**VISVESVARAYA TECHNOLOGICAL UNIVERSITY**

**“JnanaSangama”, Belgaum -590014, Karnataka.**

****

**LAB REPORT**

**on**

**Analysis and Design of Algorithms**

***Submitted by***

**PRAJWAL BHAT (1BM20CS107)**

***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 **PRAJWAL BHAT (1BM20CS107),** 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: Prof. Rekha G.S. **Dr. Jyothi S Nayak**

Assistant Professor 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 |  |
| **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:  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. |  |
| **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** | 1. 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

a) PROGRAM:-

#include <stdio.h>

void towers(int, char, char, char);

int main()

{

int num;

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

scanf("%d", &num);

printf("The sequence of moves involved in the Tower of Hanoi are :\n");

towers(num, 'A', 'C', 'B');

return 0;

}

void towers(int num, char frompeg, char topeg, char auxpeg)

{

if (num == 1)

{

printf("\n Move disk 1 from peg %c to peg %c", frompeg, topeg);

return;

}

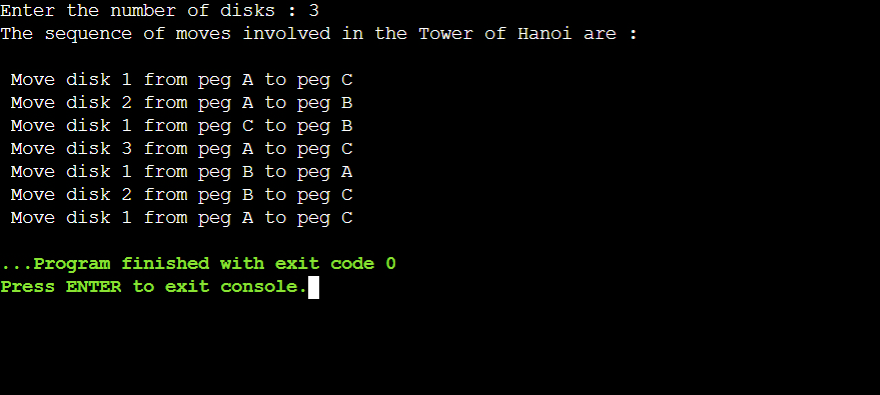
towers(num - 1, frompeg, auxpeg, topeg);

printf("\n Move disk %d from peg %c to peg %c", num, frompeg, topeg);

towers(num - 1, auxpeg, topeg, frompeg);

}

OUTPUT:-



b) PROGRAM:-

#include<stdio.h>

int gcd(int,int);

void main()

{

int m, n;

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

int ans = gcd(m,n);

printf("%d",ans);

}

int gcd(int m,int n)

{

if(n!=0)

{

return gcd(n,m%n);

}

else

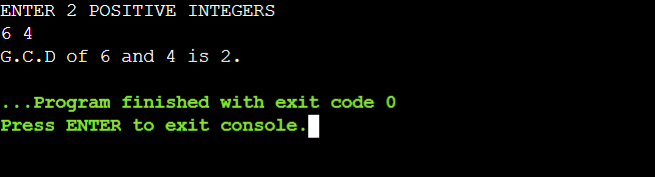
{

return m;

}

}

OUTPUT:-



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.

PROGRAM:-

#include<stdio.h>

#include<time.h>

#include<math.h>

#include<stdlib.h>

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;

long int i, j;

while(1)

{

printf("\nenter the choice: 1.linear search 2.binary search");

scanf("%d",&ch);

switch(ch)

{

case 2:

n=1000;

while(n<=5000)

{

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

{

a[i]=i;

}

key=a[n-1];

start=clock();

search\_status=bin\_srch(a,0,n-1,key);

if(search\_status==-1)

printf("\nKey Not Found");

else

printf("\n seach element is found %d",search\_status);

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 1:

n=1000;

while(n<=5000)

{

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

{

a[i]=i;

}

key=a[n-1];

start=clock();

search\_status=lin\_srch(a,0,n-1,key);

if(search\_status==-1)

printf("\nKey Not Found");

else

printf("\n seach element is found");

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

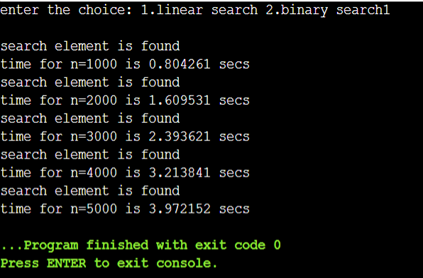
{

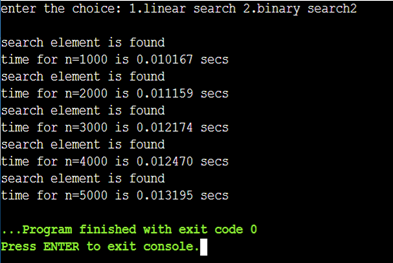
return lin\_srch(a,i+1,high,key);

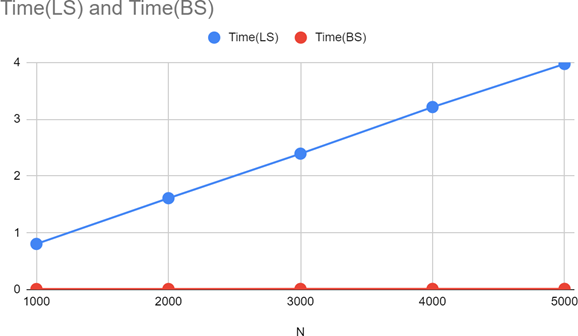
}

}

OUTPUT:-







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

PROGRAM:-

#include <stdio.h>

#include <time.h>

#include <stdlib.h>

#include<math.h>

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");

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 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 ELEMENTS 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 <= 15000)

{

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

{

a[i]=rand()%n;

//a[i] = n - i;

}

start = clock();

selsort(n, a);

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 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];

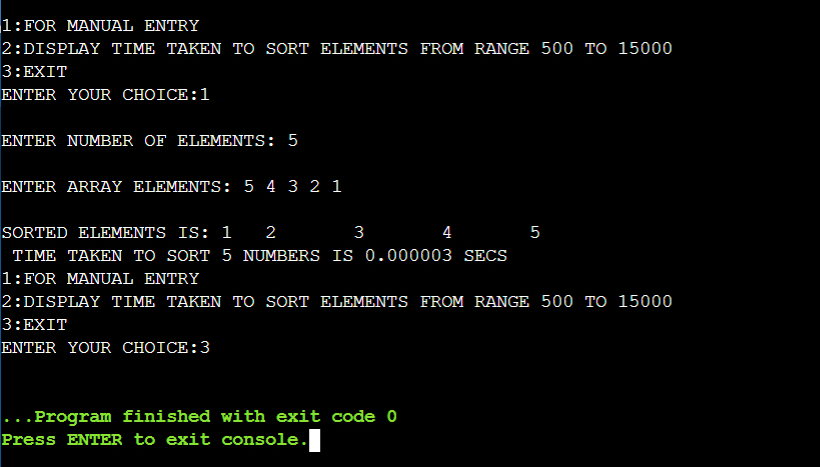
a[i] = a[pos];

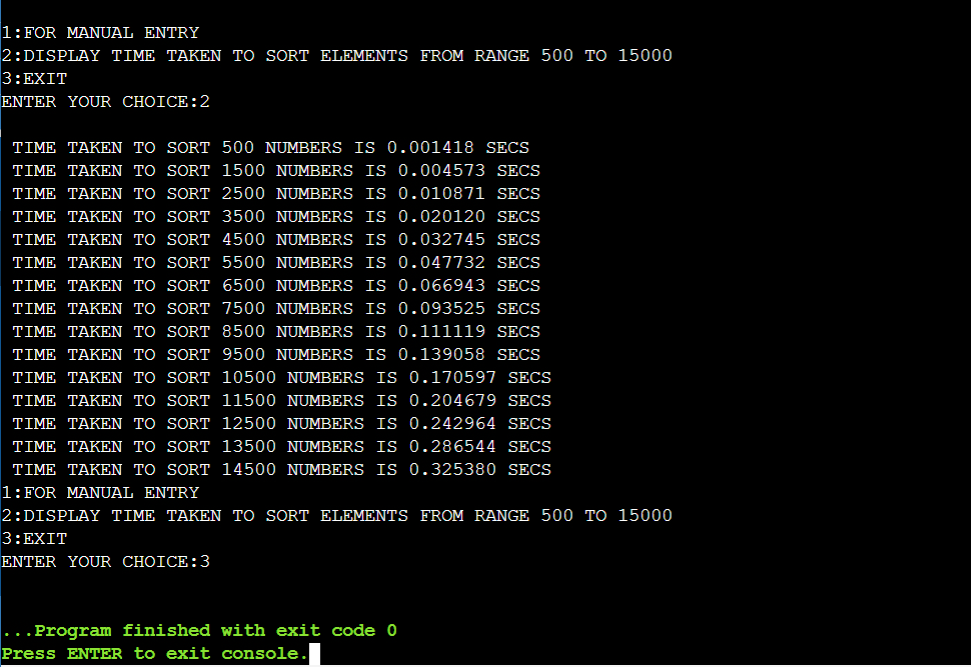
a[pos] = t;

}

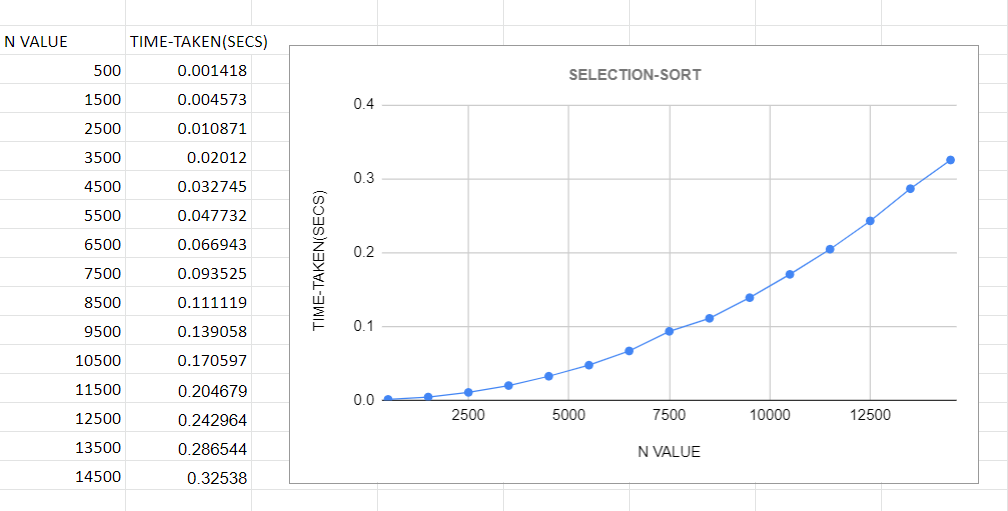
}

OUTPUT:-





GRAPH: Generated graph for N value vs Time taken for Selection sort for different values of N



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)PROGRAM:-

#include <stdio.h>

#include <conio.h>

int a[10][10], n;

void bfs(int);

int main()

{

int i, j, src;

printf("\nENTER NUMBER OF NODES:\t");

scanf("%d", &n);

printf("\nENTER ADJACENCY MATRIX:\n");

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

{

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

{

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

}

}

printf("\nENTER 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

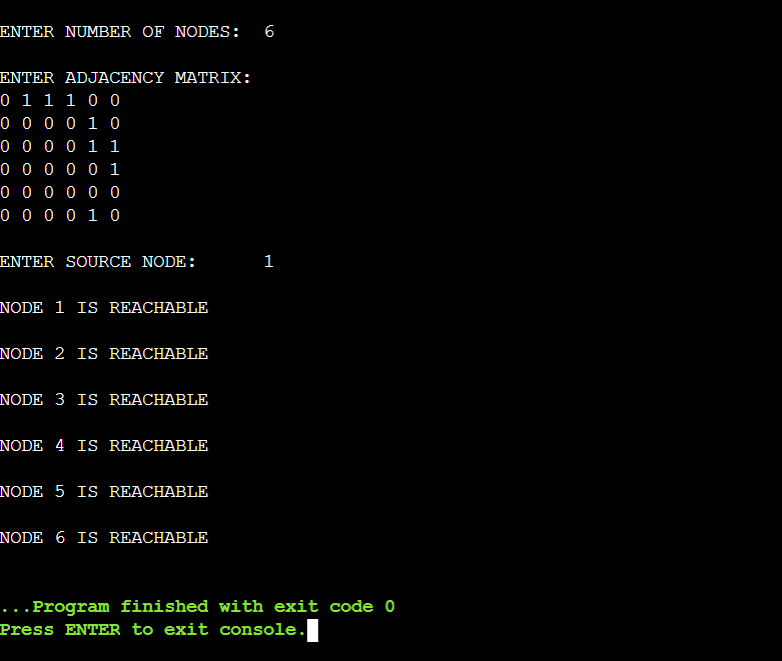
{

printf("\nNODE %d IS REACHABLE\n", j);

