1)PROGRAM:

Identify the M-th maximum number and Nth minimum number in an array and then find the sum of it and difference of it.

CODE:

#include <stdio.h>

int main ()

{

int number[30],rev[30];

int i, j, a, n,m,nt;

printf("Enter the value of N\n");

scanf("%d", &n);

printf("Enter the numbers \n");

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

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

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

{

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

{

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

{

a = number[i];

number[i] = number[j];

number[j] = a;

}

}

}

for(i=(n-1),j=0; i>=0; i--,j++)

{

rev[j]=number[i];

}

printf("\nEnter the Mth maximum =");

scanf("%d",&m);

printf("\n%d th maximum number is %d",m,number[m-1]);

printf("\nEnter the Nth minimum =");

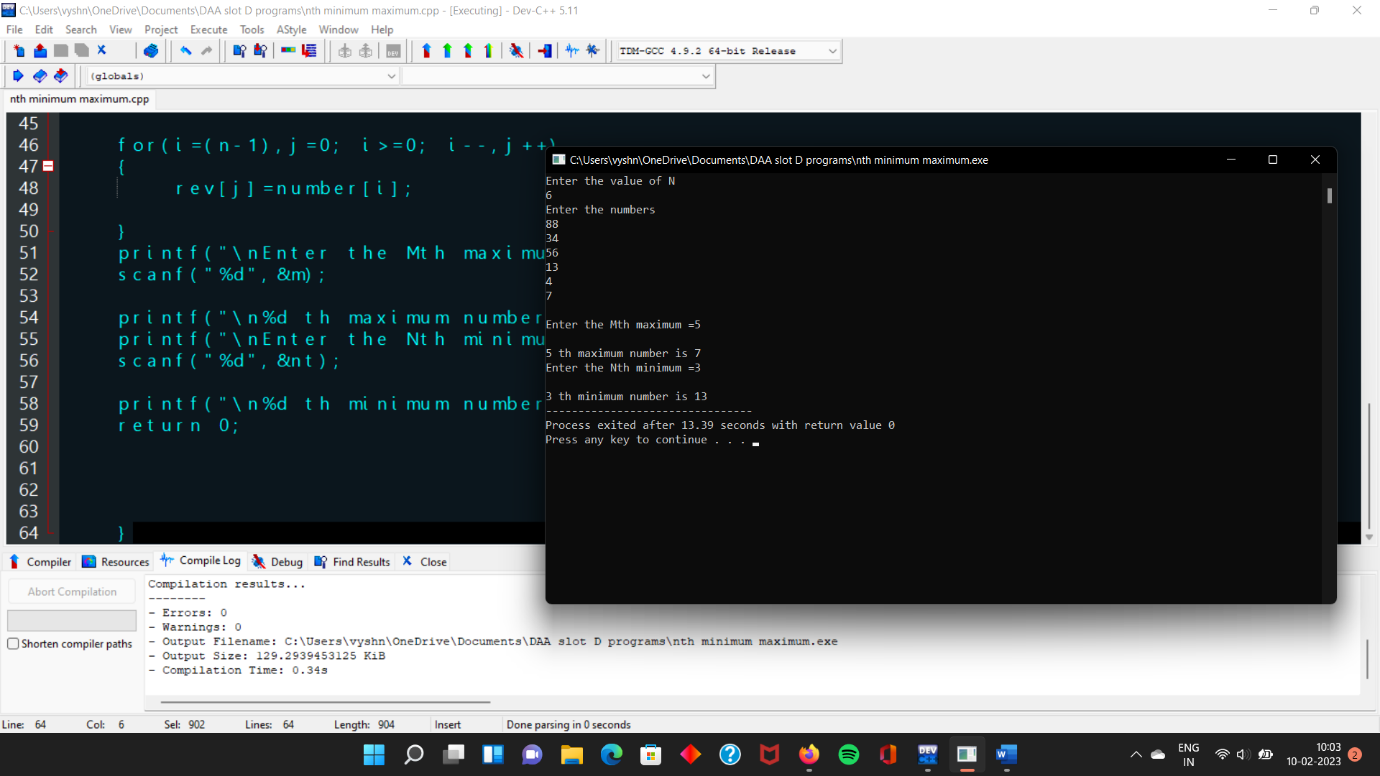
scanf("%d",&nt);

printf("\n%d th minimum number is %d",nt,rev[nt-1]);

return 0;

}

OUTPUT:



2)PROGRAM:

Write a program to perform sum of subsets problem using backtracking and estimate time complexity. Identify the test cases.

CODE:

#include <stdio.h>

#include <stdlib.h>

static int total\_nodes;

void printValues(int A[], int size){

for (int i = 0; i < size; i++) {

printf("%\*d", 5, A[i]);

}

printf("\n");

}

void subset\_sum(int s[], int t[], int s\_size, int t\_size, int sum, int ite, int const target\_sum){

total\_nodes++;

if (target\_sum == sum) {

printValues(t, t\_size);

subset\_sum(s, t, s\_size, t\_size - 1, sum - s[ite], ite + 1, target\_sum);

return;

}

else {

for (int i = ite; i < s\_size; i++) {

t[t\_size] = s[i];

subset\_sum(s, t, s\_size, t\_size + 1, sum + s[i], i + 1, target\_sum);

}

}

}

void generateSubsets(int s[], int size, int target\_sum){

int\* tuplet\_vector = (int\*)malloc(size \* sizeof(int));

subset\_sum(s, tuplet\_vector, size, 0, 0, 0, target\_sum);

free(tuplet\_vector);

}

int main(){

int set[] = { 5, 6, 12 , 54, 2 , 20 , 15 };

int sum;

int size = sizeof(set) / sizeof(set[0]);

printf("The set is ");

printValues(set , size);

printf("enter sum=");

scanf("%d",&sum);

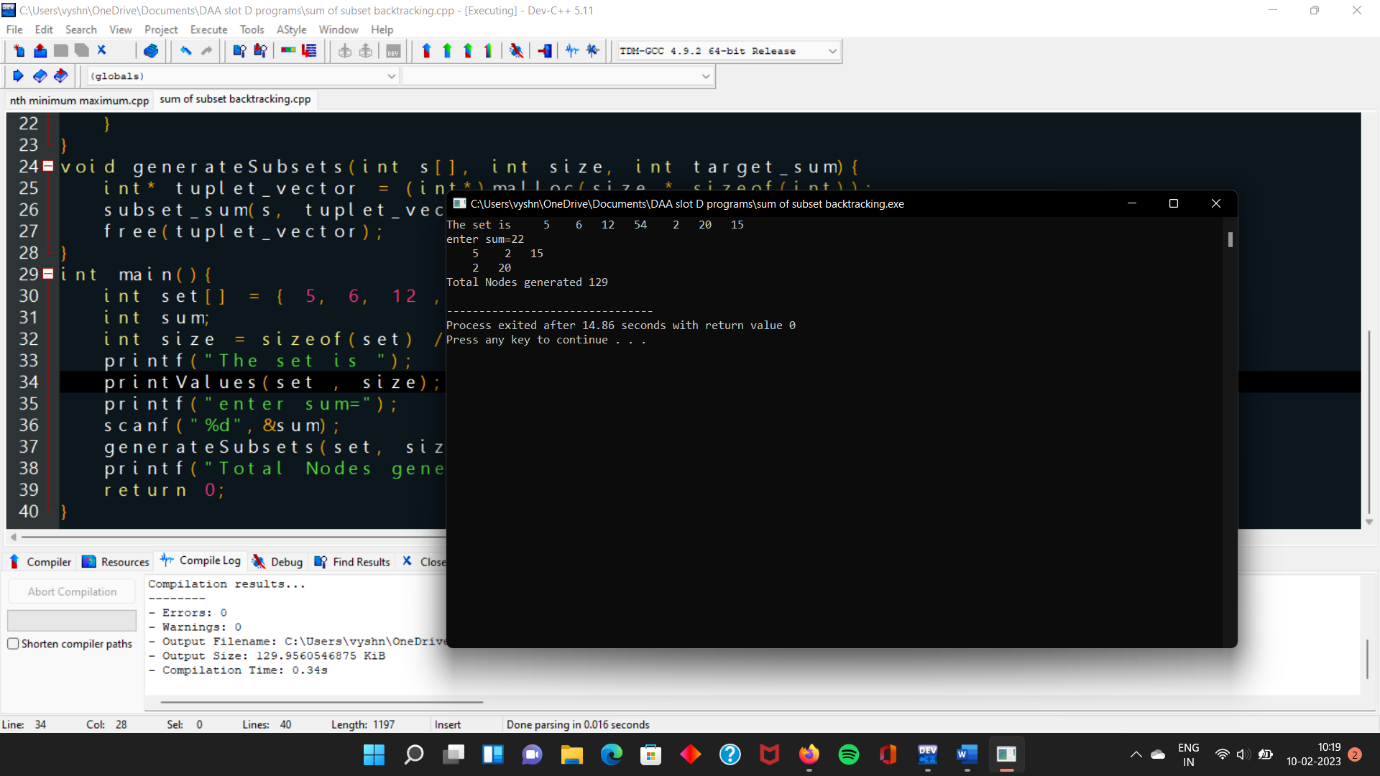
generateSubsets(set, size, sum);

printf("Total Nodes generated %d\n", total\_nodes);

return 0;

}

OUTPUT:



3)PROGRAM:

Write a program to perform Knapsack problem for the following set of object values.,

Knapsack weight 100

item Weight Profit

1 40 80

2 30 70

3 20 50

4 30 80

CODE:

#include<stdio.h>

int main ()

{

int n, m, w[100], p[100], ratio[100] , i, j, u, temp;

float xr, x[100], total\_profit=0, total\_weight=0;

printf ("Enter the number of items(n): ");

scanf ("%d", &n);

printf ("Enter the capacity of the Knapsack(m): ");

scanf ("%d", &m);

u = m;

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

{

x[i]=0;

}

printf ("Enter the Weights of items: ");

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

{

printf ("\n\tWeight of item %d = ", i + 1);

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

}

printf ("\nEnter the Profit Values of items: ");

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

{

printf ("\n\tProfit of item %d = ", i + 1);

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

}

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

{

ratio[i] = p[i] / w[i];

}

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

{

for (j = 0; j < n - 1; j++)

{

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

{

temp = ratio[i];

ratio[i] = ratio[j];

ratio[j] = temp;

temp = w[i];

w[i] = w[j];

w[j] = temp;

temp = p[i];

p[i] = p[j];

p[j] = temp;

}

}

}

printf("\n The Table After Sorting based on the Ratio: \n");

printf("\nItem:\t\t");

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

{

printf("%d\t",i+1);

}

printf("\nProfit:\t\t");

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

{

printf("%d\t",p[i]);

}

printf("\nWeights:\t");

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

{

printf("%d\t",w[i]);

}

printf ("\nRATIO:\t\t");

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

{

printf ("%d\t", ratio[i]);

}

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

{

if(w[i]<=u)

{

x[i]=1;

u=u-w[i];

}

else if(w[i]>u)

{

break;

}

}

if(i<=n)

{

xr = (float)u/w[i];

x[i] = xr;

}

printf("\n X = [");

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

{

printf("%.3f , ",x[i]);

}

printf("]");

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

{

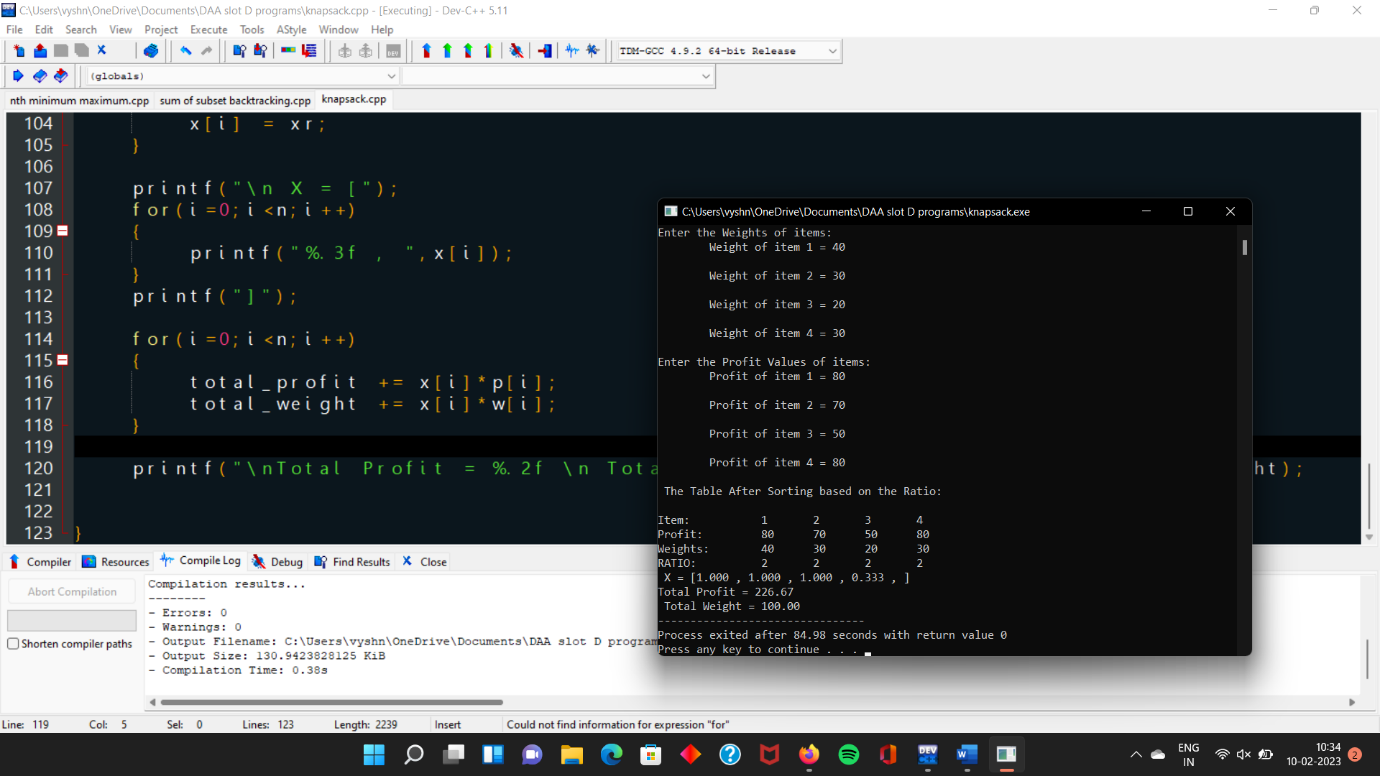
total\_profit += x[i]\*p[i];

total\_weight += x[i]\*w[i];

}

printf("\nTotal Profit = %.2f \n Total Weight = %.2f ",total\_profit,total\_weight);

}

OUTPUT:

4)PROGRAM:

The n-queens puzzle is the problem of placing n queens on an n x n chessboard such that no two queens attack each other. Given an integer n, return all distinct solutions to the n-queens puzzle. You may return the answer in any order. Write a program for the same.

CODE:

#include <stdio.h>

#include <stdbool.h>

#define N 8

int col[N];

bool check(int row) {

int i;

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

if (col[i] == col[row] ||

row - i == col[row] - col[i] ||

row - i == col[i] - col[row])

return false;

return true;

}

void backtrack(int row) {

int i;

if (row == N) {

for (i = 0; i < N; i++) printf("(%d, %d)\n", i, col[i]);

printf("\n");

return;

}

for (i = 0; i < N; i++) {

col[row] = i;

if (check(row)) backtrack(row + 1);

}

}

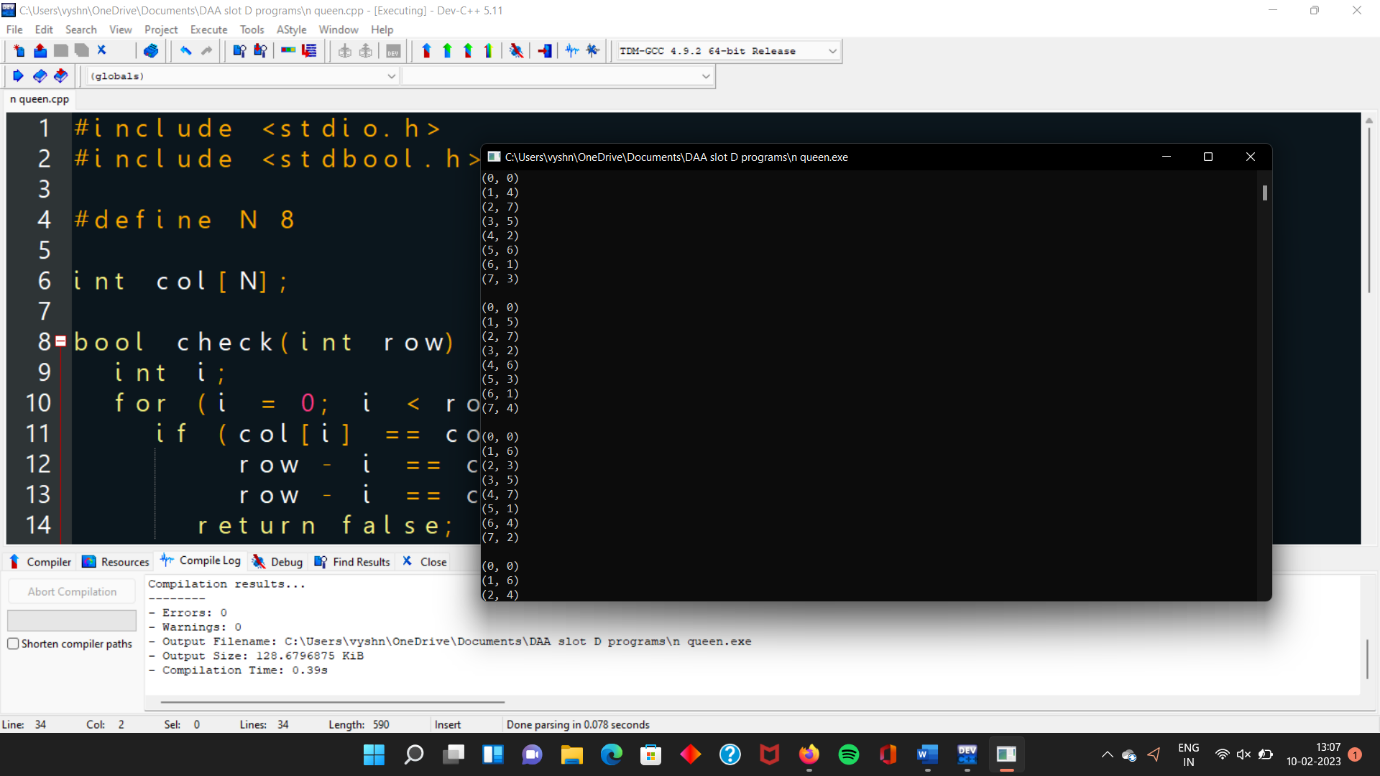
int main() {

backtrack(0);

return 0;

}

OUTPUT:



5)PROGRAM:

Determine an optimal tour in a weighted, directed graph. The weights are nonnegative numbers. The

inputs are weighted, directed graph, and n, the number of vertices in the graph. The graph is

represented by a two-dimensional array W, which has both its rows and columns indexed from 1 to n,

where W [i] [j] is the weight on the edge from the ith vertex to the jth vertex. Write a program for

travelling salesman problem using dynamic programming for the below given graph.

CODE:

#include <stdio.h>

#include <limits.h>

#define V 5

int minKey(int key[], int mstSet[]) {

int min = INT\_MAX, min\_index;

int v;

for (v = 0; v < V; v++)

if (mstSet[v] == 0 && key[v] < min)

min = key[v], min\_index = v;

return min\_index;

}

int printMST(int parent[], int n, int graph[V][V]) {

int i;

printf("Edge Weight\n");

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

printf("%d - %d %d \n", parent[i], i, graph[i][parent[i]]);

}

void primMST(int graph[V][V]) {

int parent[V]; // Array to store constructed MST

int key[V], i, v, count; // Key values used to pick minimum weight edge in cut

int mstSet[V]; // To represent set of vertices not yet included in MST

// Initialize all keys as INFINITE

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

key[i] = INT\_MAX, mstSet[i] = 0;

// Always include first 1st vertex in MST.

key[0] = 0; // Make key 0 so that this vertex is picked as first vertex

parent[0] = -1; // First node is always root of MST

// The MST will have V vertices

for (count = 0; count < V - 1; count++) {

int u = minKey(key, mstSet);

mstSet[u] = 1;

for (v = 0; v < V; v++)

if (graph[u][v] && mstSet[v] == 0 && graph[u][v] < key[v])

parent[v] = u, key[v] = graph[u][v];

}

// print the constructed MST

printMST(parent, V, graph);

}

int main() {

int graph[V][V] = { { 0, 2, 0, 6, 0 }, { 2, 0, 3, 8, 5 },

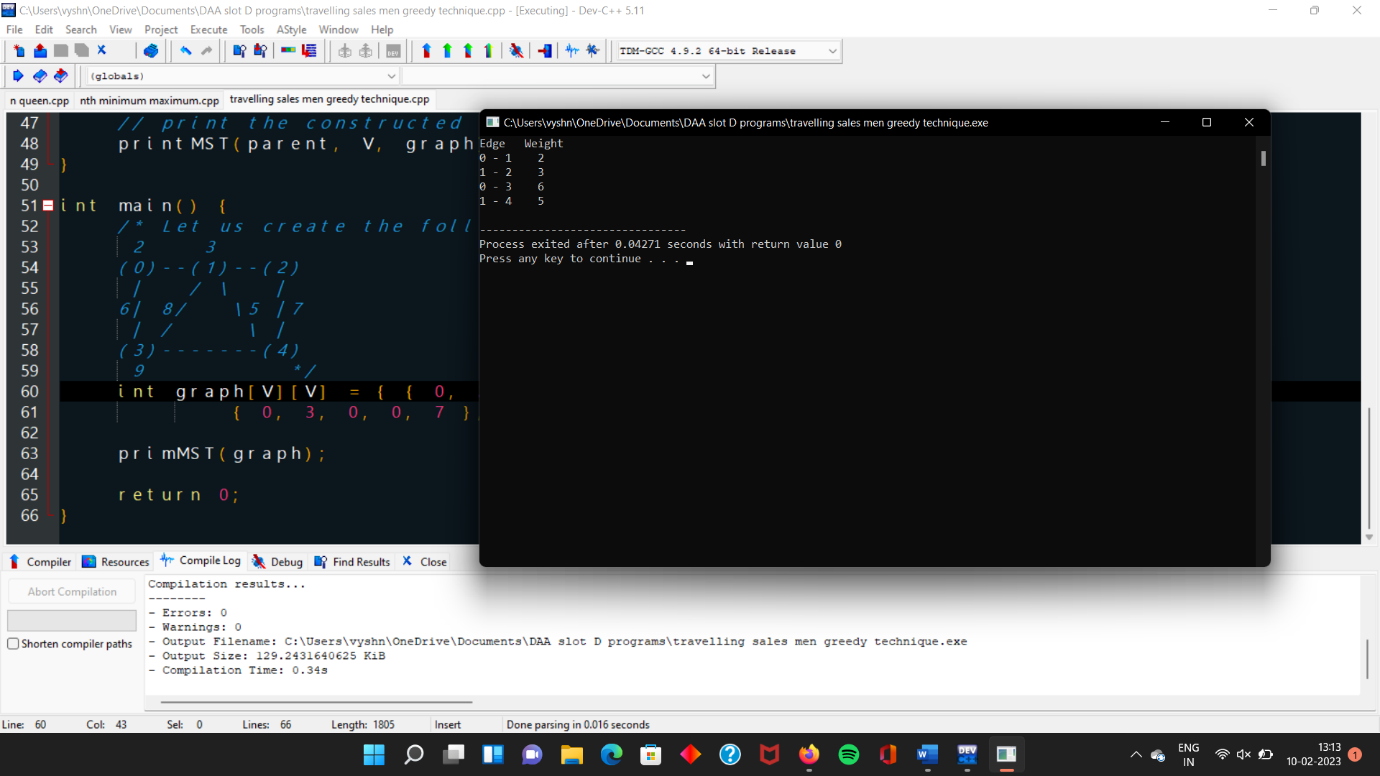
{ 0, 3, 0, 0, 7 }, { 6, 8, 0, 0, 9 }, { 0, 5, 7, 9, 0 }, };

primMST(graph);

return 0;

}

OUTPUT:



6)PROGRAM:

Write a program to perform Minimum spanning tree using greedy techniques and estimate time complexity for the given set of values.

CODE:

// Kruskal's algorithm in C

#include <stdio.h>

#define MAX 30

typedef struct edge {

int u, v, w;

} edge;

typedef struct edge\_list {

edge data[MAX];

int n;

} edge\_list;

edge\_list elist;

int Graph[MAX][MAX], n;

edge\_list spanlist;

void kruskalAlgo();

int find(int belongs[], int vertexno);

void applyUnion(int belongs[], int c1, int c2);

void sort();

void print();

// Applying Krushkal Algo

void kruskalAlgo() {

int belongs[MAX], i, j, cno1, cno2;

elist.n = 0;

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

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

if (Graph[i][j] != 0) {

elist.data[elist.n].u = i;

elist.data[elist.n].v = j;

elist.data[elist.n].w = Graph[i][j];

elist.n++;

}

}

sort();

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

belongs[i] = i;

spanlist.n = 0;

for (i = 0; i < elist.n; i++) {

cno1 = find(belongs, elist.data[i].u);

cno2 = find(belongs, elist.data[i].v);

if (cno1 != cno2) {

spanlist.data[spanlist.n] = elist.data[i];

spanlist.n = spanlist.n + 1;

applyUnion(belongs, cno1, cno2);

}

}

}

int find(int belongs[], int vertexno) {

return (belongs[vertexno]);

}

void applyUnion(int belongs[], int c1, int c2) {

int i;

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

if (belongs[i] == c2)

belongs[i] = c1;

}

// Sorting algo

void sort() {

int i, j;

edge temp;

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

for (j = 0; j < elist.n - 1; j++)

if (elist.data[j].w > elist.data[j + 1].w) {

temp = elist.data[j];

elist.data[j] = elist.data[j + 1];

elist.data[j + 1] = temp;

}

}

// Printing the result

void print() {

int i, cost = 0;

for (i = 0; i < spanlist.n; i++) {

printf("\n%d - %d : %d", spanlist.data[i].u, spanlist.data[i].v, spanlist.data[i].w);

cost = cost + spanlist.data[i].w;

}

printf("\nSpanning tree cost: %d", cost);

}

int main() {

int i, j, total\_cost;

n = 6;

Graph[0][0] = 0;

Graph[0][1] = 4;

Graph[0][2] = 4;

Graph[0][3] = 0;

Graph[0][4] = 0;

Graph[0][5] = 0;

Graph[0][6] = 0;

Graph[1][0] = 4;

Graph[1][1] = 0;

Graph[1][2] = 2;

Graph[1][3] = 0;

Graph[1][4] = 0;

Graph[1][5] = 0;

Graph[1][6] = 0;

Graph[2][0] = 4;

Graph[2][1] = 2;

Graph[2][2] = 0;

Graph[2][3] = 3;

Graph[2][4] = 4;

Graph[2][5] = 0;

Graph[2][6] = 0;

Graph[3][0] = 0;

Graph[3][1] = 0;

Graph[3][2] = 3;

Graph[3][3] = 0;

Graph[3][4] = 3;

Graph[3][5] = 0;

Graph[3][6] = 0;

Graph[4][0] = 0;

Graph[4][1] = 0;

Graph[4][2] = 4;

Graph[4][3] = 3;

Graph[4][4] = 0;

Graph[4][5] = 0;

Graph[4][6] = 0;

Graph[5][0] = 0;

Graph[5][1] = 0;

Graph[5][2] = 2;

Graph[5][3] = 0;

Graph[5][4] = 3;

Graph[5][5] = 0;

Graph[5][6] = 0;

kruskalAlgo();

print();

}

OUTPUT:

