VISVESVARAYA TECHNOLOGICAL UNIVERSITY

“JnanaSangama”, Belgaum -590014, Karnataka.



LAB REPORT

on

# Analysis and Design of Algorithms

*Submitted by*

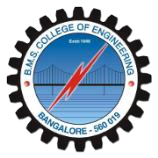
CHERRISHA U SHETTY(1BM20CS033)

*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 CHERRISHA U SHETTY(1BM20CS033), who is bonafide student of B. M. S. College of Engineering. It is in partial fulfilment 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: Vikranth B. M.

Designation Assistant Professor

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 |  |
| 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. |  |
| 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. |  |
| 4 | Write program to do the following:   1. Print all the nodes reachable from a given starting node in a digraph using BFS method. 2. Check whether a given graph is connected or not using DFS method. |  |
| 5 | Sort a given set of N integer elements using Insertion Sort technique and compute its time taken. |  |
| 6 | Write program to obtain the Topological ordering of vertices in a given digraph. |  |
| 7 | Implement Johnson Trotter algorithm to generate permutations. |  |
| 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. |  |
| 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(){

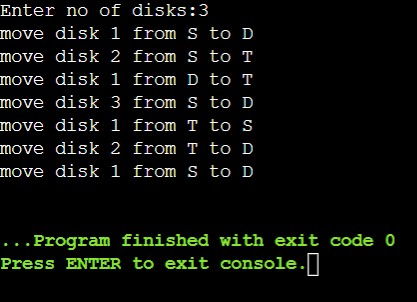
int n;

printf("Enter no of disks:"); scanf("%d",&n);

TOH(n,'S','T','D');

}

Result:



b)

#include<stdio.h>

Int gcd(int a,int b){

If(b!=0)

Return gcd(b,a%b);

Else

Return a;

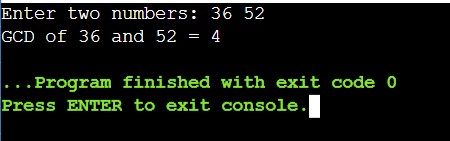
}

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:



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> int n,a[10000];

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);

}

void bub\_sort(int a[],int n)

{ int i,j,temp; for(i=0;i<=n-2;i++)

{

for(j=0;j<=n-2-i;j++)

{

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

{

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

}

}

}

}

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; //Insering numbers in Ascending order

}

key=a[n-1]; //Last element of the aray start=clock();

//bub\_sort(a,n); //Sorting numbers in Ascending order using Bubble sort

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)(endstart))/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 aray 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)(endstart))/CLOCKS\_PER\_SEC)); n=n+1000;

}

break; default: exit(0);

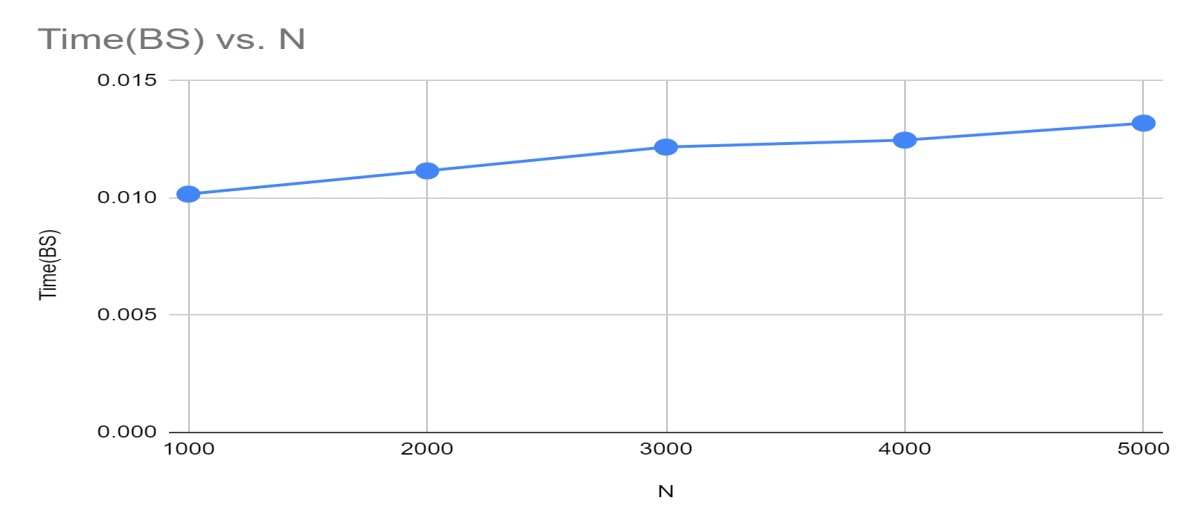
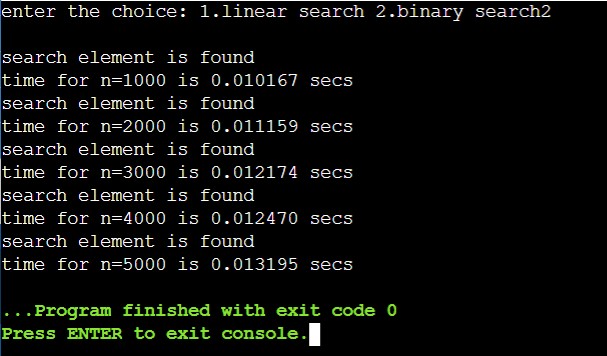
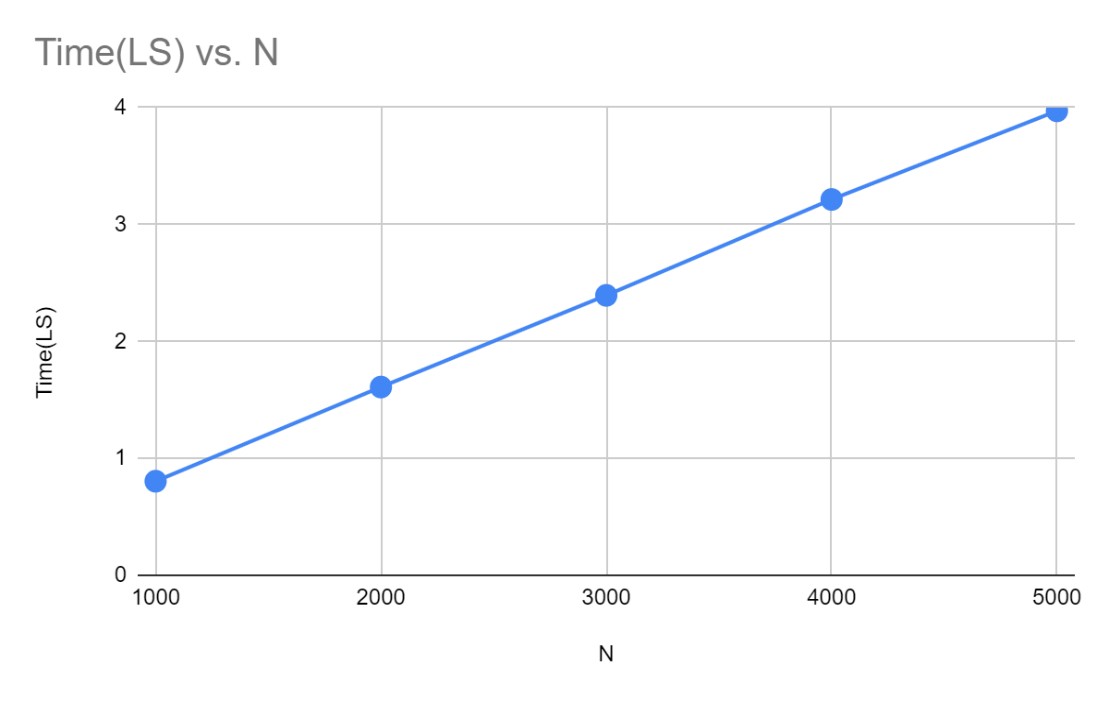
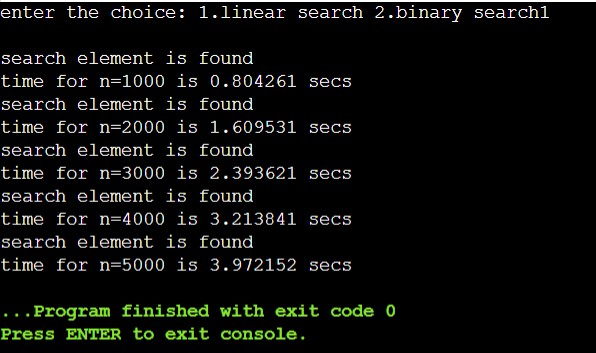
}

getchar();