}

}

OUTPUT:-



b) PROGRAM:-

#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 NO OF NODES:\t");

scanf("%d",&n);

printf("\nENTER ADJACENCY MATRIX:\n");

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

{

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

{

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

}

}

printf("\nENTER SOURCE NODE:");

scanf("%d",&src);

ans=dfs(src);

if(ans==1)

{

printf("\nGRAPH IS CONNECTED\n");

}

else

{

printf("\nGRAPH IS NOT CONNECTED\n");

}

}

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;

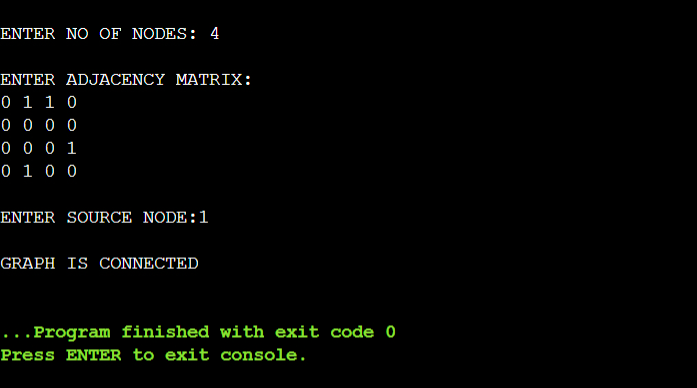
}

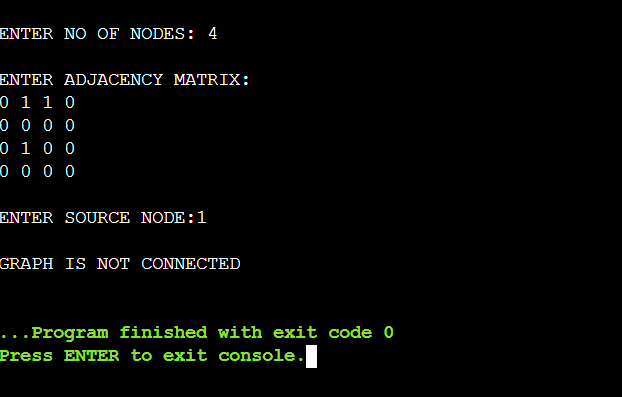
}

return 1;

}

OUTPUT:-





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

PROGRAM:-

#include <math.h>

#include <stdio.h>

#include<stdlib.h>

#include<time.h>

void insertionSort(int arr[], int n)

{

int i, key, j;

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

{

key = arr[i];

j = i - 1;

while (j >= 0 && arr[j] > key)

{

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

j = j - 1;

}

arr[j + 1] = key;

}

}

int main()

{

int i, n;

clock\_t start, end;

printf("ENTER ARRAY SIZE =");

scanf("%d", &n);

int arr[150000];

printf("ENTER ARRAY ELEMENTS");

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

{

arr[j] = rand() % 10000;

}

printf("\n");

start = clock();

insertionSort(arr, n);

end = clock();

printf("\nSORTED ELEMNETS = ");

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

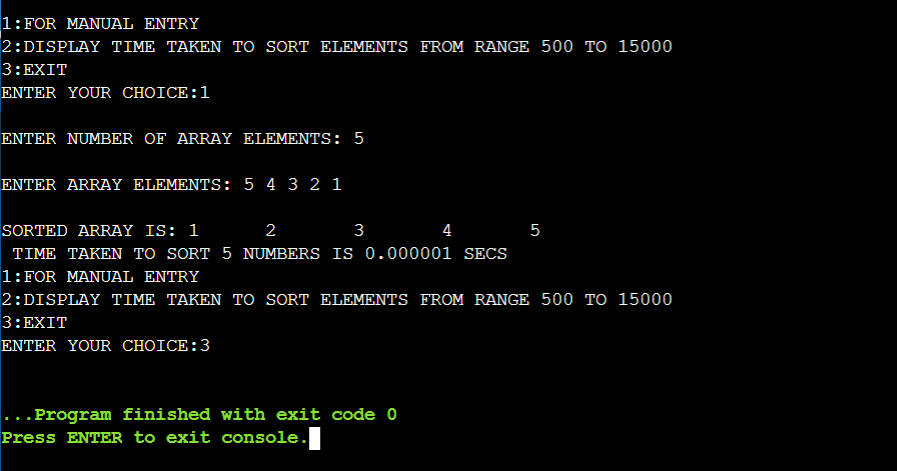
printf(" %d", arr[i]);

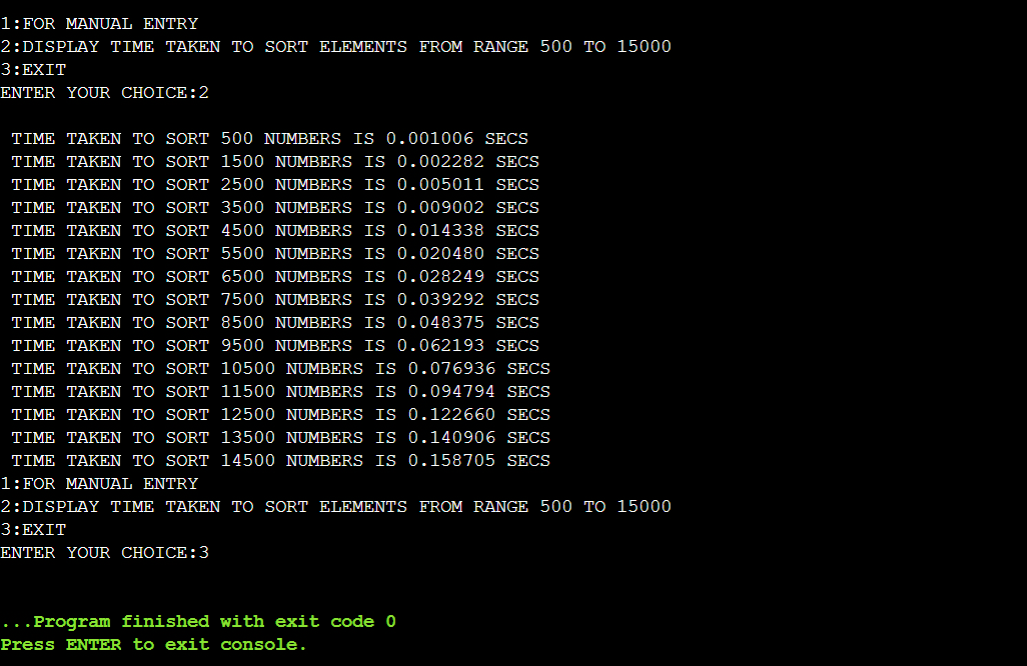
printf("\n TIME TAKEN TO SORT %d NUMBERS IS %f SECS", n, (((double)(end - start)) / CLOCKS\_PER\_SEC));

return 0;

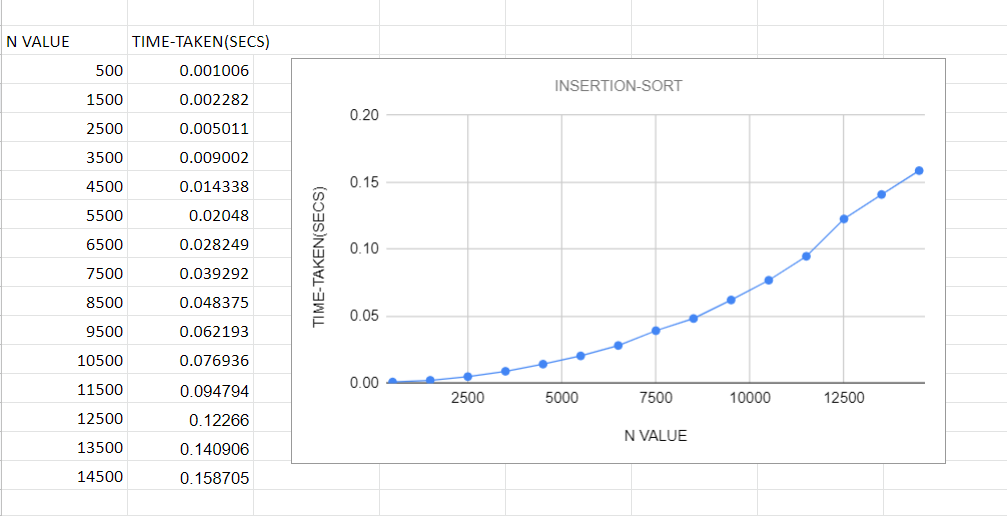
}

OUTPUT:-





GRAPH: Generated graph for N value vs Time taken for Selection sort for different values of N



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

PROGRAM:-

#include<stdio.h>

#include<conio.h>

void source\_removal(int n, int a[10][10])

{

int i,j,k,u,v,top;

int 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]);

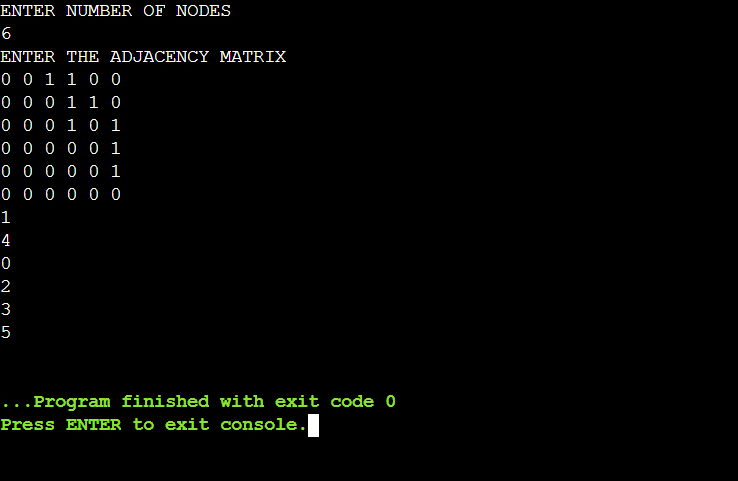
}

}

source\_removal(n,a);

}

OUTPUT:-



7) Implement Johnson Trotter algorithm to generate permutations.

PROGRAM:-

#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);

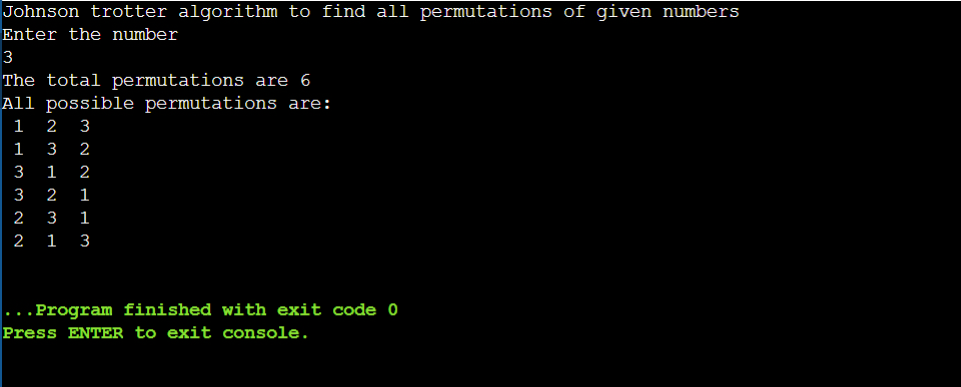
printf("\n");

}

return 0;

}

OUTPUT:-



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.

PROGRAM:-

#include <stdio.h>

#include <time.h>

#include <stdlib.h>

#include <math.h>

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

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

split(a, 0, n - 1);

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]=rand()%n;

//a[i] = n - i;

}

start = clock();

split(a, 0, n - 1);

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 split(int a[], int low, int high)

{

int mid;

if (low < high)

{

mid = (low + high) / 2;

split(a, low, mid);

split(a, mid + 1, high);

combine(a, low, mid, high);

}

}

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

{

int c[15000], i, j, k;

i = k = low;

j = mid + 1;

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

{

if (a[i] < a[j])

{

c[k] = a[i];

++k;

++i;

}

else

{

c[k] = a[j];

++k;

++j;

}

}

if (i > mid)

{

while (j <= high)

{

c[k] = a[j];

++k;

++j;

}

}

if (j > high)

{

while (i <= mid)

{

c[k] = a[i];

++k;

++i;

}

}

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

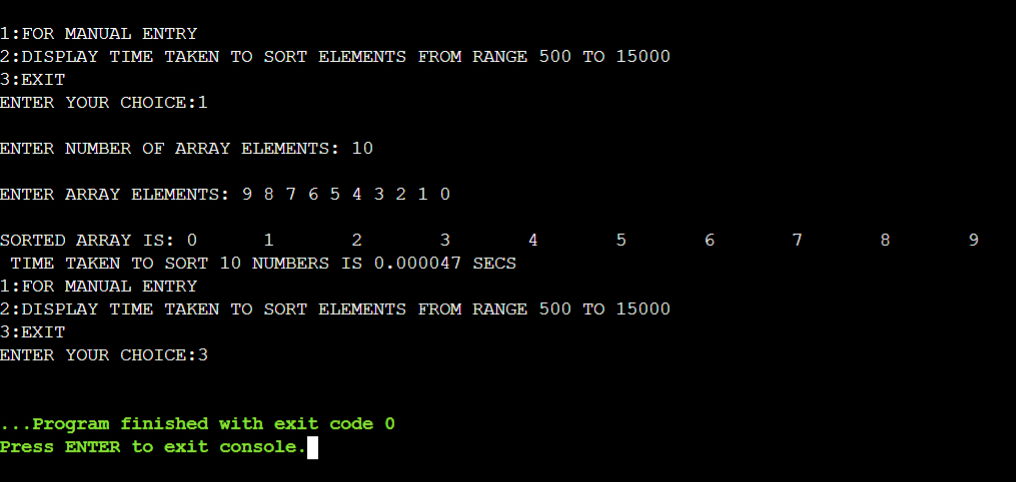
{

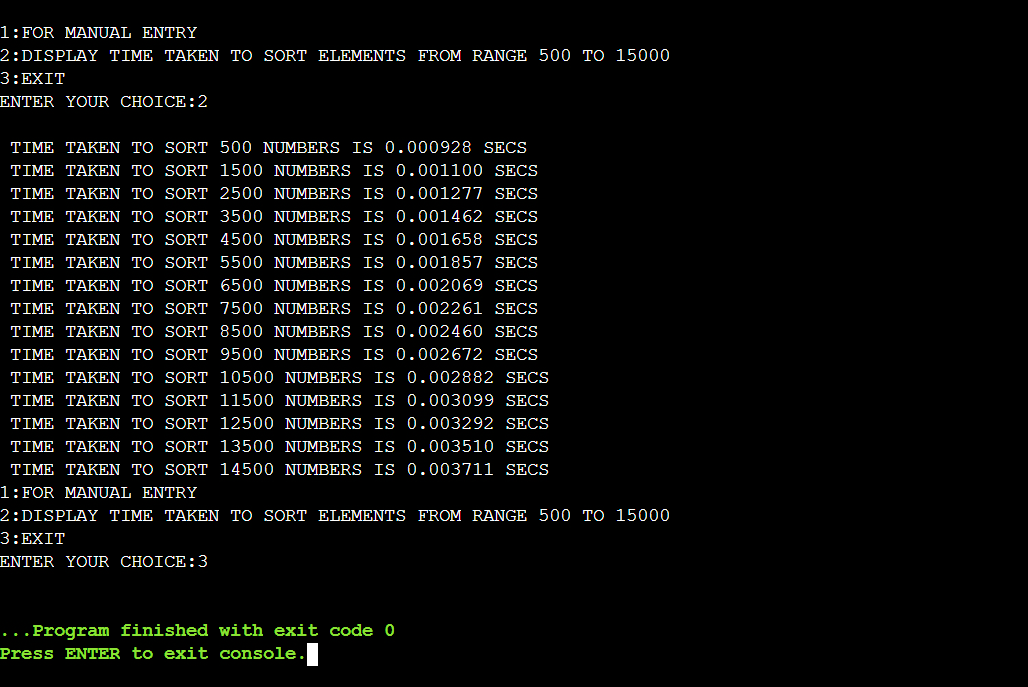
a[i] = c[i];

}

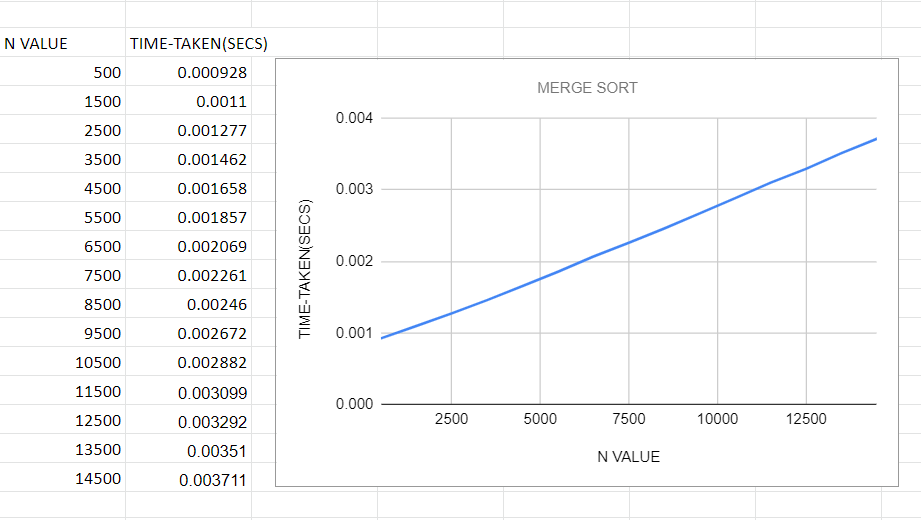
}

OUTPUT:-





GRAPH: Generated graph for N value vs Time taken for Merge sort for different values of N



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

PROGRAM:

#include <stdlib.h>

#include <stdio.h>

#include <time.h>

#include <math.h>

void quicksort(int arr[25], int first, int last)

{

int i, j, pivot, temp;

if (first < last)

{

pivot = first;

i = first;

j = last;

while (i < j)

{

while (arr[i] <= arr[pivot] && i < last)

i++;

while (arr[j] > arr[pivot])

j--;

if (i < j)

{

temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

}

}

temp = arr[pivot];

arr[pivot] = arr[j];

arr[j] = temp;

for (int p = 0; p < 1000000; p++)

;

quicksort(arr, first, j - 1);

quicksort(arr, j + 1, last);

}

}

int main()

{

int i, n;

clock\_t start, end;

printf("ENTER ARRAY SIZE =");

scanf("%d", &n);

int arr[150000];

printf("ENTER ARRAY ELEMENTS");

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

{

arr[j] = rand() % 10000;

}

printf("\n");

start = clock();

quicksort(arr, 0, n - 1);

end = clock();

printf("\nSORTED ELEMNETS = ");

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

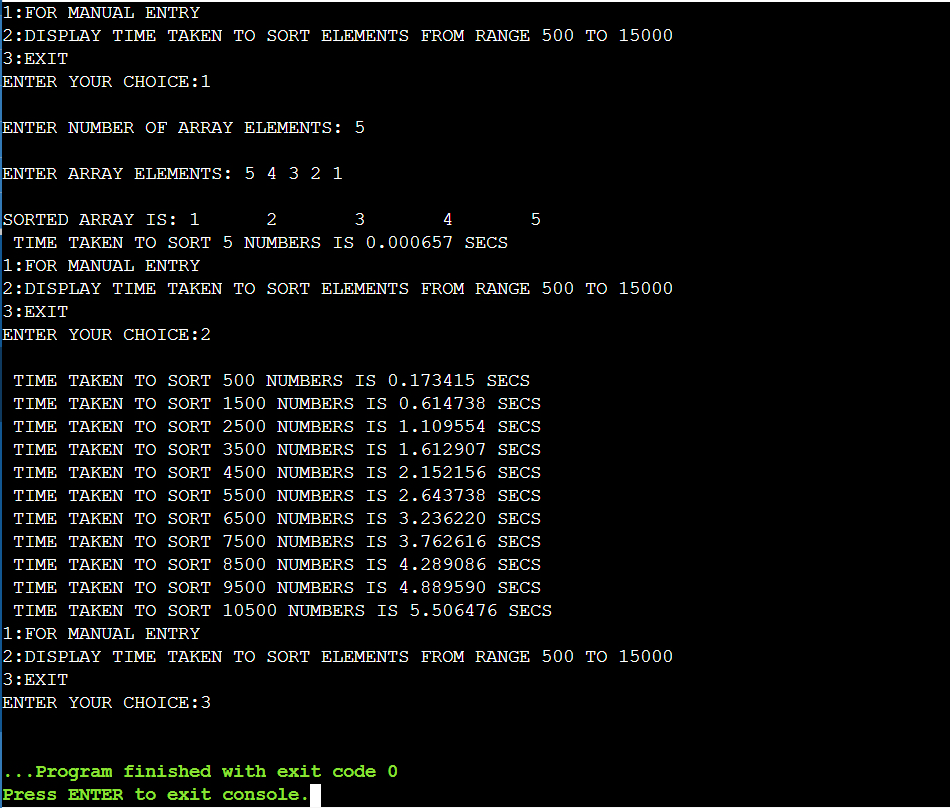
printf(" %d", arr[i]);

printf("\n TIME TAKEN TO SORT %d NUMBERS IS %f SECS", n, (((double)(end - start)) / CLOCKS\_PER\_SEC));

return 0;

}

OUTPUT:-



GRAPH: Generated graph for N value vs Time taken for Quick sort for different values of N

