VISVESVARAYA TECHNOLOGICAL UNIVERSITY

“JnanaSangama”, Belgaum -590014, Karnataka.

LAB REPORT on

Analysis and Design of Algorithms

Submitted by

KOTTURU AMARNATH (1BM20CS074)

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

B.M.S. COLLEGE OF ENGINEERING BENGALURU-560019 May-2022 to July-2022

(Autonomous Institution under VTU)

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 **ANSHUMAAN CHANDRA (1BM20CS019),** 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.

Name of the Lab-In charge:
Designation
Department of CSE
BMSCE, Bengaluru

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

Index Sheet

9	Sort a given set of N integer elements using Quick Sort technique and compute its time taken.	
10	Sort a given set of N integer elements using Heap Sort technique and compute its time taken.	
11	Implement Warshall’s algorithm using dynamic programming	
12	Implement 0/1 Knapsack problem using dynamic programming.	
13	Implement All Pair Shortest paths problem using Floyd’s algorithm.	
14	Find Minimum Cost Spanning Tree of a given undirected graph using Prim’s algorithm.	
15	Find Minimum Cost Spanning Tree of a given undirected graph using Kruskals algorithm.	
16	From a given vertex in a weighted connected graph, find shortest paths to other vertices using Dijkstra’s algorithm.	
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.	
18	Implement “N-Queens Problem” using Backtracking.	

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.

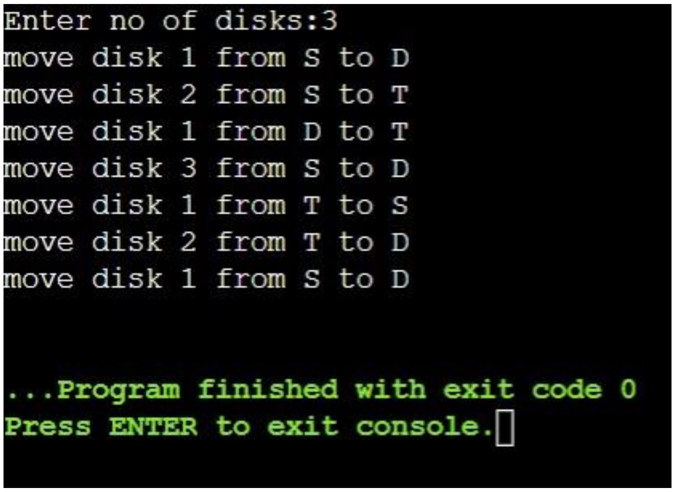
1. Write a recursive program to Solve

a) Towers-of-Hanoi problem b) To find GCD Program:

a)

```
#include<stdio.h>
void TOH(int n,char S,char T,char D){
   if(n==1)
      printf("move disk 1 from %c to %c \n",S,D);
else{
      TOH(n-1,S,D,T);
      printf("move disk %d from %c to %c\n",n,S,D);
      TOH(n-1,T,S,D);
   }
}
int main(){
   int n;
   printf("Enter no of disks:");
scanf("%d",&n);
   TOH(n,'S','T','D');

}
```

Result:

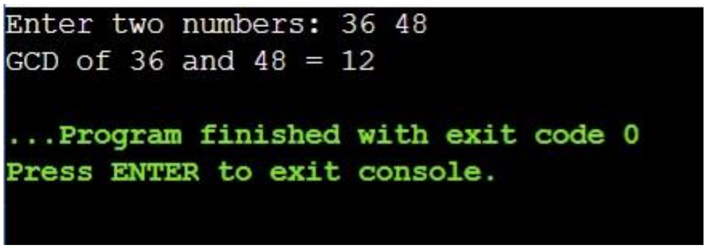


```
Enter no of disks:3
move disk 1 from S to D
move disk 2 from S to T
move disk 1 from D to T
move disk 3 from S to D
move disk 1 from T to S
move disk 2 from T to D
move disk 1 from S to D


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

b)

```
#include<stdio.h> int

gcd(int a, int b)

{     if(b!=0)        return

gcd(b, a%b);     else

return a;

}


int main()

{     int n1, n2, result;     printf("Enter two

numbers: ");     scanf("%d %d",&n1,&n2);
result = gcd(n1,n2);      printf("GCD of %d and %d
= %d",n1,n2,result);      return 0;
}
```

Result:



```
Enter two numbers: 36 48
GCD of 36 and 48 = 12

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

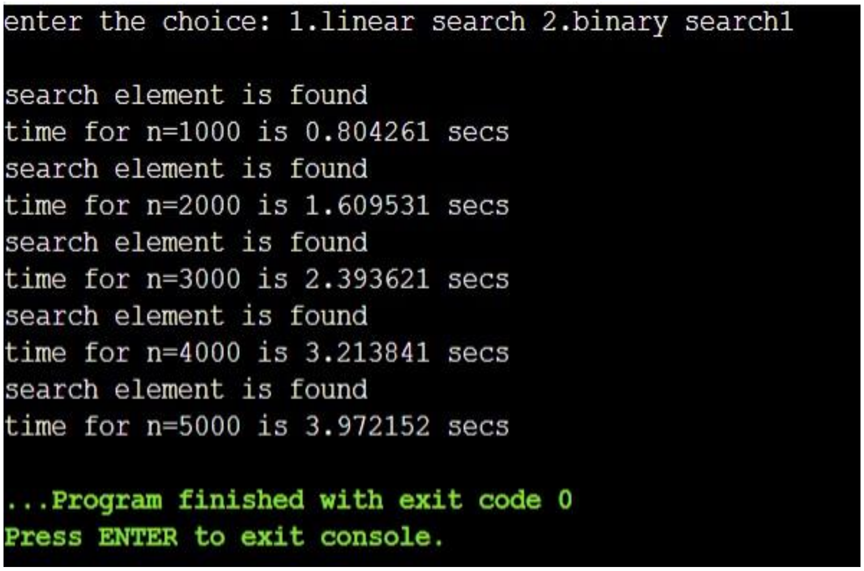
2. Implement Recursive Binary search and Linear search and determine the time required to search an element. Repeat the experiment for different values of N and plot a graph of the time taken versus N.

```
#include<stdio.h>

#include<conio.h>

#include<time.h>

void delay(){     int

i,temp;

  for(i=0;i<500000;i++){temp=33/550;}

}
```

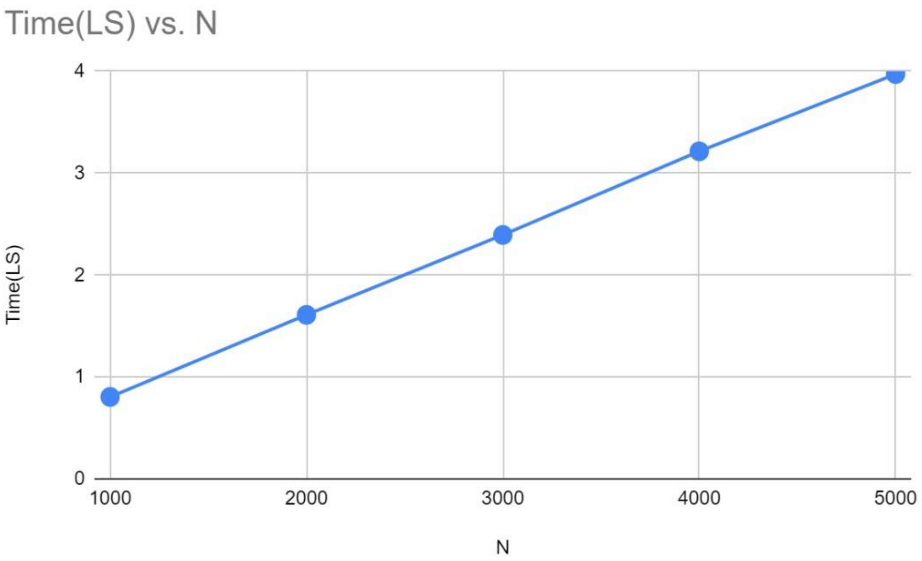
```
int main(){     int
a[15000];     int
j,i,c,key,n,o=-1;
  printf("enter the choice: 1.linear search 2.binary search");
scanf("%d",&c);    if(c==1){      n=1000;      while(n<=5000)
    {
    for(i=0;i<n;i++){
    a[i]=i;
   }
    key=a[n-1];
double start=clock();
   {
     for(i=0;i<n;i++){
delay();
if(a[i]==key)           o=0;
     }
     if(o==0)
        printf("\nsearch element is found");
        else
        printf("\nsearch failed:");
   }
  double end=clock();
  printf("\ntime for n=%d is %2f
secs",n,((endstart)/CLOCKS_PER_SEC));
n=n+1000;
}
}
else if(c==2)
{

n=1000;
while(n<=5000)
    {
    for(i=0;i<n;i++){
a[i]=i;
   }
    key=a[n-1];    int
b,e,m; b=0;e=n-1;
double start=clock();
while(b<=e){
m=(b+e)/2;   delay();
if(a[m]==key){      o=0;
break;
 }
```

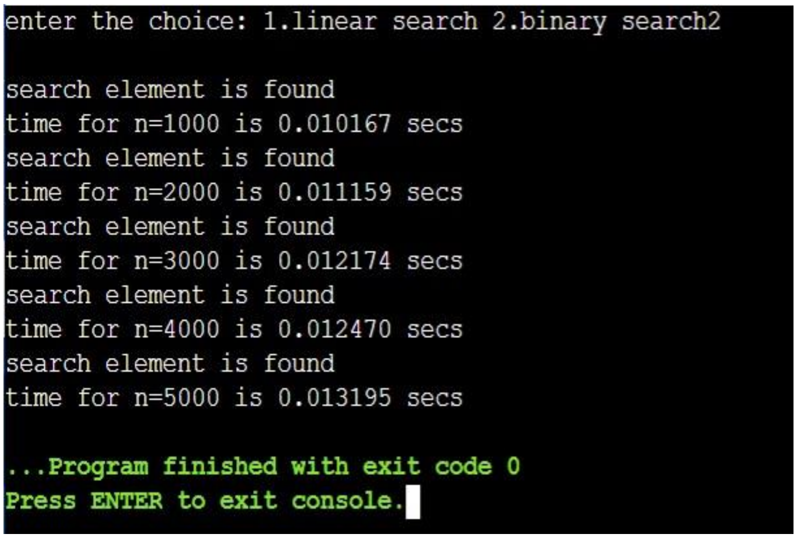
```
  else


  if(a[m]>key)   e=m-
1;
 else
b=m+1;
}
 if(o==0)
 printf("\nsearch element is found");
      else
 printf("\nsearch failed"); double
end=clock();
   printf("\ntime for n=%d is %2f
secs",n,((endstart)/CLOCKS_PER_SEC));
n=n+1000;
}
}
}
```

Result:

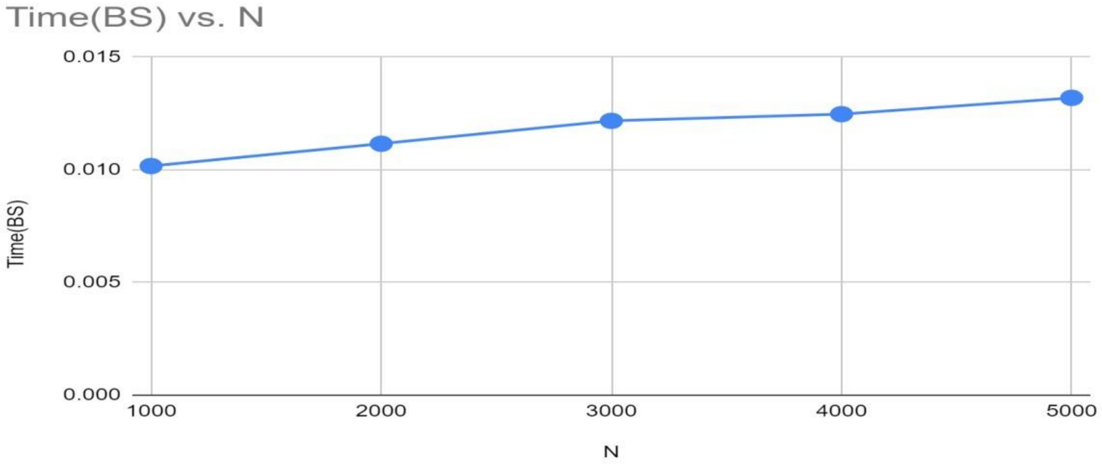


```
enter the choice: 1.linear search 2.binary search1

search element is found
time for n=1000 is 0.804261 secs
search element is found
time for n=2000 is 1.609531 secs
search element is found
time for n=3000 is 2.393621 secs
search element is found
time for n=4000 is 3.213841 secs
search element is found
time for n=5000 is 3.972152 secs

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





```
enter the choice: 1.linear search 2.binary search2

search element is found
time for n=1000 is 0.010167 secs
search element is found
time for n=2000 is 0.011159 secs
search element is found
time for n=3000 is 0.012174 secs
search element is found
time for n=4000 is 0.012470 secs
search element is found
time for n=5000 is 0.013195 secs

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

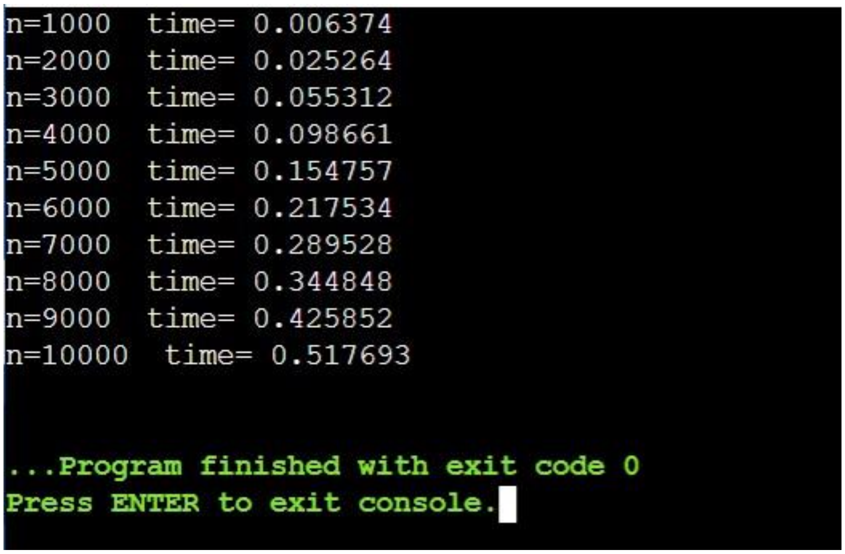


3.Sort a given set of N integer elements using Selection Sort technique and compute its time taken. Run the program for different values of N and record the time taken to sort.

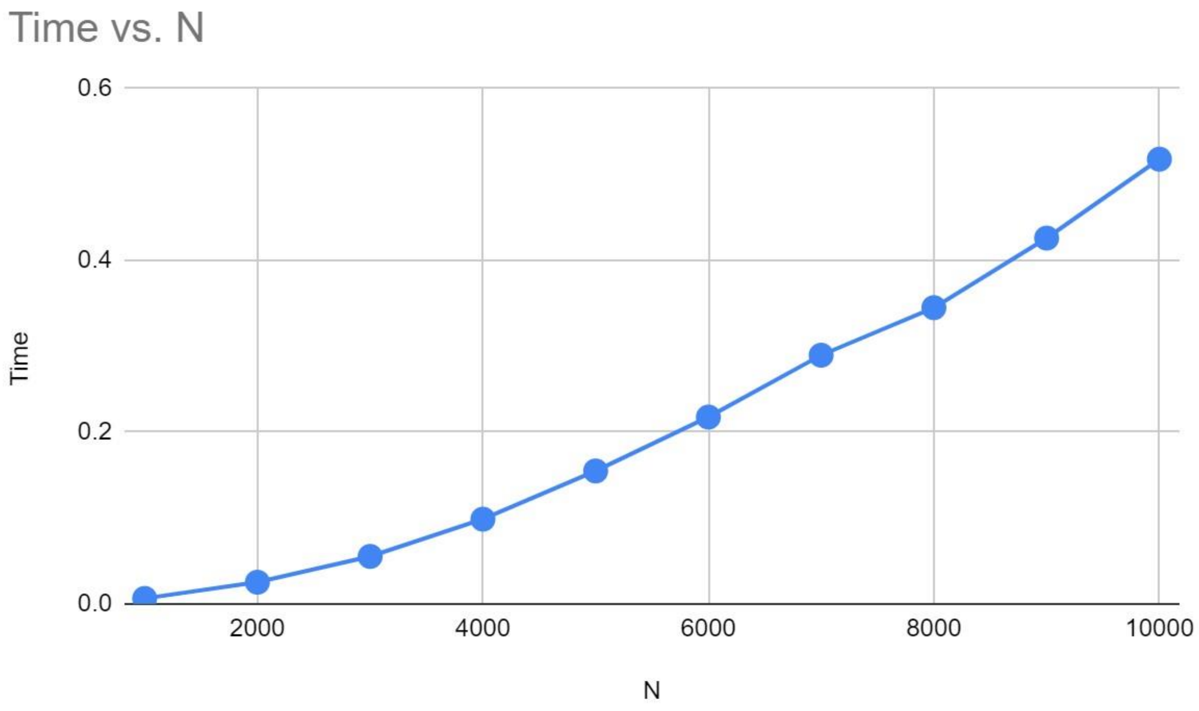
```
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
void delay(){     long
n;
for(n=0;n<10;n++){
int a = 10/10;
    }
}
void selectionsort(int arr[],int length){
int i,j;
    for(i=0;i<length-1;i++){
int min=i;
for(j=i+1;j<length;j++){
if(arr[j]>arr[min]){
min=j;                delay();
            }
        }
        {
            int temp=arr[min];
```

```
      arr[min]=arr[i];
arr[i]=temp;
    }
  }
}
int main()
{
  int arr[15000],n=1000,i;
double start,end;

  while(n<=10000){
for(i=0;i<n;i++){
arr[i]=i;
    }
    start = clock();     selectionsort(arr,n);     end=clock();
printf("n=%d  time= %f \n",n,(end-start)/CLOCKS_PER_SEC);
n=n+1000;
  }
}
```

Result:



```
n=1000  time= 0.006374
n=2000  time= 0.025264
n=3000  time= 0.055312
n=4000  time= 0.098661
n=5000  time= 0.154757
n=6000  time= 0.217534
n=7000  time= 0.289528
n=8000  time= 0.344848
n=9000  time= 0.425852
n=10000  time= 0.517693


...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.

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

a)

```
#include<stdio.h>

#include<conio.h>


int a[10][10],n;

void bfs(int);  void

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);
```

```
}

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;
   }
  }
```

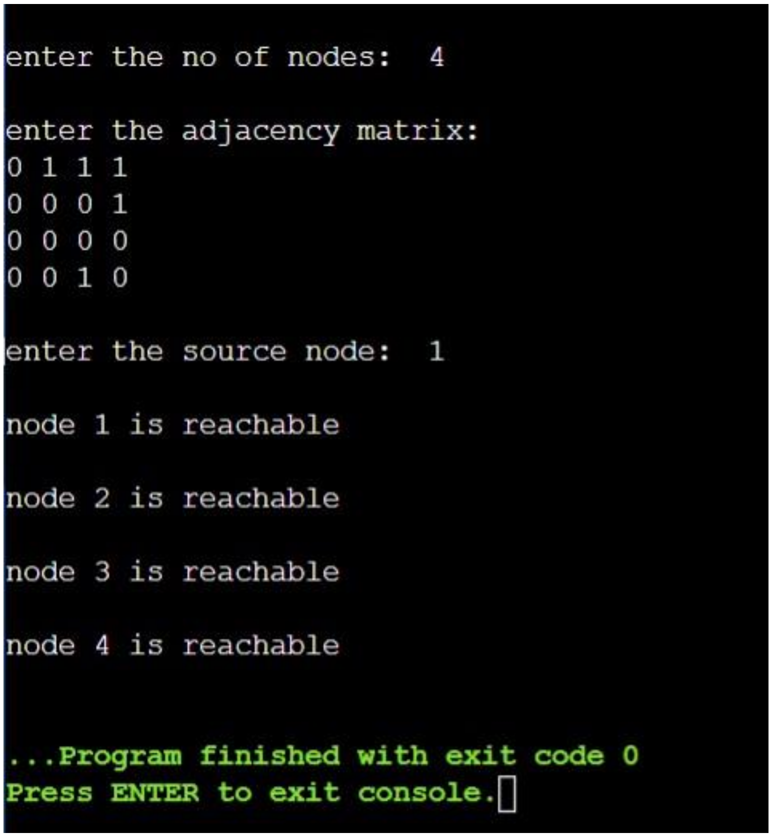
```
 }



for(j=1;j<=n;j++)  {
  if(vis[j]!=1)     printf("\nnode %d is not
reachable\n",j);    else
  {
   printf("\nnode %d is reachable\n",j);
  }
 }
}
```

b)

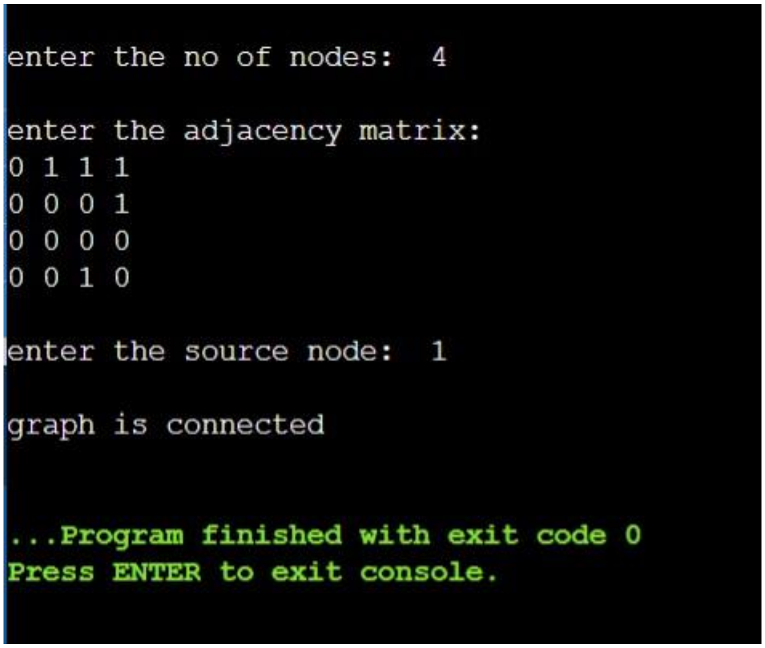
```
#include<stdio.h>
#include<conio.h>  int
a[10][10],n,vis[10];
int dfs(int);

void 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("\ngragh is not connected\n");
 }
 getch();
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;
}
```

Result:



```
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


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

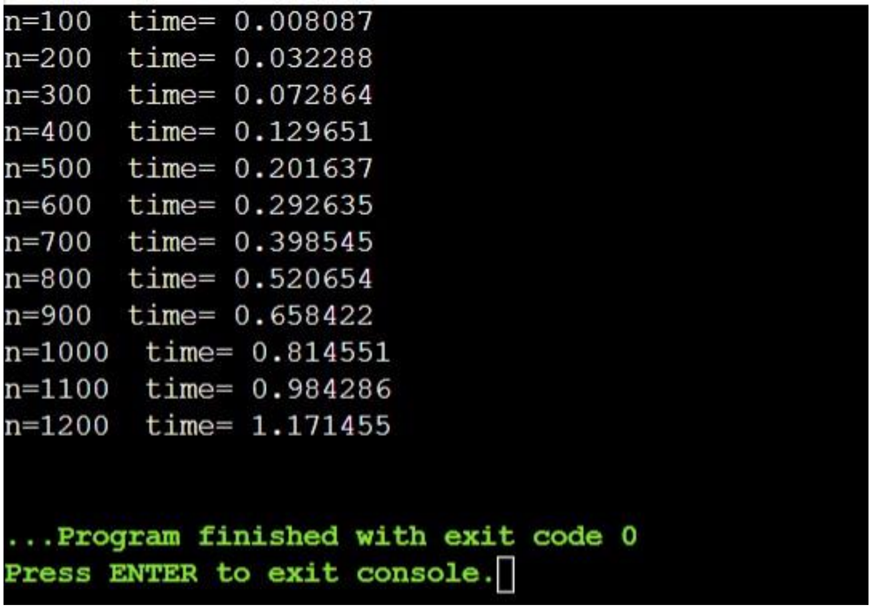


```
enter the no of nodes:  4

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

graph is connected


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

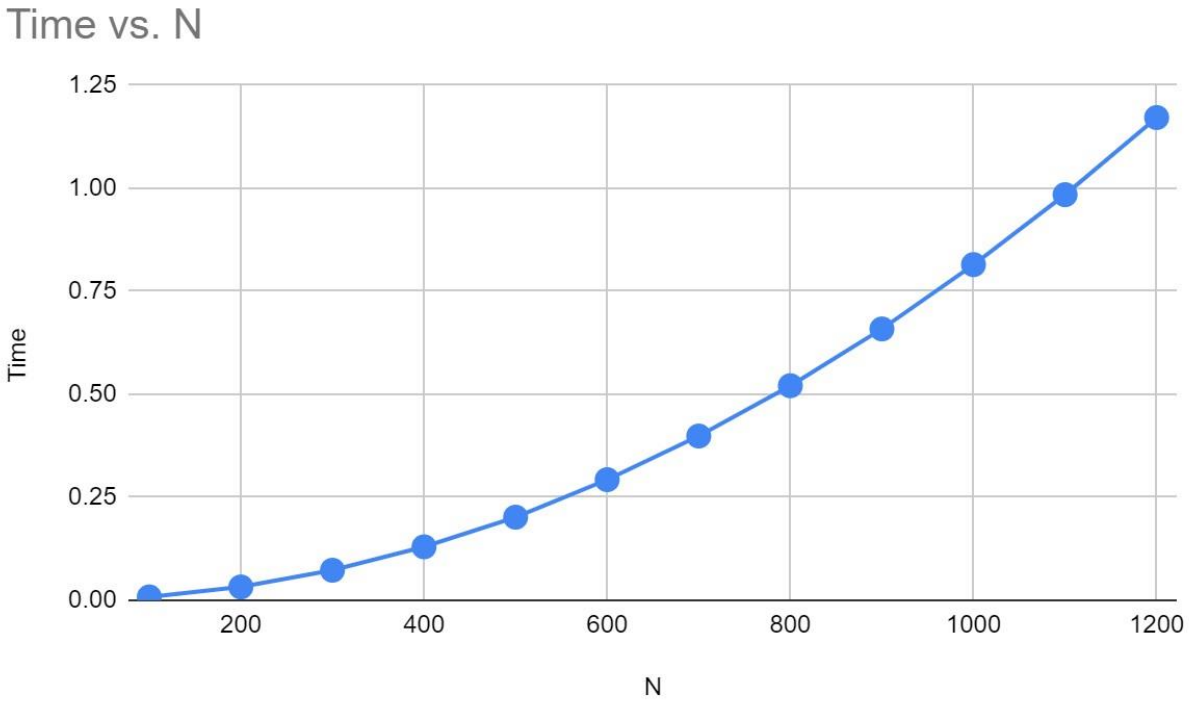
5. Sort a given set of N integer elements using Insertion Sort technique and compute its time taken.

```
#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;
}
```

Result:



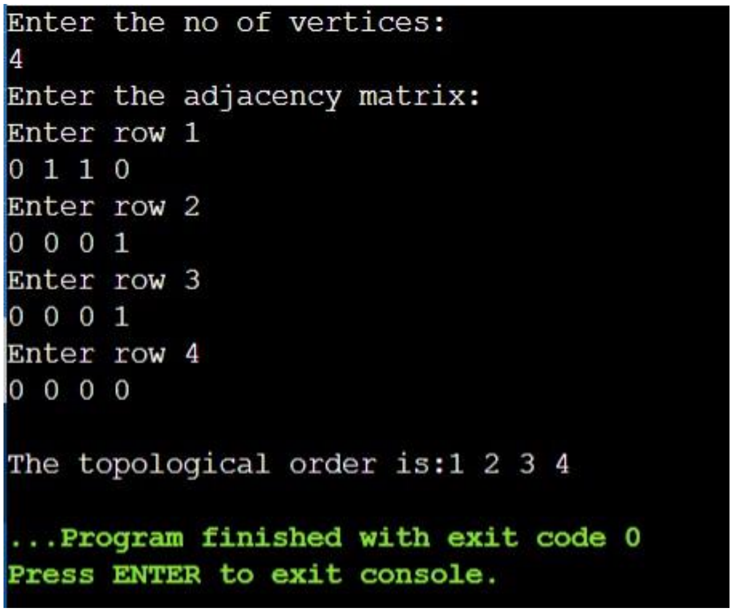
```
n=100   time= 0.008087
n=200   time= 0.032288
n=300   time= 0.072864
n=400   time= 0.129651
n=500   time= 0.201637
n=600   time= 0.292635
n=700   time= 0.398545
n=800   time= 0.520654
n=900   time= 0.658422
n=1000  time= 0.814551
n=1100  time= 0.984286
n=1200  time= 1.171455


...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>


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++;
    }
    }
```

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.
```

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++;
```

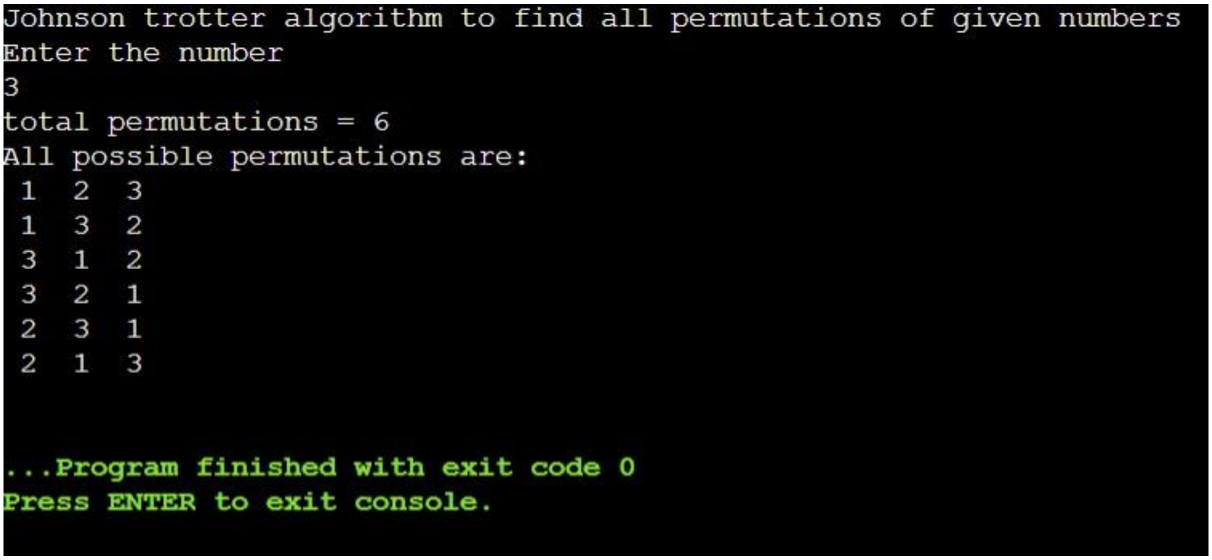
```
        }
    }
    return -1;
}

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("total permutations = %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
total permutations = 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.

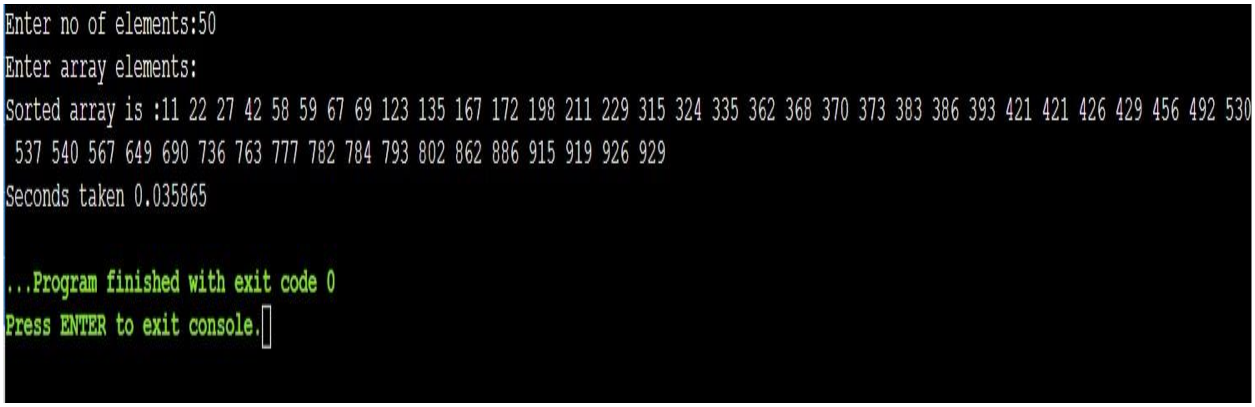
```
#include<stdio.h>
```

```
#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++];
```

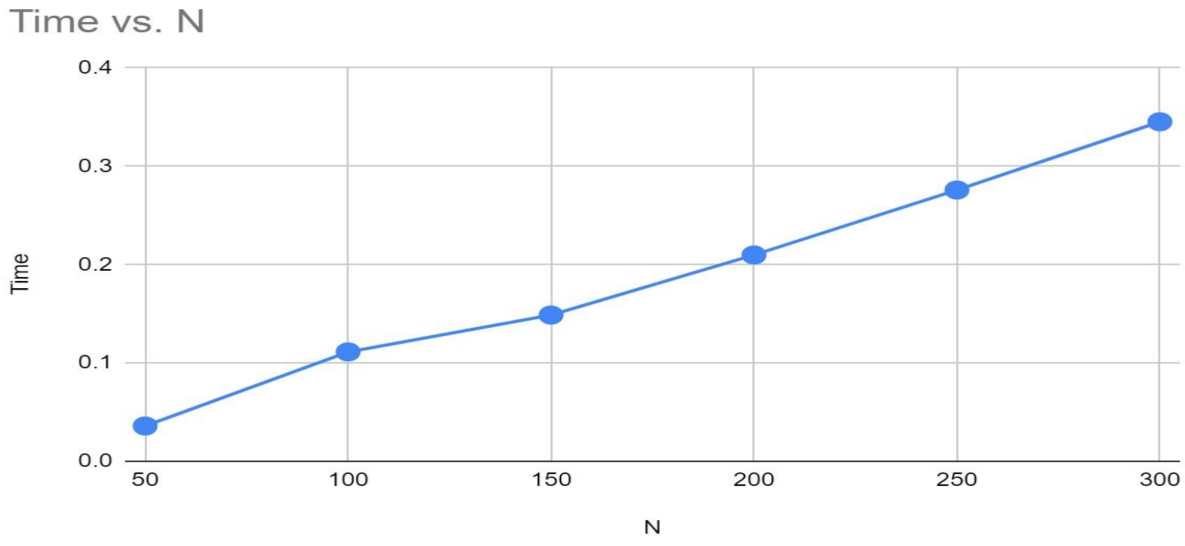
```
for(i=i1,j=0;i<=j2;i++,j++)
a[i]=temp[j];
}
```

Output:



```
Enter no of elements:50
Enter array elements:
Sorted array is :11 22 27 42 58 59 67 69 123 135 167 172 198 211 229 315 324 335 362 368 370 373 383 386 393 421 421 426 429 456 492 530
 537 540 567 649 690 736 763 777 782 784 793 802 862 886 915 919 926 929
Seconds taken 0.035865

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

N	Time
50	0.035865
100	0.111199
150	0.148658
200	0.209777
250	0.275837
300	0.345228

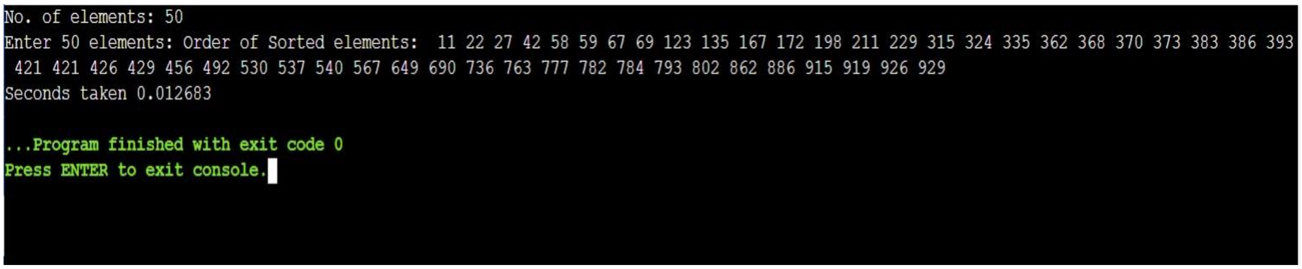


9.Sort a given set of N integer elements using Quick sort technique and compute its time taken.

```
#include<stdio.h>

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

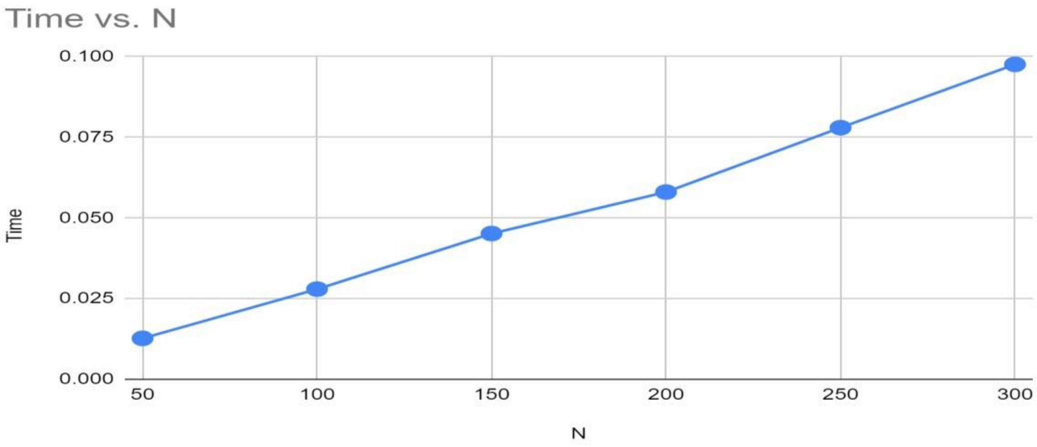
```
void quicksort(int number[5000],int first,int last){
int i, j, pivot, temp; if(first<last){ pivot=first;
i=first; j=last; while(i<j){
for(int x=0;x<100000;x++);
while(number[i]<=number[pivot]&&i<last) i++;
while(number[j]>number[pivot])
j--; if(i<j){
temp=number[i]; number[i]=number[j];
number[j]=temp;
}
}
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:



```
No. of elements: 50
Enter 50 elements: Order of Sorted elements:  11 22 27 42 58 59 67 69 123 135 167 172 198 211 229 315 324 335 362 368 370 373 383 386 393
 421 421 426 429 456 492 530 537 540 567 649 690 736 763 777 782 784 793 802 862 886 915 919 926 929
Seconds taken 0.012683

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

N	Time
50	0.012683
100	0.027947
150	0.045167
200	0.058032
250	0.077984
300	0.097606


Time vs. N
0.100
0.075
0.050
0.025
0.000
Time
50
100
150
200
250
300
N