}

}

Result:



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;

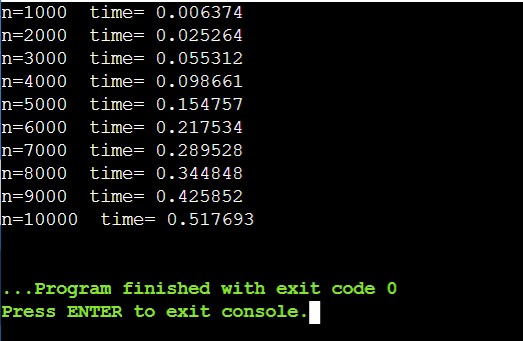
}

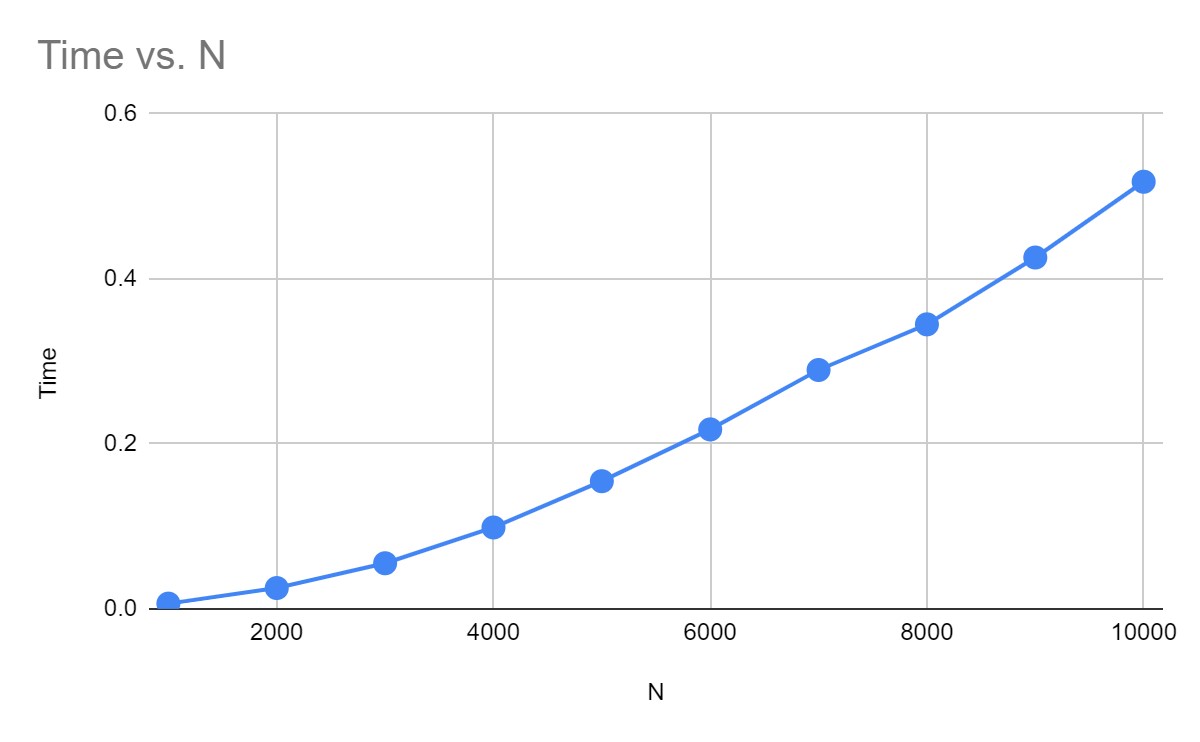
start = clock(); selectionsort(arr,n); end=clock(); printf("n=%d time= %f \n",n,(end-start)/CLOCKS\_PER\_SEC); n=n+1000;

}

}

Result:





4. Write program to do the following:

1. Print all the nodes reachable from a given starting node in a digraph

using BFS method.

1. 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;

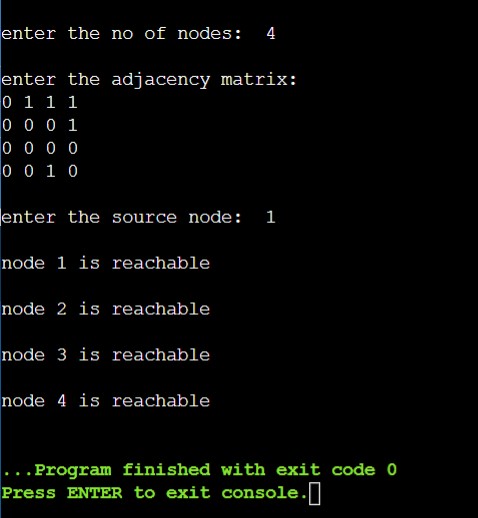
}

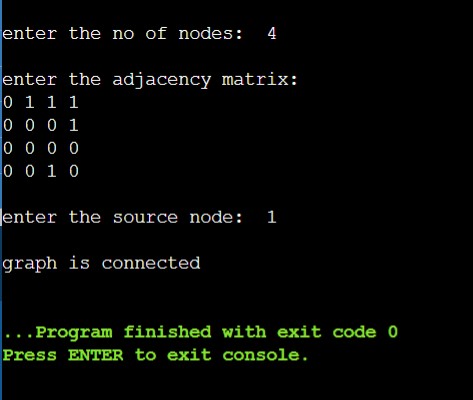
}

return 1;

}

Result:





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;

}

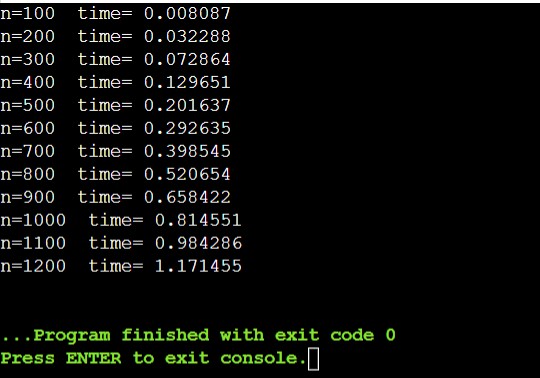
start = clock(); insertionSort(arr, n); end=clock(); printf("n=%d time= %f \n",n,(end-start)/CLOCKS\_PER\_SEC); n=n+100;

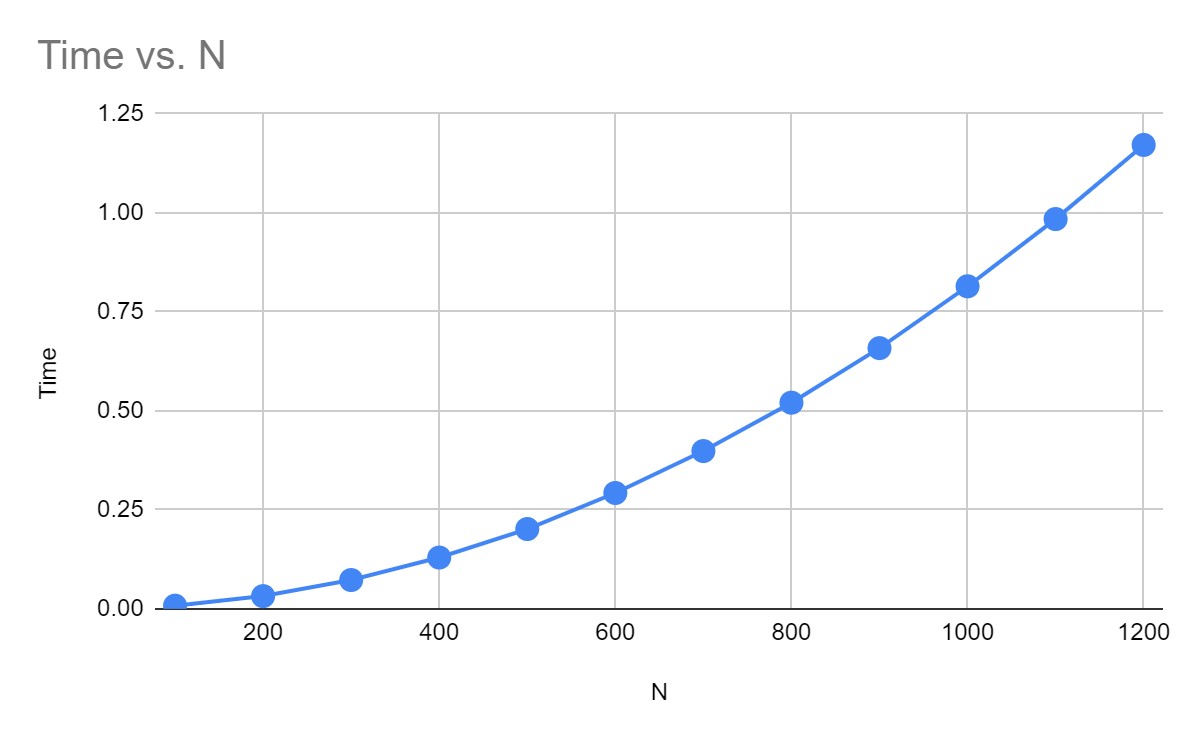
}

return 0;

}

Result:





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]);

}

}

void 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]);

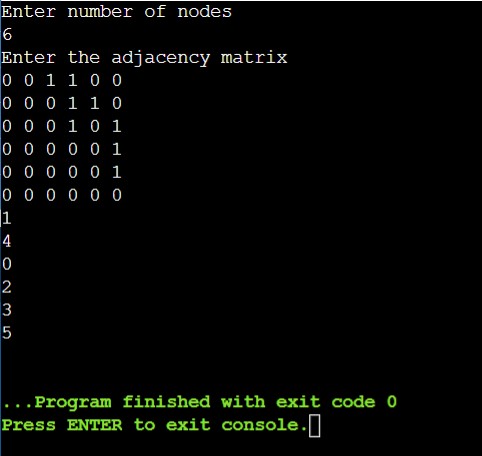
}

}

source\_removal(n,a); getch();

}

Output:



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++)

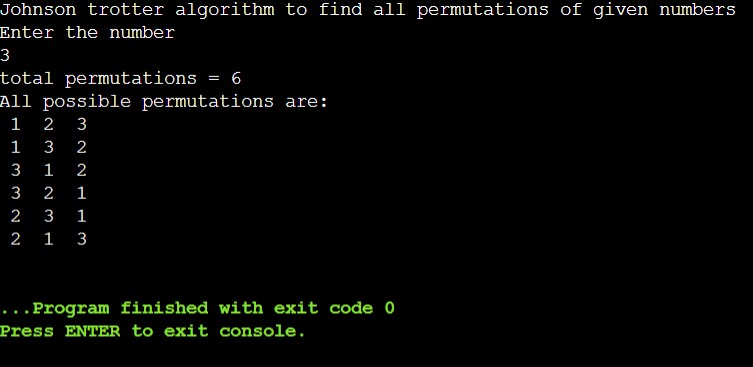
{

permutations(arr,d,num); printf("\n");

} return 0;

}

Output:



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])

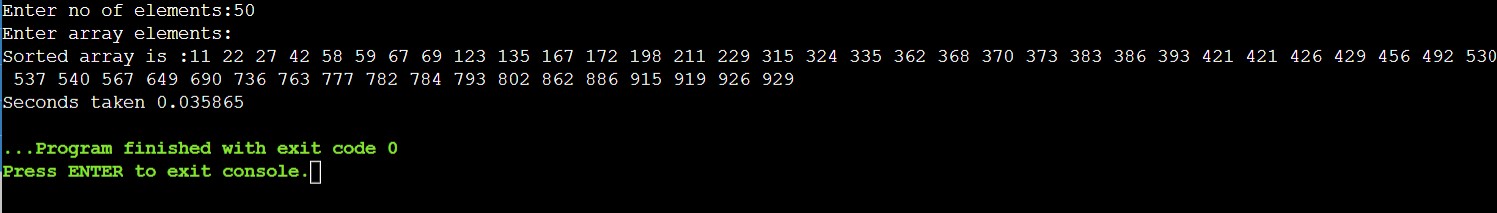
temp[k++]=a[i++]; else temp[k++]=a[j++];

}

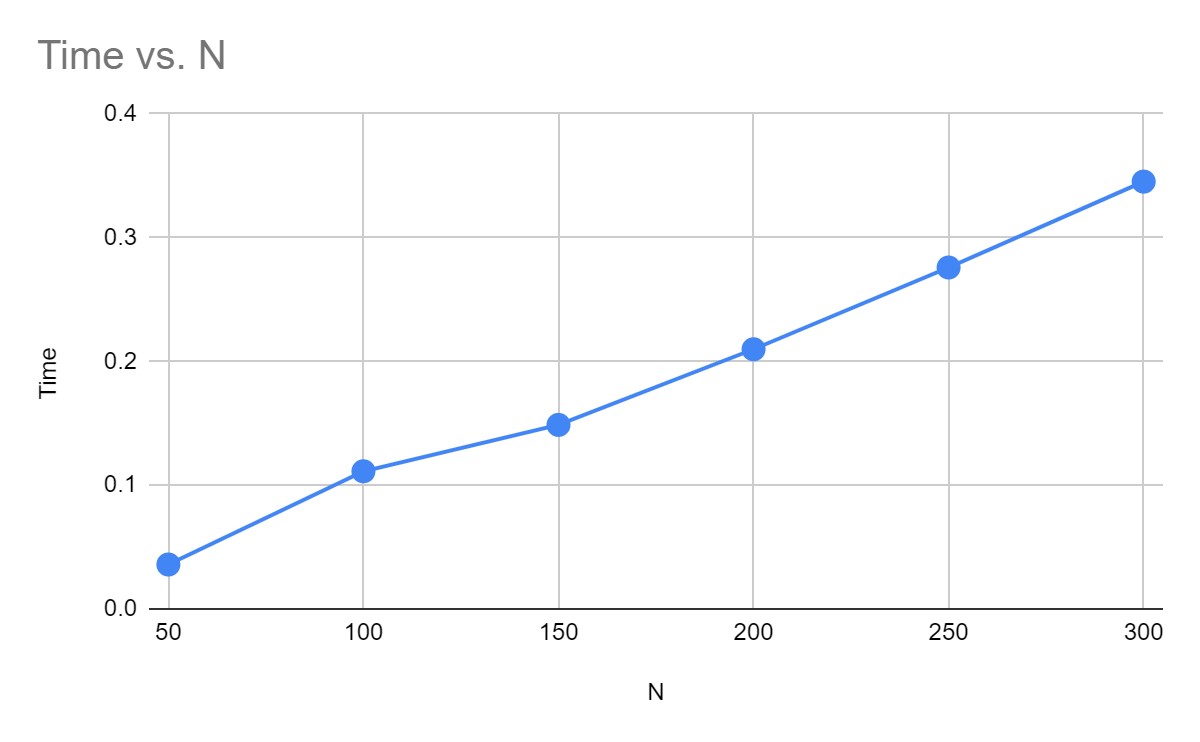
while(i<=j1) temp[k++]=a[i++]; while(j<=j2) temp[k++]=a[j++]; for(i=i1,j=0;i<=j2;i++,j++) a[i]=temp[j];

}

Output:



|  |  |
| --- | --- |
| 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();

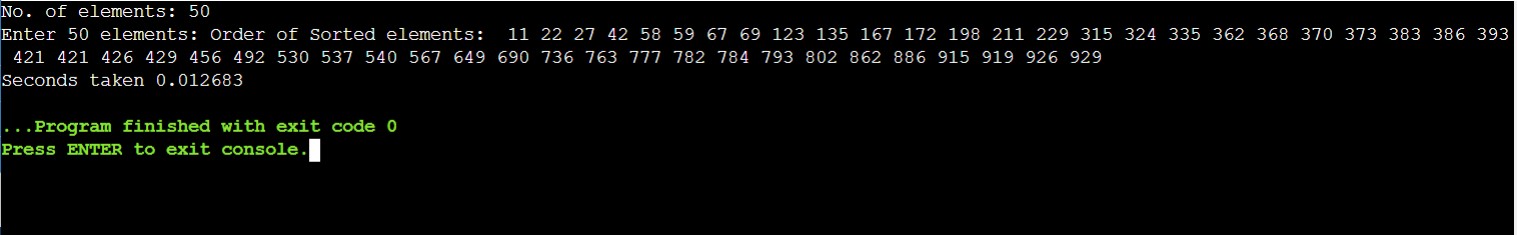
quicksort(number,0,count-1); end = clock();

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:



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

