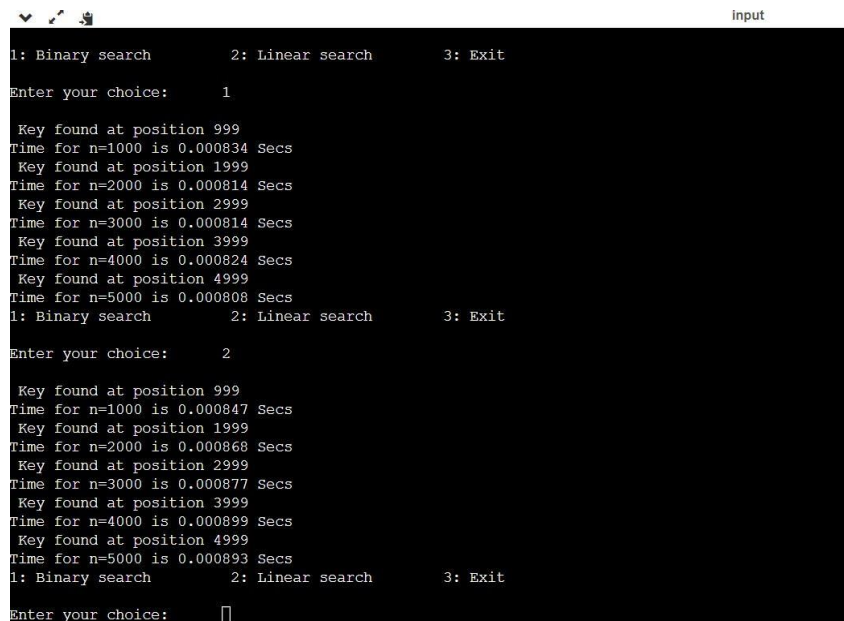
        return i;

    else

        return lin_srch(a,i+1,high,key); }

```

## OUTPUT -



```

input
1: Binary search      2: Linear search      3: Exit
Enter your choice:    1

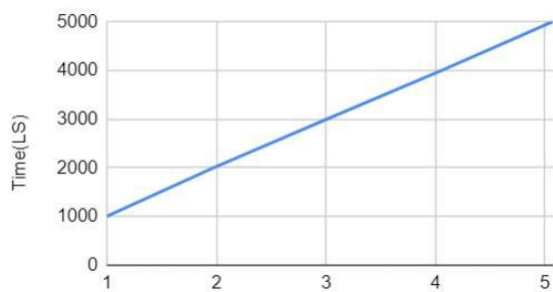
    Key found at position 999
Time for n=1000 is 0.000834 Secs
    Key found at position 1999
Time for n=2000 is 0.000814 Secs
    Key found at position 2999
Time for n=3000 is 0.000814 Secs
    Key found at position 3999
Time for n=4000 is 0.000824 Secs
    Key found at position 4999
Time for n=5000 is 0.000808 Secs
1: Binary search      2: Linear search      3: Exit
Enter your choice:    2

    Key found at position 999
Time for n=1000 is 0.000847 Secs
    Key found at position 1999
Time for n=2000 is 0.000868 Secs
    Key found at position 2999
Time for n=3000 is 0.000877 Secs
    Key found at position 3999
Time for n=4000 is 0.000899 Secs
    Key found at position 4999
Time for n=5000 is 0.000893 Secs
1: Binary search      2: Linear search      3: Exit
Enter your choice:

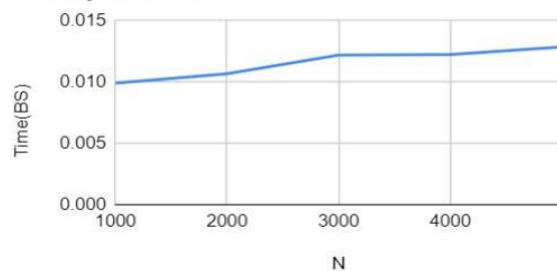
```

Time(LS)	N	Time(BS)
0.9987	1000	0.009863
1.9786	2000	0.010633
3.011	3000	0.012154
4.061	4000	0.012196
5.077	5000	0.012823

Linear search



Binary search



**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>

#include<stdlib.h>

/* To recognise exit function when compiling with gcc*/

void selsort(int n,int a[]);

int main()

{

    int a[15000],n,i,j,ch,temp;

    clock_t start,end;

    while(1)

    {

        printf("\n1:For manual entry of N value and array elements");

        printf("\n2:To display time taken for sorting number of elements N in

the range 500 to 14500");

        printf("\n3:To exit");

        printf("\nEnter your choice:");

        scanf("%d", &ch);

        switch(ch)

        {

            case 1: printf("\nEnter the number of elements: ");
```

```

scanf("%d",&n);
printf("\nEnter array elements: ");
for(i=0;i<n;i++)
{
    scanf("%d",&a[i]);
}
start=clock();
selsort(n,a);
end=clock();
printf("\nSorted array is: ");
for(i=0;i<n;i++)
printf("%d\t",a[i]);
printf("\n Time taken to sort %d numbers is %f Secs",n,
(((double)(end-start))/CLOCKS_PER_SEC));
break;
case 2:
n=500;
while(n<=14500) {
    for(i=0;i<n;i++)
    {
        //a[i]=random(1000);
        a[i]=n-i;
    }
}

```

```

        }
start=clock();
selsort(n,a);
//Dummy loop to create delay
for(j=0;j<500000;j++){ temp=38/600;}
end=clock();
printf("\n Time taken to sort %d numbers is %f Secs",n,
(((double)(end-start))/CLOCKS_PER_SEC));
        n=n+1000;
    }
    break;
case 3: exit(0);
}
return 0;
}
}
void selsort(int n,int a[])
{
    int i,j,t,small,pos;
    for(i=0;i<n-1;i++)
    {
        pos=i;

```

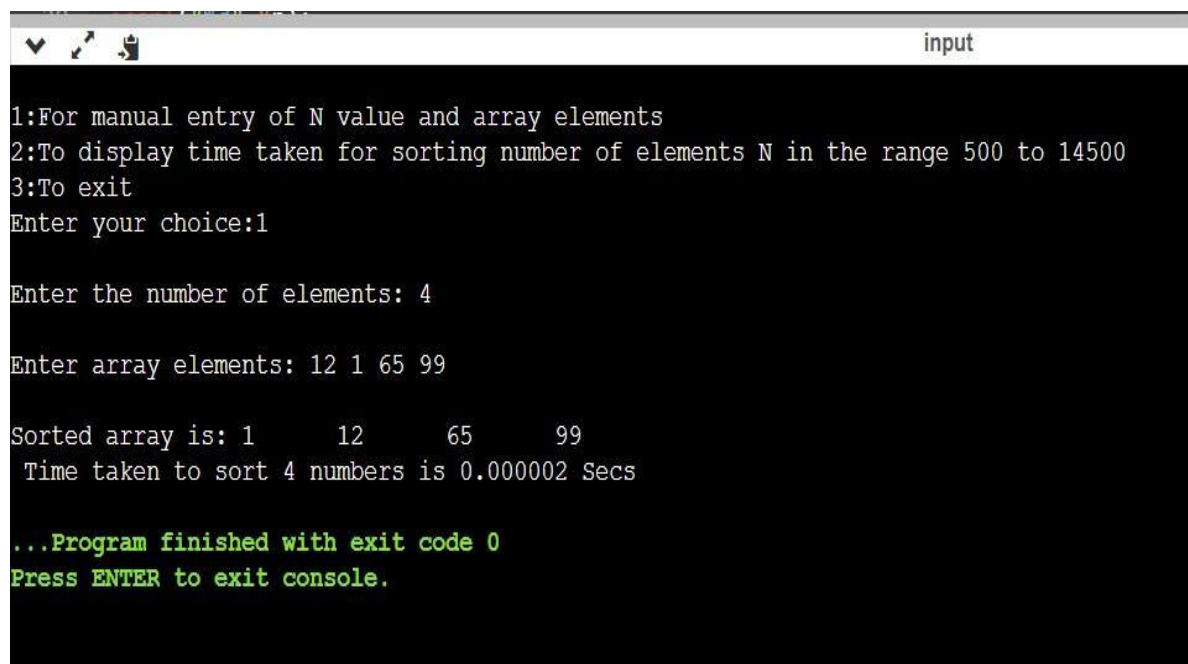


```

small=a[i];
for(j=i+1;j<n;j++)
{ if(a[j]<small)
{ small=a[j];
  pos=j; } }
t=a[i];
a[i]=a[pos];
a[pos]=t; } }

```

## OUTPUT-



```

input
1:For manual entry of N value and array elements
2:To display time taken for sorting number of elements N in the range 500 to 14500
3:To exit
Enter your choice:1

Enter the number of elements: 4

Enter array elements: 12 1 65 99

Sorted array is: 1      12      65      99
Time taken to sort 4 numbers is 0.000002 Secs

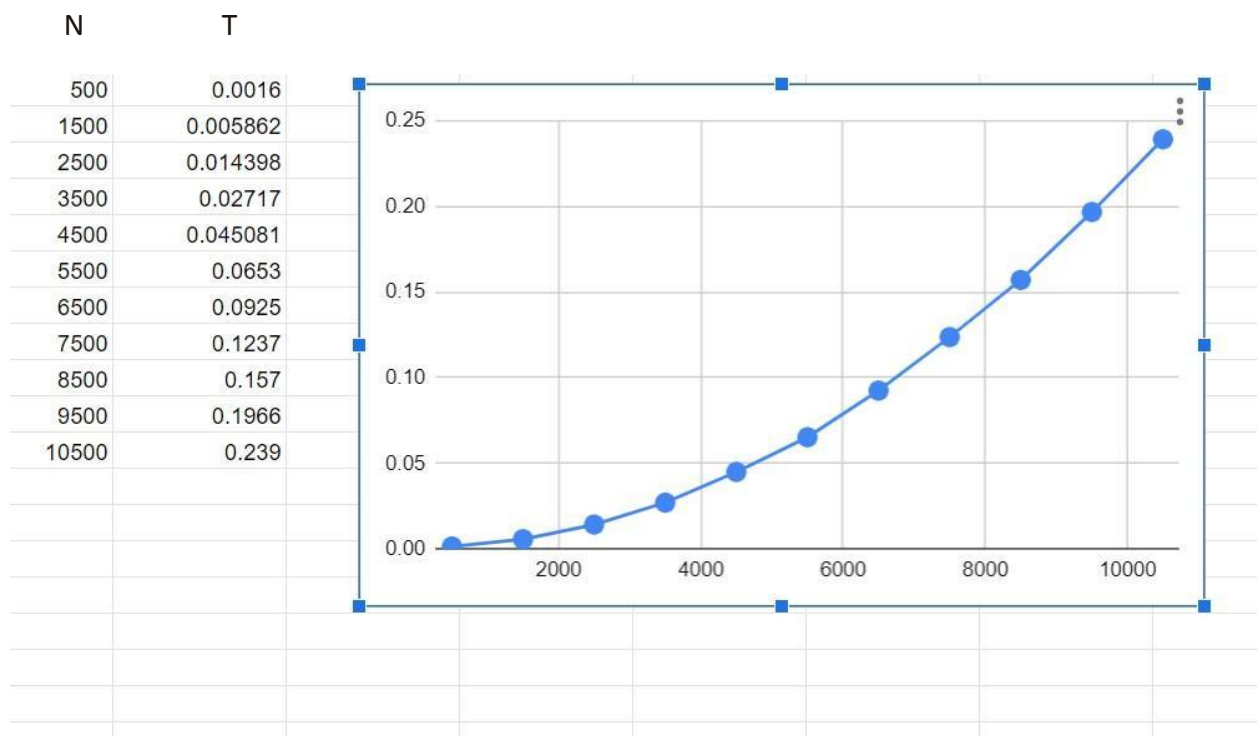
...Program finished with exit code 0
Press ENTER to exit console.

```

```
input
1:For manual entry of N value and array elements
2:To display time taken for sorting number of elements N in the range 500 to 14500
3:To exit
Enter your choice:2

Time taken to sort 500 numbers is 0.001610 Secs
Time taken to sort 1500 numbers is 0.005862 Secs
Time taken to sort 2500 numbers is 0.014398 Secs
Time taken to sort 3500 numbers is 0.027170 Secs
Time taken to sort 4500 numbers is 0.045081 Secs
Time taken to sort 5500 numbers is 0.065311 Secs
Time taken to sort 6500 numbers is 0.092552 Secs
Time taken to sort 7500 numbers is 0.123708 Secs
Time taken to sort 8500 numbers is 0.157582 Secs
Time taken to sort 9500 numbers is 0.196691 Secs
Time taken to sort 10500 numbers is 0.239021 Secs
Time taken to sort 11500 numbers is 0.291980 Secs
Time taken to sort 12500 numbers is 0.340763 Secs
Time taken to sort 13500 numbers is 0.403541 Secs
Time taken to sort 14500 numbers is 0.468520 Secs

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



**4. Write program to do the following: a) Print all the nodes reachable from a given starting node in a digraph using BFS method.**

```
#include<stdio.h>

#include<conio.h>

int a[10][10],n;

void bfs(int);

int main()
{
    int i,j,src;

    printf("\nEnter the no of nodes:\t");
    scanf("%d",&n);

    printf("\nEnter the adjacency matrix:\n");
    for(i=1;i<=n;i++)
    {
        for(j=1;j<=n;j++)
        {
            scanf("%d",&a[i][j]);
        }
    }

    printf("\nEnter the source node:\t");
    scanf("%d",&src);

    bfs(src);

    return 0;
```

```

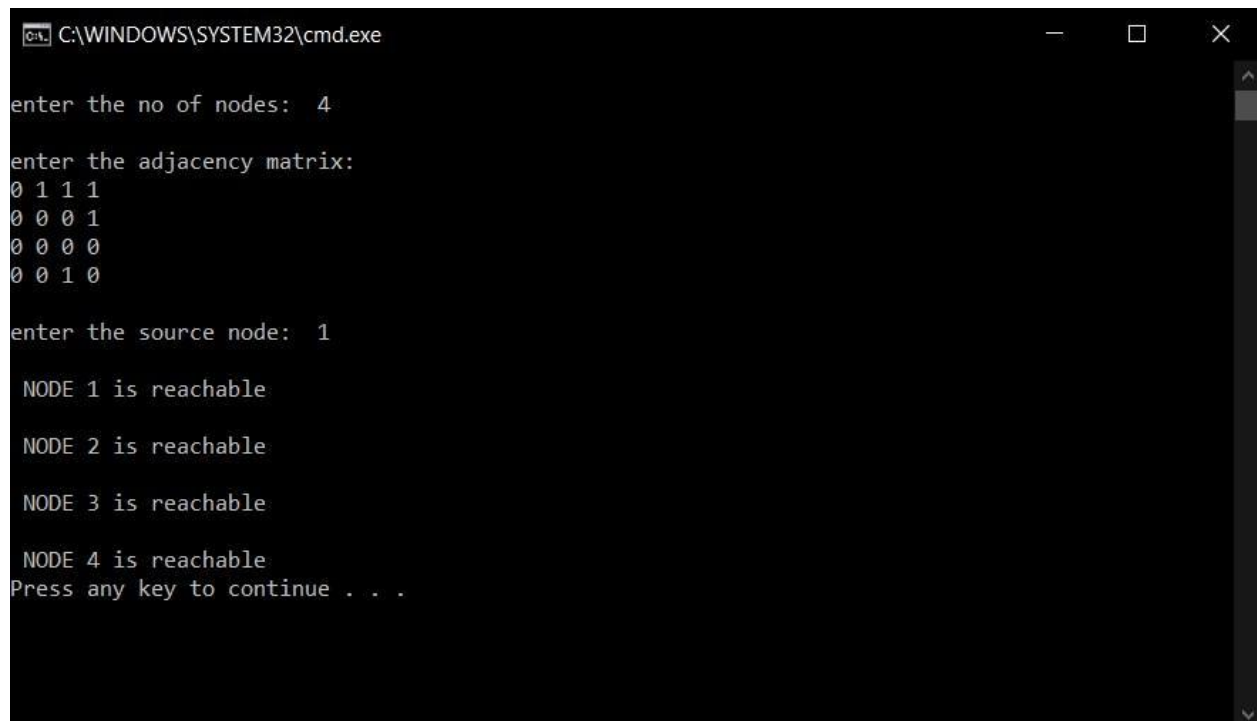
}
void bfs(int src) {
int q[10],f=0,r=-1,vis[10],i,j;
for(j=1;j<=n;j++)
vis[j]=0;
vis[src]=1;
r=r+1;
q[r]=src;
while(f<=r) {
i=q[f];
f=f+1;
for(j=1;j<=n;j++)
{
if(a[i][j]==1&&vis[j]!=1) {
vis[j]=1;
r=r+1;
q[r]=j;
}
}
}
for(j=1;j<=n;j++) {
if(vis[j]!=1)

```

```

printf("\n NODE %d is not reachable\n",j);
else
{ printf("\n NODE %d is reachable\n",j);
}}
```

**OUTPUT -**



```

C:\WINDOWS\SYSTEM32\cmd.exe

enter the no of nodes: 4

enter the adjacency matrix:
0 1 1 1
0 0 0 1
0 0 0 0
0 0 1 0

enter the source node: 1

NODE 1 is reachable

NODE 2 is reachable

NODE 3 is reachable

NODE 4 is reachable
Press any key to continue . . .
```

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

```

#include<stdio.h>

#include<conio.h>

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

int dfs(int);

int main()
{
```

```

int i,j,src,ans;
for(j=1;j<=n;j++)
{
vis[j]=0;
}
printf("\nenter the no of nodes:\t");
scanf("%d",&n);
printf("\nenter the adjacency matrix:\n");
for(i=1;i<=n;i++)
{
for(j=1;j<=n;j++)
{
scanf("%d",&a[i][j]);
}
}
printf("\nenter the source node:\t");
scanf("%d",&src);
ans=dfs(src);
if(ans==1)
{
printf("\ngraph is connected\n");
}
else

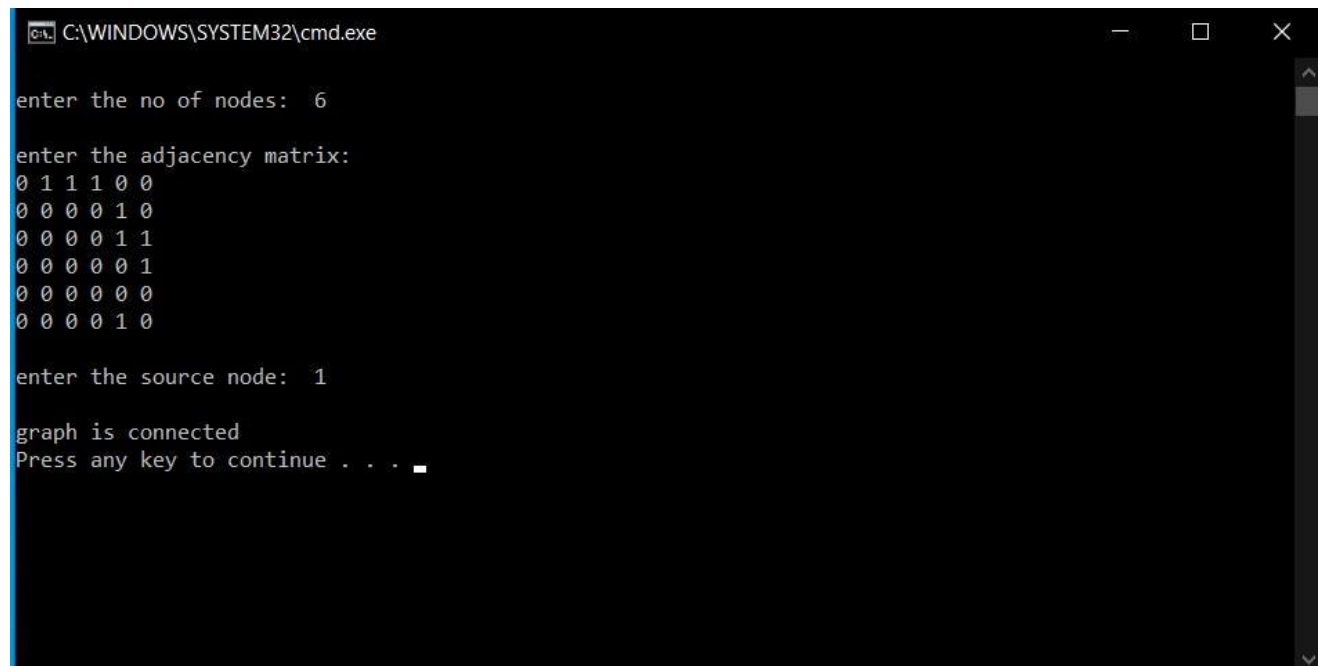
```

```

{ printf("\ngraph is not connected\n"); }
return 0;
}
int dfs(int src)
{
int j;
vis[src]=1;
for(j=1;j<=n;j++)
{
if(a[src][j]==1&&vis[j]!=1)
{
dfs(j);
} }
for(j=1;j<=n;j++)
{
if(vis[j]!=1)
{
return 0;
} }
return 1; }

```

## OUTPUT-



```
C:\WINDOWS\SYSTEM32\cmd.exe

enter the no of nodes: 6

enter the adjacency matrix:
0 1 1 1 0 0
0 0 0 0 1 0
0 0 0 0 1 1
0 0 0 0 0 1
0 0 0 0 0 0
0 0 0 0 1 0

enter the source node: 1

graph is connected
Press any key to continue . . .
```



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

```
#include <math.h>

#include <stdio.h>

#include <time.h>

void delay(){

    long n;

    for(n=0;n<1000;n++){

        int a = 10/10;

    }

}

void insertionSort(int arr[], int n)

{

    int i, val, j;

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

        val = arr[i];

        j = i - 1;

        while (j >= 0 && arr[j] > val) {

            arr[j + 1] = arr[j];

            j --;

            delay();

        }

    }
```

```

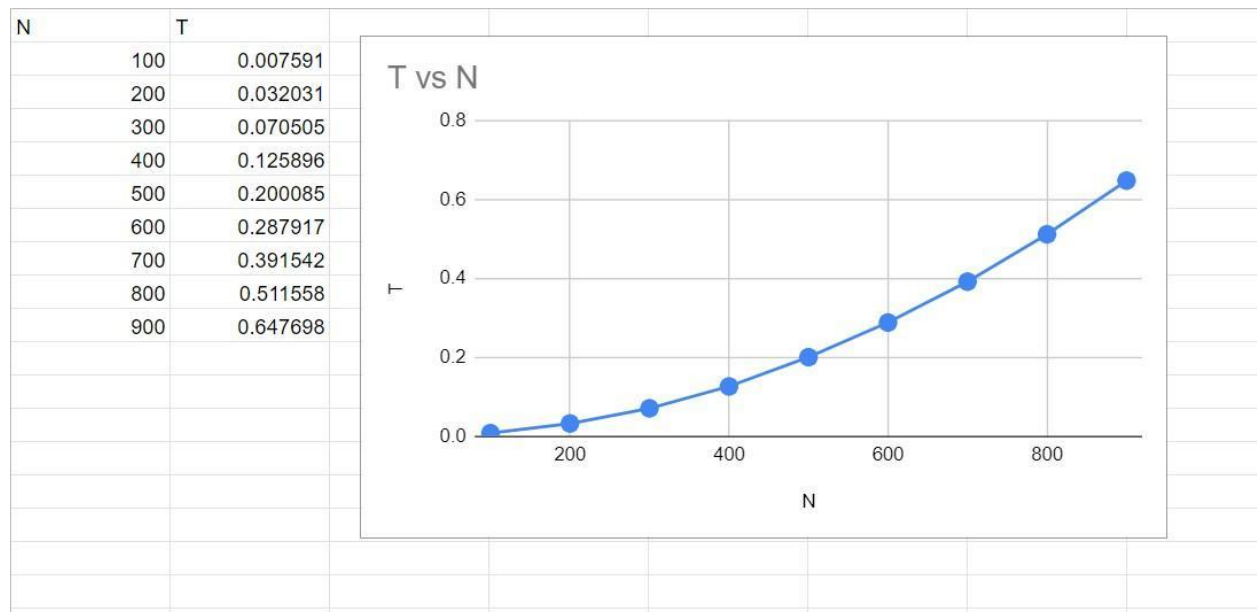
arr[j + 1] = val;
}
}
int main()
{ int arr[1500],n=100,i;
  double start,end;
  while(n<=1200){
    for(i=0;i<n;i++){
      arr[i]=i;
    }
    start = clock();
    insertionSort(arr, n);
    end=clock();
    printf("n=%d time= %f \n",n,(end-start)/CLOCKS_PER_SEC);
    n=n+100;
  }
  return 0;
}

```

## OUTPUT-

```
input
to sort out element using insertion sort
enter size of array
12
enter array elements
array elements are
7 10 9 7 5 7 10 0 9 1 2 7 sorted array is
0 1 2 5 7 7 7 7 9 9 10 10 time taken is 0.001569

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



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

```
#include<stdio.h>

#include<conio.h>

void source_removal(int n, int a[10][10])
{
    int i,j,k,u,v,top,s[10],t[10],indeg[10],sum;
    for(i=0;i<n;i++)
    {
        sum=0;
        for(j=0;j<n;j++)
        {
            sum+=a[j][i];
        }
        indeg[i]=sum;
    }
    top=-1;
    for(i=0;i<n;i++)
    {
        if(indeg[i]==0)
        {
            s[++top]=i;
```

```

}
}
k=0;
while(top!=-1)
{
u=s[top--];
t[k++]=u;
for(v=0;v<n;v++)
{
if(a[u][v]==1)
{
indeg[v]=indeg[v]-1;
if(indeg[v]==0)
s[++top]=v;
}
}
}
for(i=0;i<n;i++)
{
printf("%d\n", t[i]);
}
}

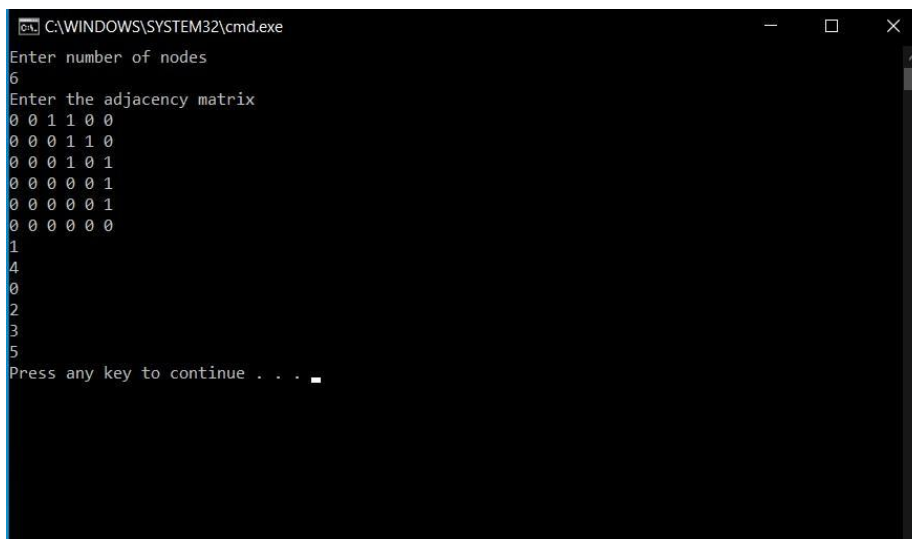
```

```

int main()
{ int i,j,a[10][10],n;
printf("Enter number 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]);
} }
source_removal(n,a);
return 0;
}

```

## OUTPUT-



```

C:\WINDOWS\SYSTEM32\cmd.exe
Enter number of nodes
6
Enter the adjacency matrix
0 0 1 1 0 0
0 0 0 1 1 0
0 0 0 1 0 1
0 0 0 1 0 1
0 0 0 0 0 1
0 0 0 0 0 1
0 0 0 0 0 0
1
4
0
2
3
5
Press any key to continue . . .

```

## 7.Implement Johnson Trotter algorithm to generate permutations

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
int flag = 0;
```

```
int swap(int *a,int *b)
```

```
{
```

```
    int t = *a;
```

```
    *a = *b;
```

```
    *b = t;
```

```
}
```

```
int search(int arr[],int num,int mobile)
```

```
{
```

```
    int g;
```

```
    for(g=0;g<num;g++)
```

```
    {
```

```
        if(arr[g] == mobile)
```

```
        {
```

```
            return g+1;
```

```
        }
```

```
    else
```

```
    {
```

```

        flag++;
    }
}
return -1;
}

int find_Moblie(int arr[],int d[],int num)
{
    int mobile = 0;
    int mobile_p = 0;
    int i;
    for(i=0;i<num;i++)
    {
        if((d[arr[i]-1] == 0) && i != 0)
        {
            if(arr[i]>arr[i-1] && arr[i]>mobile_p)
            {
                mobile = arr[i];
                mobile_p = mobile;
            }
            else
            {
                flag++ ; }
        }
    }
}

```



```

    }
    else if((d[arr[i]-1] == 1) & i != num-1)
    {
        if(arr[i]>arr[i+1] && arr[i]>mobile_p)
        {
            mobile = arr[i];
            mobile_p = mobile;
        }
        else
        {
            flag++;
        }
    }
    else
    {
        flag++;
    }
}

if((mobile_p == 0) && (mobile == 0))
    return 0;
else
    return mobile;

```

```

}

void permutations(int arr[],int d[],int num)
{
    int i;
    int mobile = find_Moblie(arr,d,num);
    int pos = search(arr,num,mobile);
    if(d[arr[pos-1]-1]==0)
        swap(&arr[pos-1],&arr[pos-2]);
    else
        swap(&arr[pos-1],&arr[pos]);
    for(int i=0;i<num;i++)
    {
        if(arr[i] > mobile)
        {
            if(d[arr[i]-1]==0)
                d[arr[i]-1] = 1;
            else
                d[arr[i]-1] = 0;
        }
    }
    for(i=0;i<num;i++)
    {

```

```
        printf(" %d ",arr[i]);  
    }  
}
```

```
int factorial(int k)
```

```
{  
    int f = 1;  
    int i = 0;  
    for(i=1;i<k+1;i++)  
    {  
        f = f*i;  
    }  
    return f;  
}
```

```
int main()
```

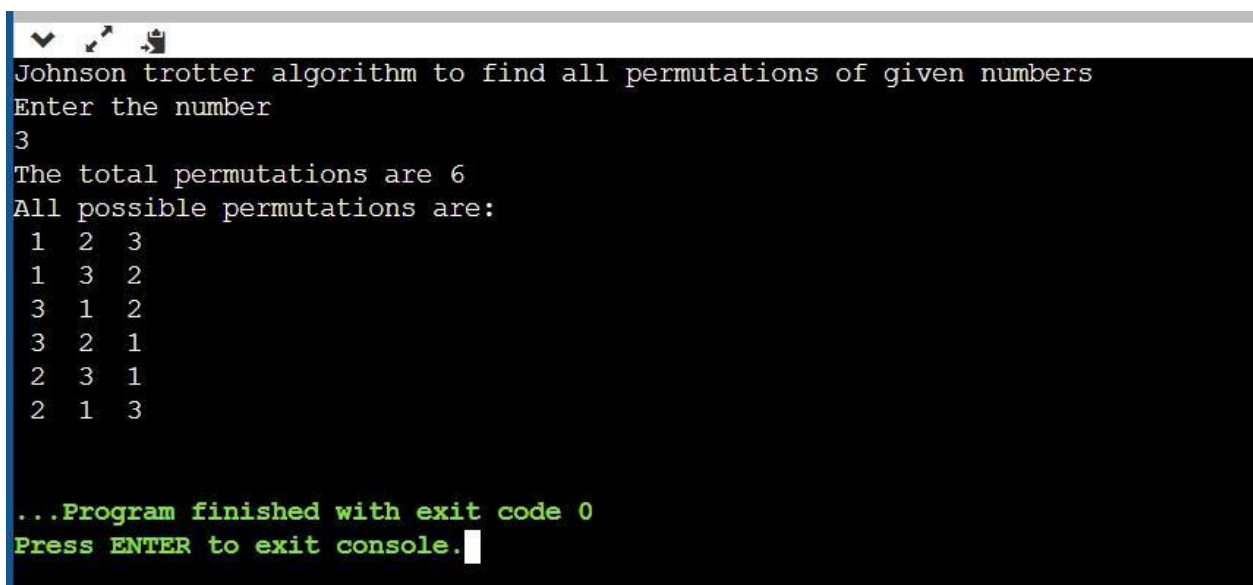
```
{  
    int num = 0;  
    int i;  
    int j;  
    int z = 0;  
  
    printf("Johnson trotter algorithm to find all permutations of given  
numbers \n");
```

```

printf("Enter the number\n");
scanf("%d",&num);
int arr[num],d[num];
z = factorial(num);
printf("The total permutations are %d",z);
printf("\nAll possible permutations are: \n");
for(i=0;i<num;i++)
{
    d[i] = 0;
    arr[i] = i+1;
    printf(" %d ",arr[i]);
}
printf("\n");
for(j=1;j<z;j++)
{
    permutations(arr,d,num);
    printf("\n");
}
return 0;
}

```

## OUTPUT -



```
Johnson trotter algorithm to find all permutations of given numbers
Enter the number
3
The total permutations are 6
All possible permutations are:
1 2 3
1 3 2
3 1 2
3 2 1
2 3 1
2 1 3

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

**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 mergesort(int a[],int i,int j);

void merge(int a[],int i1,int j1,int i2,int j2);

int main()

{

    clock_t start,end;

    int a[3000],n,i;

    printf("Enter no of elements:");

    scanf("%d",&n);

    printf("Enter array elements:");

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

        a[i] = rand()%1000;

    start = clock();

    mergesort(a,0,n-1);

    end = clock();

    printf("\nSorted array is :");

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

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

```

printf("\nSeconds taken %lf", (double)(end-start)/CLOCKS_PER_SEC);
return 0;
}
void mergesort(int a[], int i, int j)
{
    int mid;
    if(i < j)
    {
        mid = (i+j)/2;
        mergesort(a, i, mid);
        mergesort(a, mid+1, j);
        merge(a, i, mid, mid+1, j);
    }
}
void merge(int a[], int i1, int j1, int i2, int j2)
{
    int temp[3000];
    int i, j, k;
    i = i1;
    j = i2;
    k = 0;
    while(i <= j1 && j <= j2)

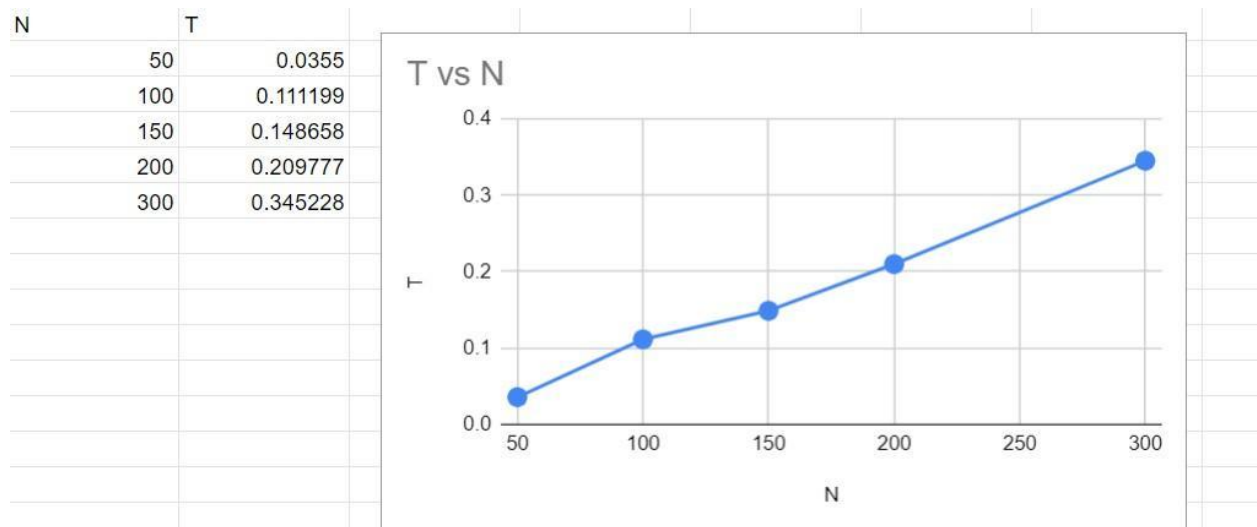
```

```
{for(int j=0;j<100000;j++);  
if(a[i]<a[j])  
temp[k++]=a[i++];  
else  
temp[k++]=a[j++];  
}  
while(i<=j1)  
temp[k++]=a[i++];  
while(j<=j2)  
temp[k++]=a[j++];  
for(i=i1,j=0;i<=j2;i++,j++)  
a[i]=temp[j];  
}
```



## OUTPUT-

```
C:\WINDOWS\SYSTEM32\cmd.exe
Enter no of elements:7
Enter array elements:
Sorted array is :41 169 334 467 478 500 724
Seconds taken 0.000000Press any key to continue . . .
```



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

```
#include<stdio.h>
```

```
#include<time.h>
```

```
#include<stdlib.h>
```

```
void quicksort(int number[5000],int first,int last){
```

```
    int i, j, pivot, temp;
```

```
    if(first<last){
```

```
        pivot=first;
```

```
        i=first;
```

```
        j=last;
```

```
        while(i<j){
```

```
            for(int x=0;x<100000;x++);
```

```
            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(){
    clock_t start,end;
    int i, count, number[5000];
    printf("No. of elements: ");
    scanf("%d",&count);
    printf("Enter %d elements: ", count);
    for(i=0;i<count;i++)
        number[i] = rand()%1000;
    start = clock();
    quicksort(number,0,count-1);
    end = clock();
    printf("Order of Sorted elements: ");
    for(i=0;i<count;i++)

```

```

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

    printf("\nSeconds taken %lf", (double)(end-start)/CLOCKS_PER_SEC);

    return 0;
}

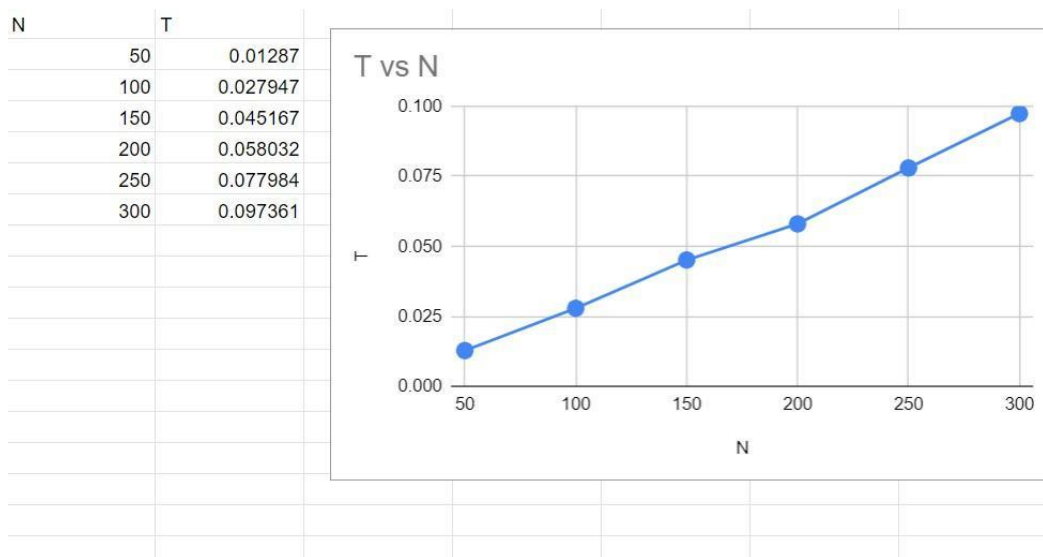
```

## OUTPUT-

```

C:\WINDOWS\SYSTEM32\cmd.exe
No. of elements: 20
Enter 20 elements: Order of Sorted elements: 41 145 169 281 334 358 436 464 467 478 491 500 705 724 827 827 942 961 962 995
Seconds taken 0.000000Press any key to continue . . .

```



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

```
#include <stdio.h>

#include <time.h>

#include <stdlib.h>

#include <math.h>

void swap(int *,int *);

void heapify(int [],int,int);

void heapSort(int[], int);

int main()

{

    int a[15000], n, i, j, ch, temp;

    clock_t start, end;

    while (1)

    {

        printf("\n1:FOR MANUAL ENTRY");

        printf("\n2:DISPLAY TIME TAKEN TO SORT ELEMENTS FROM RANGE 500 TO 15000");

        printf("\n3:EXIT");

        printf("\nENTER YOUR CHOICE:");

        scanf("%d", &ch);

        switch (ch)

        {
```

**case 1:**

```
printf("\nEnter NUMBER OF ARRAY ELEMENTS: ");  
scanf("%d", &n);  
printf("\nEnter ARRAY ELEMENTS: ");  
for (i = 0; i < n; i++)  
{  
    scanf("%d", &a[i]);  
}  
start = clock();  
heapSort(a, n);  
end = clock();  
printf("\nSORTED ARRAY IS: ");  
for (i = n-1; i >= 0; i--)  
    printf("%d\t", a[i]);  
  
    printf("\n TIME TAKEN TO SORT %d NUMBERS IS %f SECS", n,  
((double)(end - start)) / CLOCKS_PER_SEC));  
  
    break;
```

**case 2:**

```
n = 500;  
while (n <= 14500)  
{  
    for (i = 0; i < n; i++)
```

```

    {
        //a[i]=rand()%n;
        a[i] = n - i;
    }
    start = clock();
    heapSort(a, n);
    for (j = 0; j < 500000; j++)
    {
        temp = 38 / 600;
    }
    end = clock();
    printf("\n TIME TAKEN TO SORT %d NUMBERS IS %f SECS", n,
(((double)(end - start)) / CLOCKS_PER_SEC));
    n = n + 1000;
}
break;
case 3:
    exit(0);
}
getchar();
}
}

```

```

void swap(int *a, int *b)
{
    int temp = *a;
    *a = *b;
    *b = temp;
}

void heapify(int arr[], int n, int i)
{
    int largest = i;
    int left = 2 * i + 1;
    int right = 2 * i + 2;
    if (left < n && arr[left] > arr[largest])
        largest = left;
    if (right < n && arr[right] > arr[largest])
        largest = right;
    if (largest != i)
    {
        swap(&arr[i], &arr[largest]);
        heapify(arr, n, largest);
    }
}

```



```
void heapSort(int arr[], int n)
{
    for (int i = n / 2 - 1; i >= 0; i--)
        heapify(arr, n, i);
    for (int i = n - 1; i >= 0; i--)
    {
        swap(&arr[0], &arr[i]);
        heapify(arr, i, 0);
    }
}
```

**OUTPUT-**

```

1:FOR MANUAL ENTRY
2:DISPLAY TIME TAKEN TO SORT ELEMENTS FROM RANGE 500 TO 15000
3:EXIT
ENTER YOUR CHOICE:1

ENTER NUMBER OF ARRAY ELEMENTS: 5

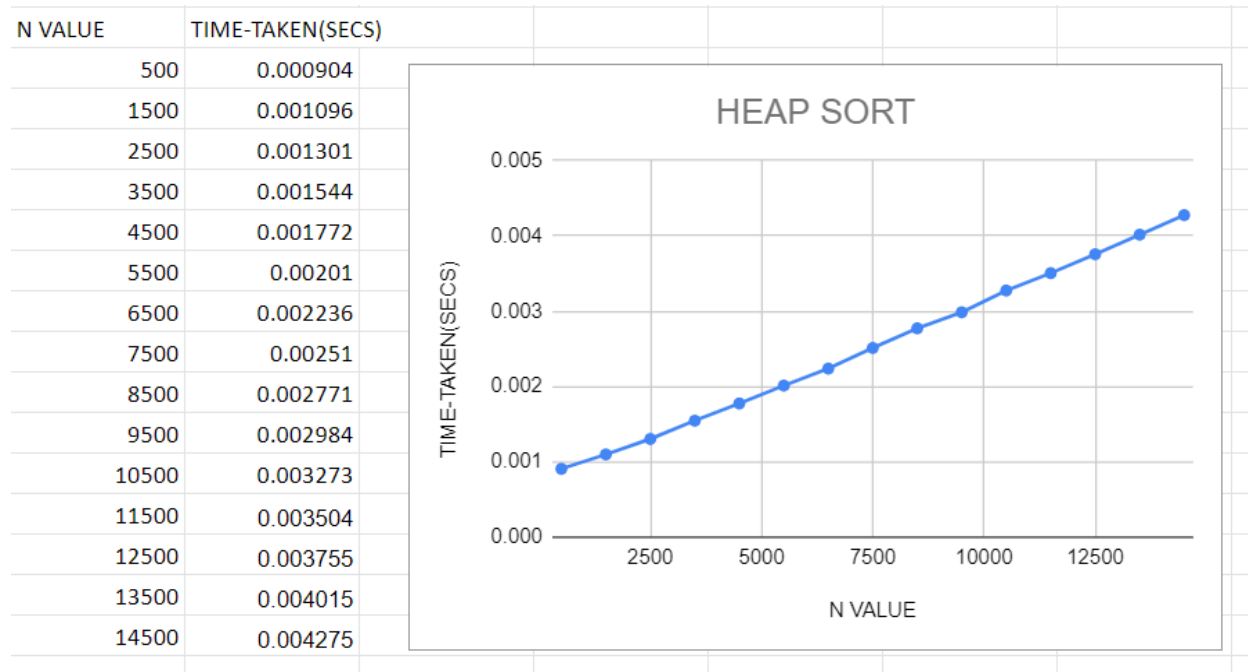
ENTER ARRAY ELEMENTS: 5 4 3 2 1

SORTED ARRAY IS: 1      2      3      4      5
TIME TAKEN TO SORT 5 NUMBERS IS 0.000003 SECS
1:FOR MANUAL ENTRY
2:DISPLAY TIME TAKEN TO SORT ELEMENTS FROM RANGE 500 TO 15000
3:EXIT
ENTER YOUR CHOICE:2

TIME TAKEN TO SORT 500 NUMBERS IS 0.000904 SECS
TIME TAKEN TO SORT 1500 NUMBERS IS 0.001096 SECS
TIME TAKEN TO SORT 2500 NUMBERS IS 0.001301 SECS
TIME TAKEN TO SORT 3500 NUMBERS IS 0.001544 SECS
TIME TAKEN TO SORT 4500 NUMBERS IS 0.001772 SECS
TIME TAKEN TO SORT 5500 NUMBERS IS 0.002010 SECS
TIME TAKEN TO SORT 6500 NUMBERS IS 0.002236 SECS
TIME TAKEN TO SORT 7500 NUMBERS IS 0.002510 SECS
TIME TAKEN TO SORT 8500 NUMBERS IS 0.002771 SECS
TIME TAKEN TO SORT 9500 NUMBERS IS 0.002984 SECS
TIME TAKEN TO SORT 10500 NUMBERS IS 0.003273 SECS
TIME TAKEN TO SORT 11500 NUMBERS IS 0.003504 SECS
TIME TAKEN TO SORT 12500 NUMBERS IS 0.003755 SECS
TIME TAKEN TO SORT 13500 NUMBERS IS 0.004015 SECS
TIME TAKEN TO SORT 14500 NUMBERS IS 0.004275 SECS
1:FOR MANUAL ENTRY
2:DISPLAY TIME TAKEN TO SORT ELEMENTS FROM RANGE 500 TO 15000
3:EXIT
ENTER YOUR CHOICE:3

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

```



### 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 finished with exit code 0
Press ENTER to exit console.

```

## 12) Implement 0/1 Knapsack problem using dynamic programming.

```
#include<stdio.h>

#include<conio.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();  
getch();  
}
```

```
void knapsack()  
{  
    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 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;
    }
}
```



## 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 solution vector is:
1      1      0      1

...Program finished with exit code 255
Press ENTER to exit console.
```

### 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;
  }
}

```

```

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 finished with exit code 0
Press ENTER to exit console.

```

**14) Find Minimum Cost Spanning Tree of a given undirected graph using Prim's algorithm.**

```
#include<stdio.h>

void prims();

int c[10][10],n;

void main()
{
    int i,j;

    printf("\nenter the no. of vertices: ");
    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();
}

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:
9999 3 9999 9999 6 5
3 9999 1 9999 9999 4
9999 1 9999 6 9999 4
9999 6 6 9999 8 5
6 9999 9999 8 9999 2
5 4 4 5 2 9999

1----->2=3
2----->3=1
2----->6=4
6----->5=2
6----->4=5

mincost=15

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

```

### 15) Find Minimum Cost Spanning Tree of an undirected graph using Kruskals algorithm

```
#include<stdio.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();
}
```



```

void kruskals()
{
    int i,j,u,v,a,b,min;
    int ne=0,mincost=0;
    int parent[10];
    for(i=1;i<=n;i++)
    {
        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 -

```
input
enter the no. of vertices:      6

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

2----->3=1
5----->6=2
1----->2=3
2----->6=4
4----->6=5

mincost=15

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

**16) From a given vertex in a weighted connected graph, find shortest paths to other vertices using Dijkstra's algorithm.**

```
#include<stdio.h>

void dijkstras();

int c[10][10],n,src;

void main()
{
    int i,j;

    printf("\nenter the no of vertices: ");
    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: ");
    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("\nthe shortest distance is:\n");
for(j=1;j<=n;j++)
{
    printf("\n%d----->%d=%d",src,j,dist[j]);
}
}

```

## OUTPUT -

```
enter the no of vertices: 5

enter the cost matrix:
99 3 99 7 99
3 99 4 2 99
99 4 99 5 6
7 2 5 99 4
99 99 6 4 99

enter the source node: 1

the shortest distance is:

1----->1=0
1----->2=3
1----->3=7
1----->4=5
1----->5=9

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

17) Implement “Sum of Subsets” using Backtracking. “Sum of Subsets” problem: Find a subset of a given set  $S = \{s_1, s_2, \dots, s_n\}$  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>
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("\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
1 2 6
Subset solution = 2
1 8

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

## 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;

```

```
    printf("Enter the number of Queens\n");  
    scanf("%d", &n);  
    nqueens(n);  
}
```

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