PRACTICAL-4 (DAA )

NAME-VISHWAJIT DEVRAM KATE

REG NO – 2020BIT011

1. Travelling salesman Problem.

// Reg No:2020BIT011

#include <bits/stdc++.h>

using namespace std;

#define V 4

int travllingSalesmanProblem(int graph[][V], int s)

{

vector<int> vertex;

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

if (i != s)

vertex.push\_back(i);

int min\_path = INT\_MAX;

do {

int current\_pathweight = 0;

int k = s;

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

current\_pathweight += graph[k][vertex[i]];

k = vertex[i];

}

current\_pathweight += graph[k][s];

min\_path = min(min\_path, current\_pathweight);

} while (

next\_permutation(vertex.begin(), vertex.end()));

return min\_path;

}

int main()

{

int graph[][V] = { { 0, 10, 15, 20 },

{ 10, 0, 35, 25 },

{ 15, 35, 0, 30 },

{ 20, 25, 30, 0 } };

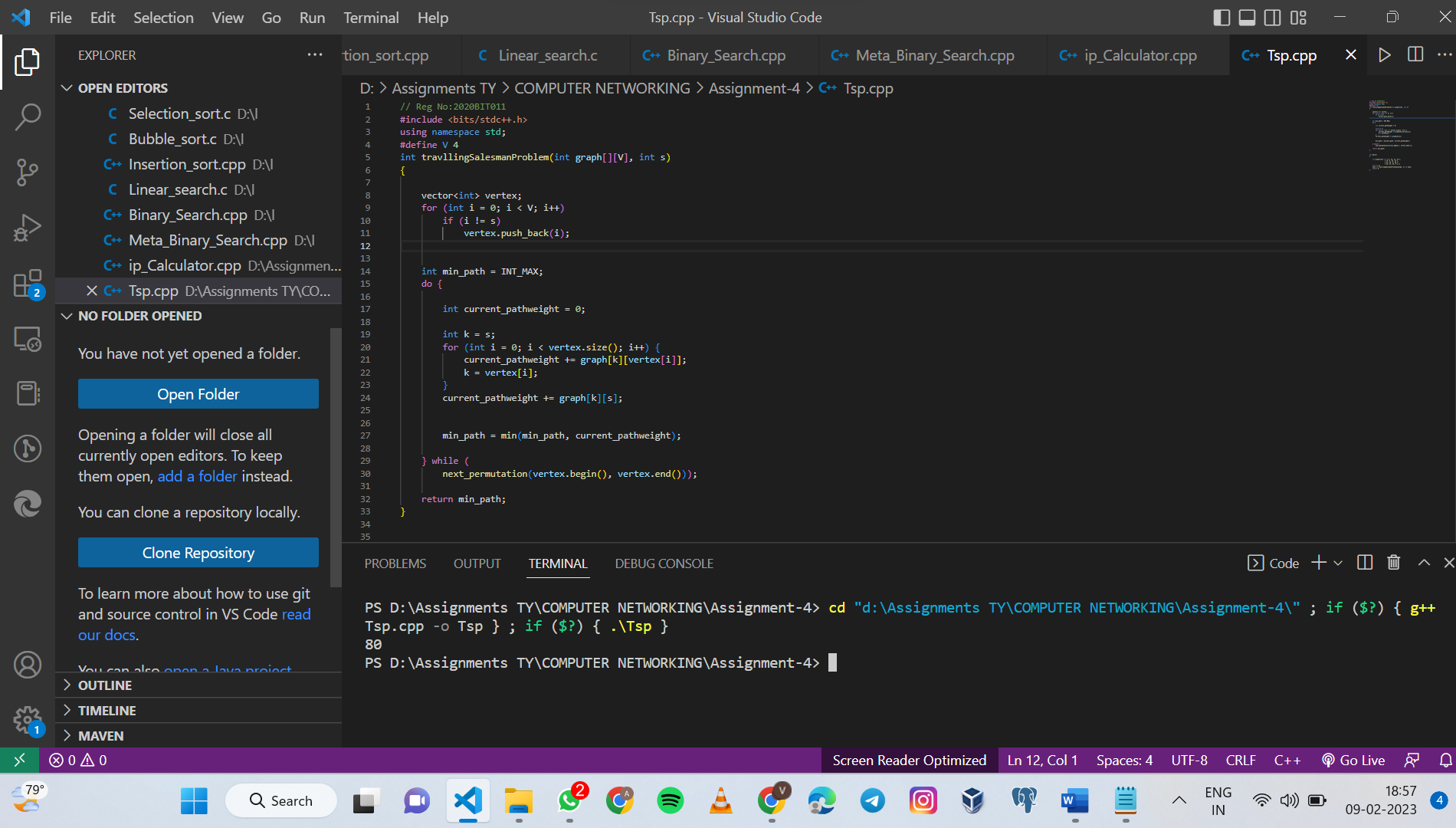
int s = 0;

cout << travllingSalesmanProblem(graph, s) << endl;

return 0;

}

Output:



1. **BF string Matching Algorithm**

int BFindex(String S, String T)

{

if (S.size() < 1 || T.size() < 1 )

return -1;

int i = 0, j = 0;

while (i < S.size() && j < T.size())

{

if (S[i] == T[j])

{

++i; ++j;

}

else

{

i = i- j+ 2;

j = 1;

}

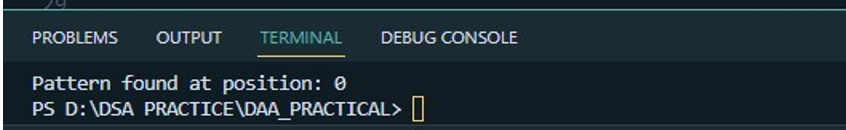
}

if(j > T[0]) return i - T[0];

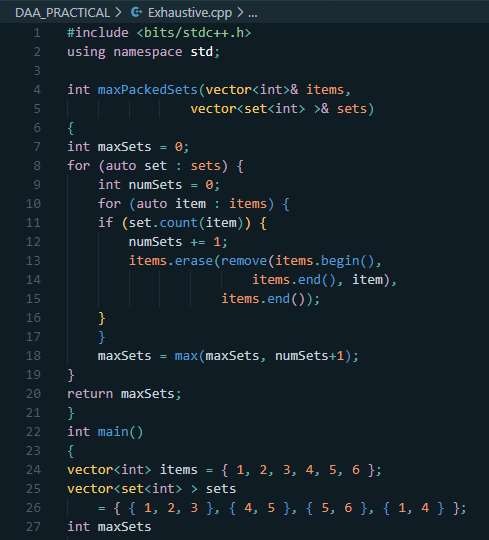
return ERROR;

}

Output:



1. **Exhaustive Search Algorithm**



Output:

