VISVESVARAYATECHNOLOGICALUNIVERSITY

“JnanaSangama”, Belgaum -590014, Karnataka.



LAB REPORT on

Analysis and Design of Algorithms

*Submitted by*

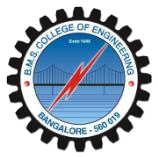
K ADITYA ARVIND (1BM20CS065)

*in partial fulfillment for the award of the degree of*

# BACHELOROFENGINEERING

*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 K ADITYA ARVIND (1BM20CS065), 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.

|  |  |
| --- | --- |
| Nagarathna N | Dr. Jyothi S Nayak |
| 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 | 5-6 |
| 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. | 7-10 |
| 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. | 11-13 |
| 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. | 14-17 |
| 5 | Sort a given set of N integer elements using Insertion Sort technique and compute its time taken. | 18-20 |
| 6 | Write program to obtain the Topological ordering of vertices in a given digraph. | 21-22 |
| 7 | Implement Johnson Trotter algorithm to generate permutations. | 23-26 |
| 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. | 27-29 |
| 9 | Sort a given set of N integer elements using Quick Sort technique and compute its time taken. | 30-32 |
| 10 | Sort a given set of N integer elements using Heap Sort technique and compute its time taken. | 33-36 |
| 11 | Implement Warshall’s algorithm using dynamic programming | 37-38 |
| 12 | Implement 0/1 Knapsack problem using dynamic programming. | 39-41 |
| 13 | Implement All Pair Shortest paths problem using Floyd’s algorithm. | 42-43 |
| 14 | Find Minimum Cost Spanning Tree of a given undirected graph using Prim’s algorithm. | 44-46 |
| 15 | Find Minimum Cost Spanning Tree of a given undirected graph using Kruskals algorithm. | 47-49 |
| 16 | From a given vertex in a weighted connected graph, find shortest paths to other vertices using Dijkstra’s algorithm. | 50-52 |
| 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 | 53-55 |
|  | 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. | 56-58 |

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

CODE:

a) TOWER OF HANOI

#include <stdio.h> void hanoi(int n, char a,char b,char c) { if(n==1) printf("move from %c to %c\n",a,c);

else {

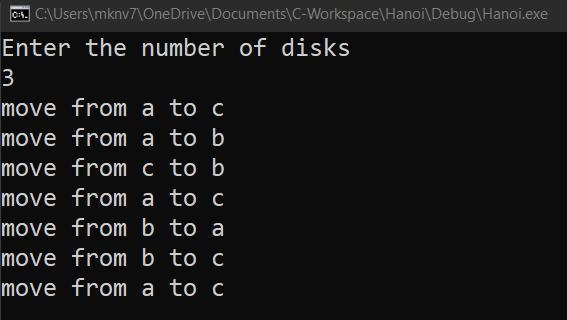
hanoi(n-1,a,c,b); printf("move from %c to %c\n",a,c); hanoi(n-1,b,a,c); }

}

void main() { int n; int moves; printf("Enter the number of disks\n"); scanf("%d",&n);

hanoi(n,'a','b','c'); }

OUTPUT:



b) GREATEST COMMON DIVISOR CODE:

#include <stdio.h>

int gcd(int a,int b) { if(b!=0) return gcd(b,a%b);

else

return a;

}

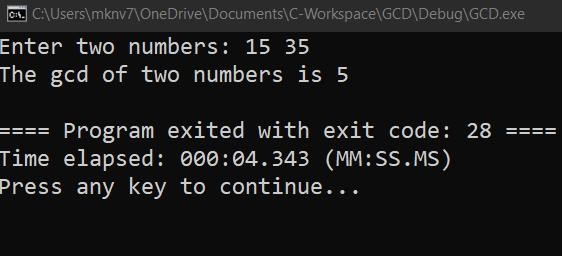
void main()

{ int a,b,c; printf("Enter two numbers: "); scanf("%d%d",&a,&b);

c=gcd(a,b); printf("The gcd of two numbers is %d\n",c);

}

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.

CODE:

#include<stdio.h>

#include<time.h> #include<stdlib.h> int bin\_srch(int [],int,int,int);

int lin\_srch(int [],int,int,int); int n,a[1000000];

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<=7000) 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 Key found at position %d",search\_status);

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<=7000) {

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 Key found at position %d",search\_status);

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) {

for(int j=0;j<1000000;j++){ int temp=38/600;}

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) {

for(int j=0;j<10000;j++){ int temp=38/600;}

if(i>high) return -1;

if(key==a[i])

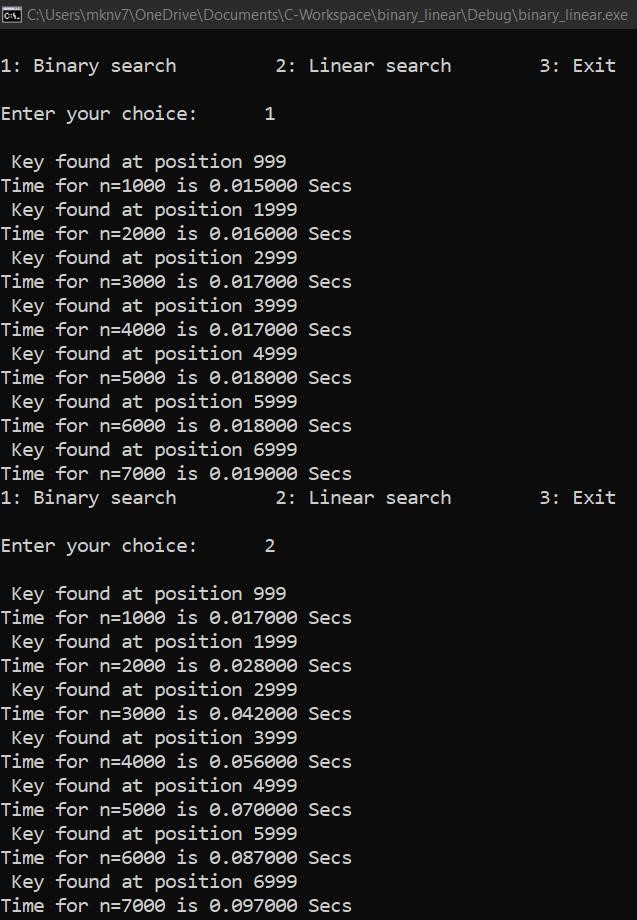
return i;

else

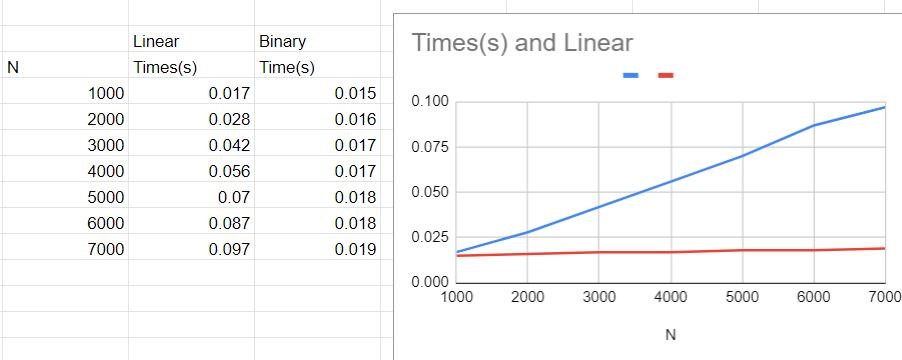
return lin\_srch(a,i+1,high,key);

}

OUTPUT:



GRAPH:



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.

CODE:

#include<stdio.h>

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

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;

}

}

void main()

{ int a[15000],n,i,j,ch,temp; clock\_t start,end;

while(1) {

printf("\n1:To display time taken for sorting number of elements N

in the range 500 to 14500"); printf("\n2:To exit"); printf("\nEnter your choice:"); scanf("%d", &ch);

switch(ch)

{

case 1:

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

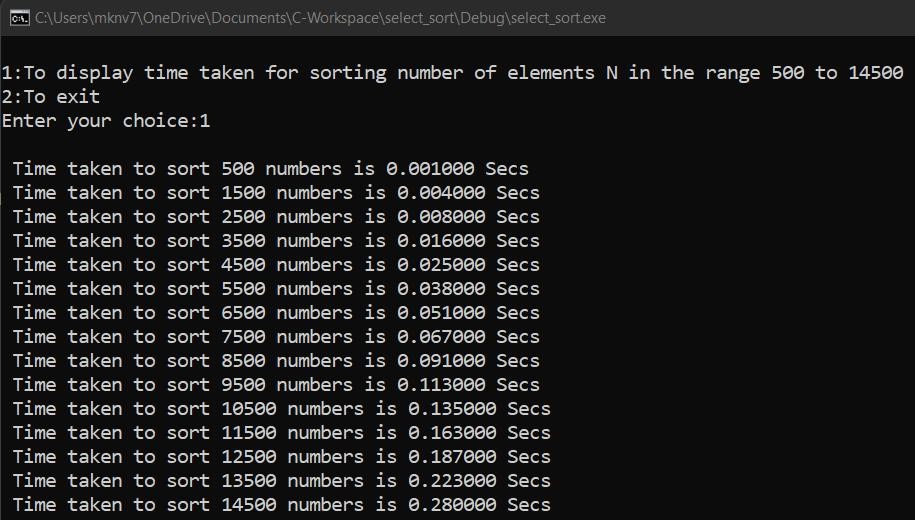
case 2: exit(0);

} getchar();

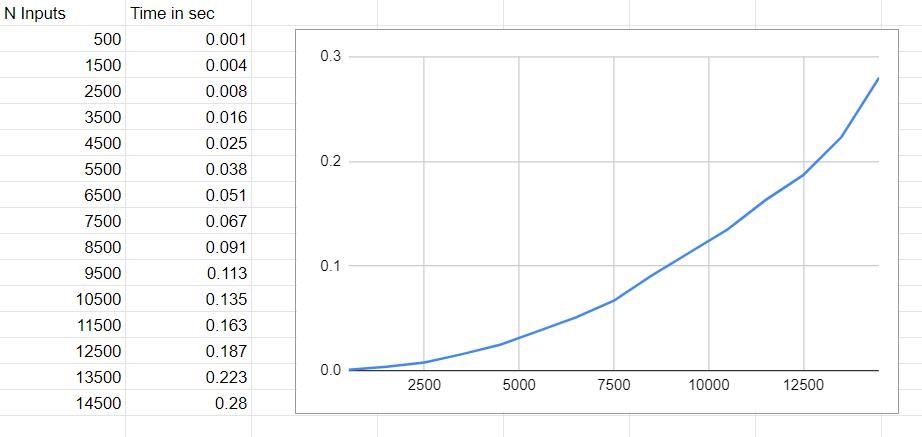
}

}

OUTPUT:



GRAPH:



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.

a) BREADTH FIRST SEARCH CODE:

#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)

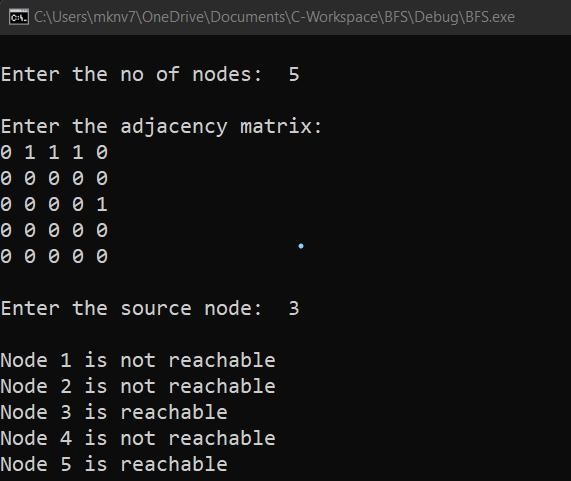
printf("\nNode %d is not reachable",j);

else printf("\nNode %d is reachable",j);

}

}

OUTPUT:



b)DEPTH FIRST SEARCH CODE:

#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++) {

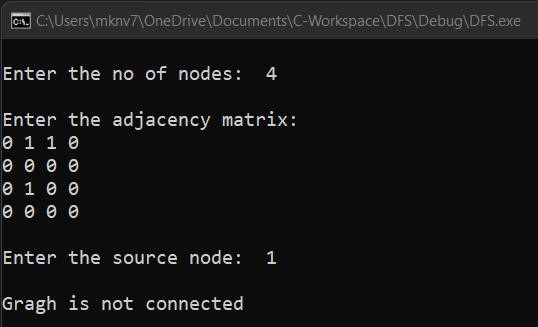
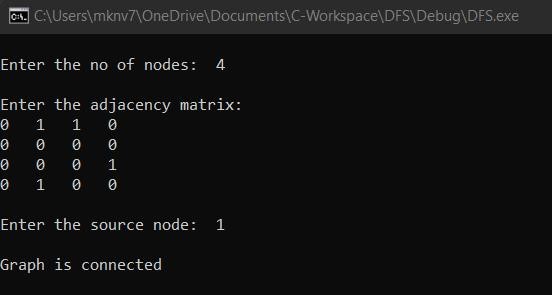
if(vis[j]!=1) return 0;

}

return 1;

}

OUTPUT:



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

CODE:

#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)

{

for(int k=0;k<100000;k++); 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;

}

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

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

} printf("\n"); start = clock(); insertionSort(arr, n); end = clock();

printf("\nSORTED ELEMNETS = "); for (i = 0; i < n; i++) {

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

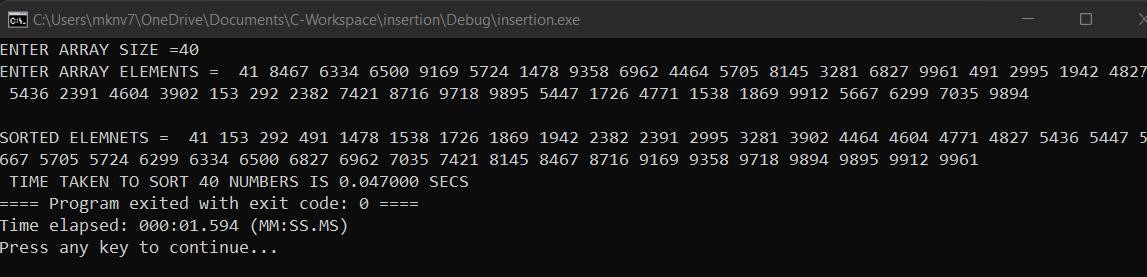
}

printf("\n TIME TAKEN TO SORT %d NUMBERS IS %f SECS", n,

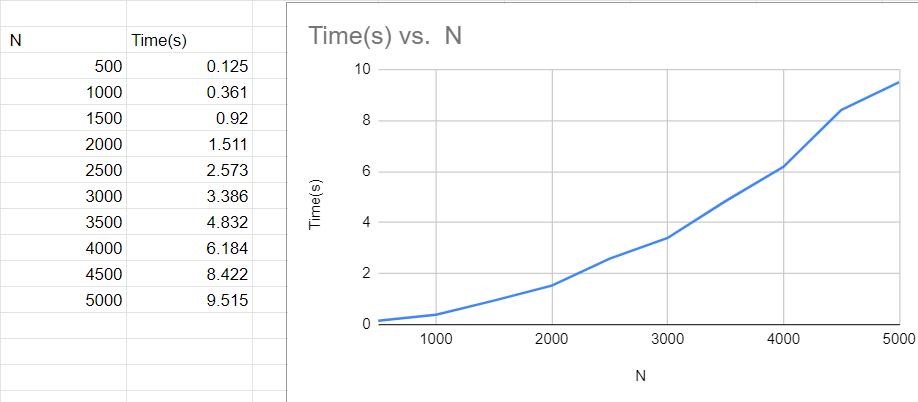
(((double)(end - start)) / CLOCKS\_PER\_SEC)); return 0;

}

OUTPUT:



GRAPH:



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

CODE:

#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;

}

}

}

printf("Topological order :"); for(i=0;i<n;i++)

printf(" %d", 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");

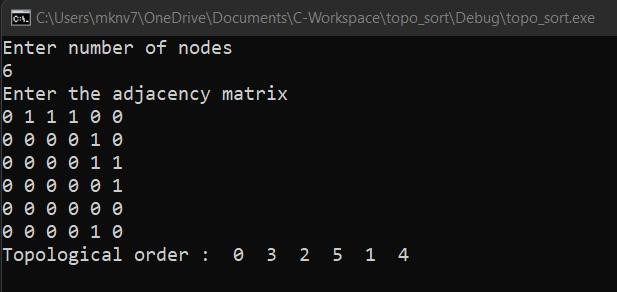
for(i=0;i<n;i++)

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.

CODE:

#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]);

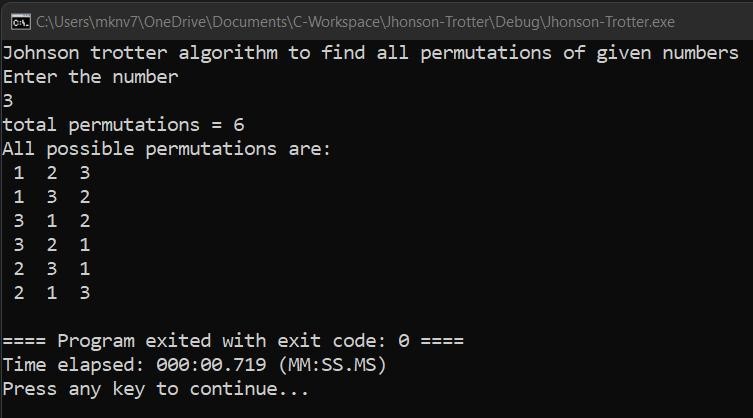
} printf("\n"); for(j=1;j<z;j++) { permutations(arr,d,num); printf("\n");

}

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.

CODE:

#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[30000],n=500,i; while(n<=5000){ for(i=0;i<n;i++)

{

a[i] = rand()%1000;

} start = clock(); mergesort(a,0,n-1); end = clock();

printf("\nSorted array of %d numbers = ",n);

printf("Seconds taken %lf",(double)(end-start)/CLOCKS\_PER\_SEC); printf(“\n”);

n+=500;

}

}

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[30000];

int i,j,k; i=i1; j=i2; k=0; while(i<=j1 && j<=j2)

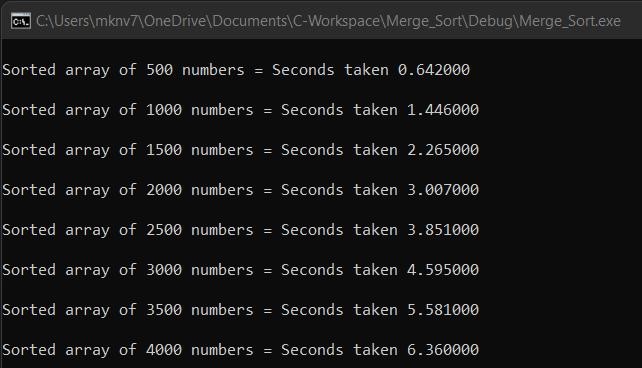
{ for(int j=0;j<100000;j++);

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

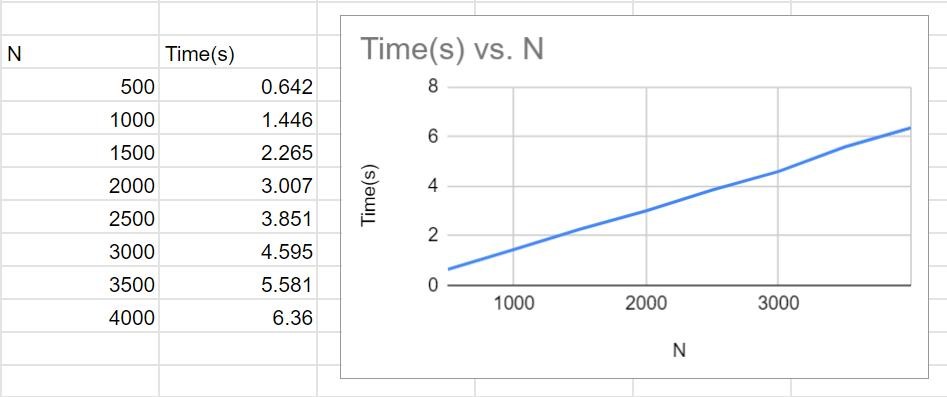
temp[k++]=a[i++]; else temp[k++]=a[j++];

} 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:



GRAPH:



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

CODE:

#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<10000000;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++)\

{

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

} start = clock();

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

printf("Order of Sorted elements: ");

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

{

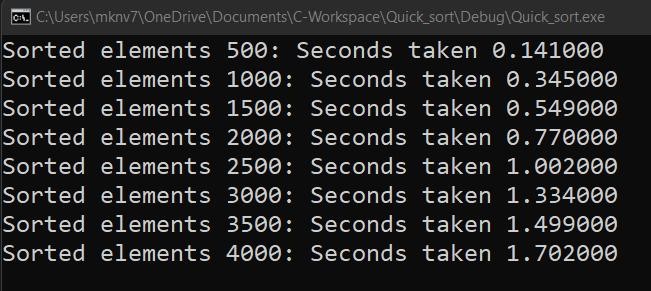
printf(" %d",number[i]);

}

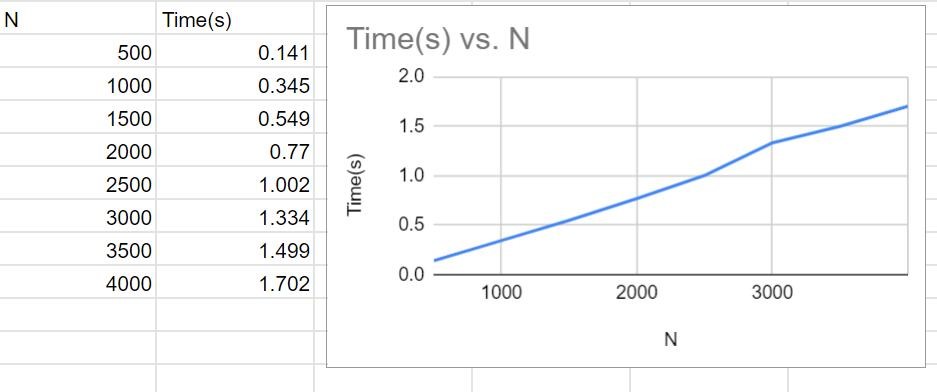
printf("\nSeconds taken %lf",(double)(end-start)/CLOCKS\_PER\_SEC); return 0;

}

OUTPUT:



GRAPH:



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

CODE:

#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; clock\_t start, end;

while (1)

{

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

printf("\n2:DISPLAY TIME TAKEN TO SORT ELEMENTS FROM

RANGE 500 TO 5500"); printf("\n3:EXIT"); printf("\nENTER YOUR CHOICE:"); scanf("%d", &ch);

switch (ch)

{

case 1:

printf("\nENTER NUMBER OF ARRAY ELEMENTS: ");

scanf("%d", &n);

printf("ENTER 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 <= 5500)

{

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

{

//a[i]=rand()%n; a[i] = n - i;

} start = clock(); heapSort(a, n);

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 temp;

for (int j = 0; j < 50000; j++)

temp = 38 / 600;

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);

}

}

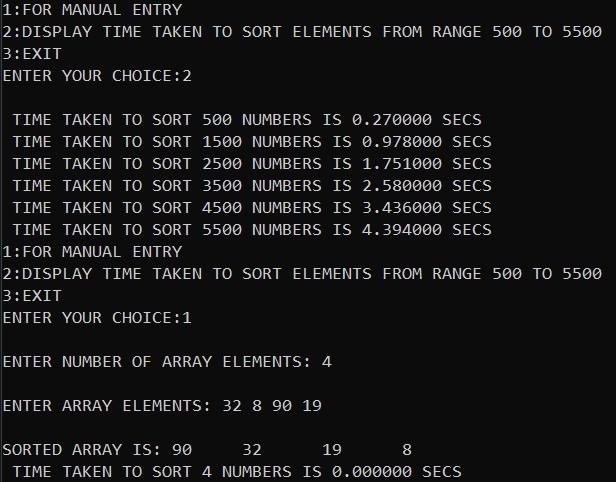
void heapSort(int arr[], int n)

{

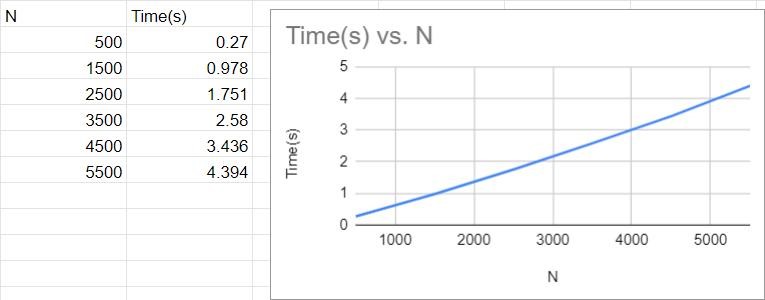
for (int i = n / 2 - 1; i >= 0; i--) heapify(arr, n, i);

for (int i = n - 1; i >= 0; i--)

{ swap(&arr[0], &arr[i]); heapify(arr, i, 0); } } OUTPUT:



GRAPH:



11. Implement Warshall’s algorithm using dynamic programming

CODE:

#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++)

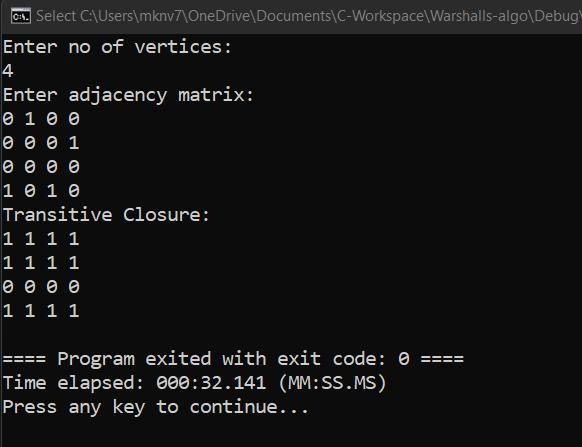
printf("%d ",a[i][j]);

printf("\n");

}

}

OUTPUT:



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

CODE:

#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()

{ clrscr();

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]);

}

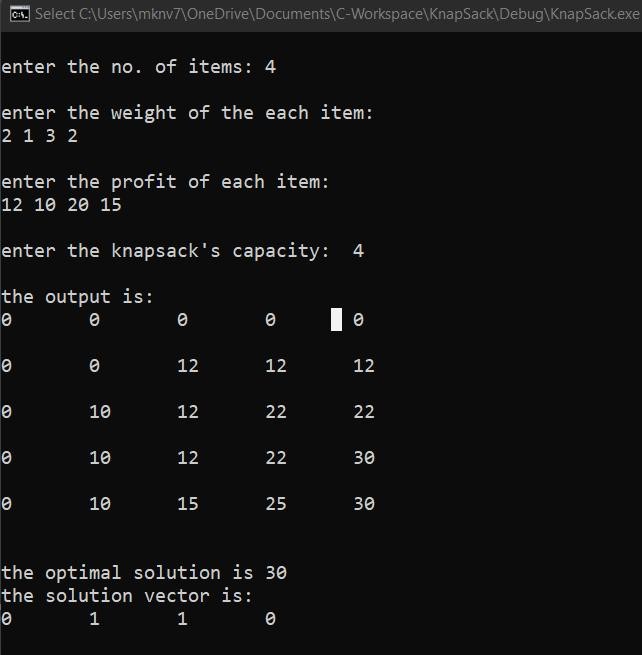
int max(int x,int y) { if(x>y) return x;

else

return y;

}

OUTPUT:



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

CODE:

#include<stdio.h> int n; void display(int dist[][n]); void floyd (int graph[][n])

{ int dist[n][n], i, j, k;

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

dist[i][j] = graph[i][j];

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

if (dist[i][k] + dist[k][j] < dist[i][j]) dist[i][j] = dist[i][k] + dist[k][j];

display(dist);

}

void display(int dist[][n]) { printf ("DISTANCE MATRIX \n"); for (int i = 0; i < n; i++)

{

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

{ if (dist[i][j] == 99) printf("99 ");

else printf ("%d ", dist[i][j]); }

printf("\n");

}

}

int main()

{

printf("ENTER ORDER OF MATRIX \n");

scanf("%d",&n);

int graph[n][n];

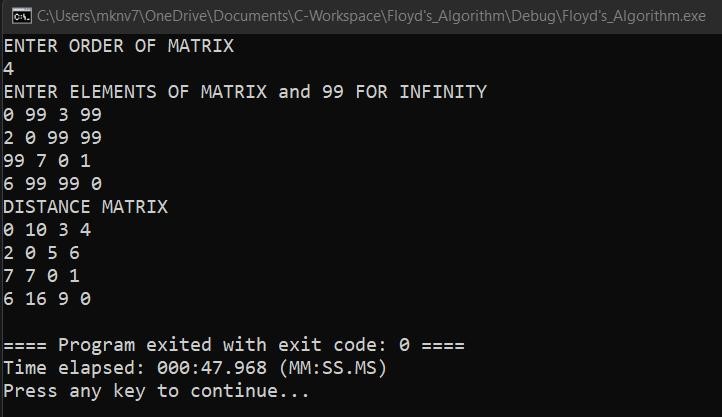
printf("ENTER ELEMENTS OF MATRIX and 99 FOR INFINITY\n"); for(int i = 0;i < n;i++) for(int j = 0;j < n; j++) scanf("%d",&graph[i][j]);

floyd(graph);

return 0;

}

OUTPUT:



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

CODE:

#include<stdio.h>

#include<conio.h> #include<process.h>

void main()

{ int i,j; int c[10][10],n;

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]); if(c[i][j]==0)

c[i][j]=1000;

}

} int 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=1000;

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]==0)

{

printf("\n%d----->%d=%d\n",u,v,min);

ne=ne+1;

mincost=mincost+min;

} elec[v]=1;

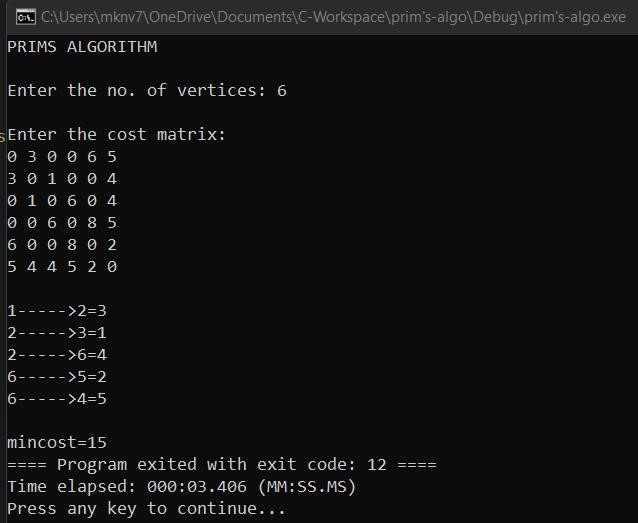
c[u][v]=c[v][u]=1000;

}

printf("\nmincost=%d",mincost);

}

OUTPUT:



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

CODE:

#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]);

if(c[i][j]==0) c[i][j]=9999;

}

} 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;

}

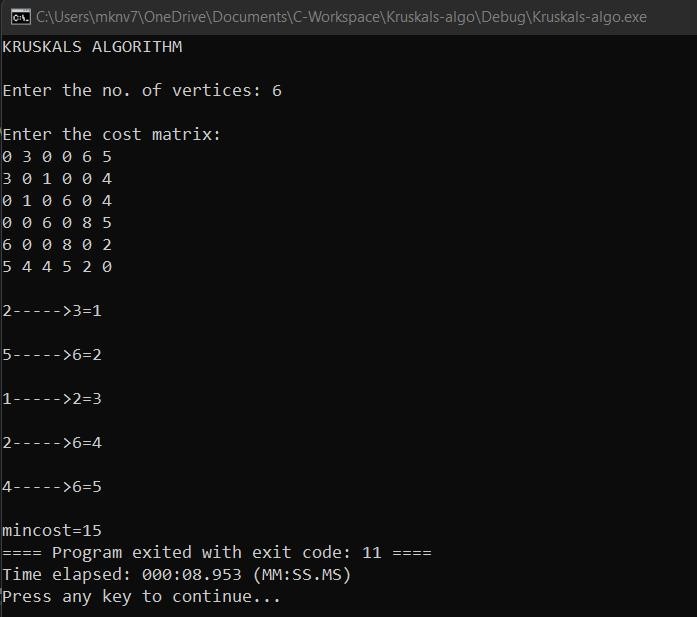
c[a][b]=c[b][a]=9999;

}

printf("\nmincost=%d",mincost);

}

OUTPUT:



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

CODE:

#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]);

if(c[i][j]==0) c[i][j]=9999;

}

}

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];

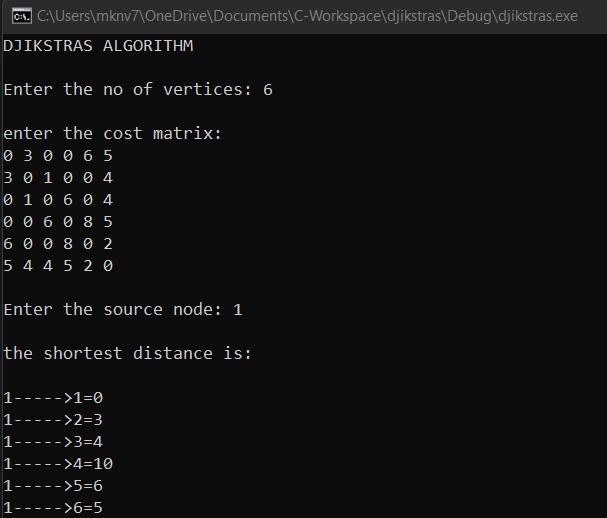
}

}

}

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

OUTPUT:



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.

CODE:

#include<stdio.h>

#include<conio.h>

#define TRUE 1 #define FALSE 0 int inc[50],w[50],sum,n; int promising(int i,int wt,int total) { return(((wt+total)>=sum)&&((wt==sum)||(wt+w[i+1]<=sum)));

}

void main() { int i,j,n,temp,total=0;

printf("Enter how many numbers:\n"); scanf("%d",&n);

printf("Enter %d numbers to th set:\n",n);

for (i=0;i<n;i++) { scanf("%d",&w[i]); total+=w[i];

}

printf("Input the sum value to create sub set: "); scanf("%d",&sum); for (i=0;i<=n;i++) for (j=0;j<n-1;j++) if(w[j]>w[j+1]) { temp=w[j];

w[j]=w[j+1];

w[j+1]=temp;

}

printf("The given %d numbers in ascending order:\n",n); for (i=0;i<n;i++) printf("%d ",w[i]);

if((total<sum))

printf("\n Subset construction is not possible"); else { for (i=0;i<n;i++)

inc[i]=0;

printf("\nSolution:\n"); sumset(-1,0,total);

} getch();

}

void sumset(int i,int wt,int total) { int j;

if(promising(i,wt,total)) { if(wt==sum) {

printf("{"); for (j=0;j<=i;j++) if(inc[j])

printf("%d ",w[j]);

printf("}\n");

} else { inc[i+1]=TRUE;

sumset(i+1,wt+w[i+1],total-w[i+1]); inc[i+1]=FALSE;

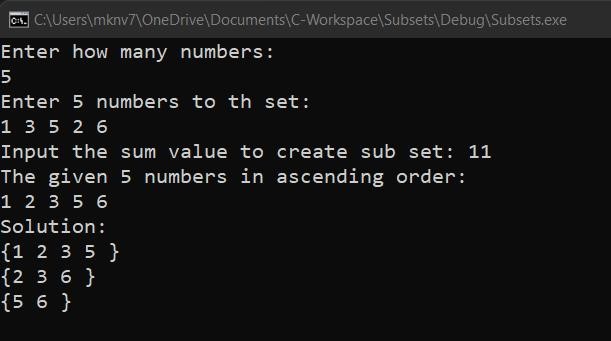
sumset(i+1,wt,total-w[i+1]);

}

}

}

OUTPUT:



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

CODE:

#include<stdio.h> #include<math.h> int board[20],count;

int main()

{

int n,i,j;

void queen(int row,int n);

printf("Enter number of Queens:"); scanf("%d",&n); queen(1,n);

return 0;

}

void print(int n)

{

int i,j;

printf("\n\nSolution %d:\n\n",++count);

for(i=1;i<=n;++i) printf("\t%d",i);

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

{ printf("\n\n%d",i); for(j=1;j<=n;++j)

{ if(board[i]==j) printf("\tQ"); else

printf("\t-");

}

}

}

int place(int row,int column)

{ int i; for(i=1;i<=row-1;++i)

{

if(board[i]==column)

return 0;

else

if(abs(board[i]-column)==abs(i-row))

return 0;

}

return 1;

}

void queen(int row,int n)

{

int column;

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

{

if(place(row,column))

{

board[row]=column; if(row==n)

print(n);

else queen(row+1,n);

}

}

}

OUTPUT:

