**Floyd-Warshall algorithm**

**CODE:**

**#include<iostream>**

**#include<iomanip>**

**using namespace std;**

**void floyd(int n,int \*\*d,int \*\*pi){**

**for(int k=0;k<n;k++){**

**for(int i=0;i<n;i++){**

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

**if(d[i][k]+d[k][j]<d[i][j]){**

**d[i][j]=d[i][k]+d[k][j];**

**pi[i][j]=pi[k][j];**

**}**

**}**

**}**

**}**

**cout<<"\nD matrix\n";**

**for(int i=0;i<n;i++){**

**cout<<"Edges starting with vertex "<<i<<": ";**

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

**cout<<d[i][j]<<" ";**

**}**

**cout<<endl;**

**}**

**cout<<"\nPI matrix\n";**

**for(int i=0;i<n;i++){**

**cout<<"Edges starting with vertex "<<i<<": ";**

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

**cout<<pi[i][j]<<" ";**

**}**

**cout<<endl;**

**}**

**}**

**void path(int i,int j,int \*\*pi){**

**if(pi[i][j]==-1){**

**cout<<i;**

**}**

**else{**

**path(i,pi[i][j],pi);**

**cout<<" -> "<<j;**

**}**

**}**

**int main(){**

**int n;**

**cout<<"Enter the number of the vertices: ";**

**cin>>n;**

**int \*\*matrix=new int\*[n];**

**int \*\*pi=new int\*[n];**

**cout<<"Vertices in the graph are: ";**

**for(int i=0;i<n;i++){**

**cout<<i<<" ";**

**matrix[i]=new int[n];**

**pi[i]=new int[n];**

**}**

**cout<<endl<<endl;**

**cout<<"Enter the edges of the graph\n";**

**for(int i=0;i<n;i++){**

**cout<<"Edges starting with vertex "<<i<<": ";**

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

**cin>>matrix[i][j];**

**if(matrix[i][j]==-1 || matrix[i][j]==0){**

**pi[i][j]=-1;**

**}**

**else{**

**pi[i][j]=i;**

**}**

**if(matrix[i][j]==-1){**

**matrix[i][j]=INT\_MAX/2;**

**}**

**}**

**}**

**floyd(n,matrix,pi);**

**int a,b;**

**int x,exit=0;**

**while(exit==0){**

**cout<<"\n1.Find smallest path\n2.Exit\n";**

**cout<<"What do you want to perform: ";**

**cin>>x;**

**cout<<"\n";**

**switch(x){**

**case 1:**

**cout<<"Shortest path between 2 vertices\n";**

**cout<<"Enter Starting vertex: ";**

**cin>>a;**

**cout<<"Enter Ending vertex: ";**

**cin>>b;**

**cout<<"Shortest path: ";**

**path(a,b,pi);**

**cout<<endl;**

**break;**

**case 2:**

**cout<<"Program Exited";**

**exit=1;**

**break;**

**default:**

**cout<<"ERROR\nThe input is invalid. \n\n";**

**break;**

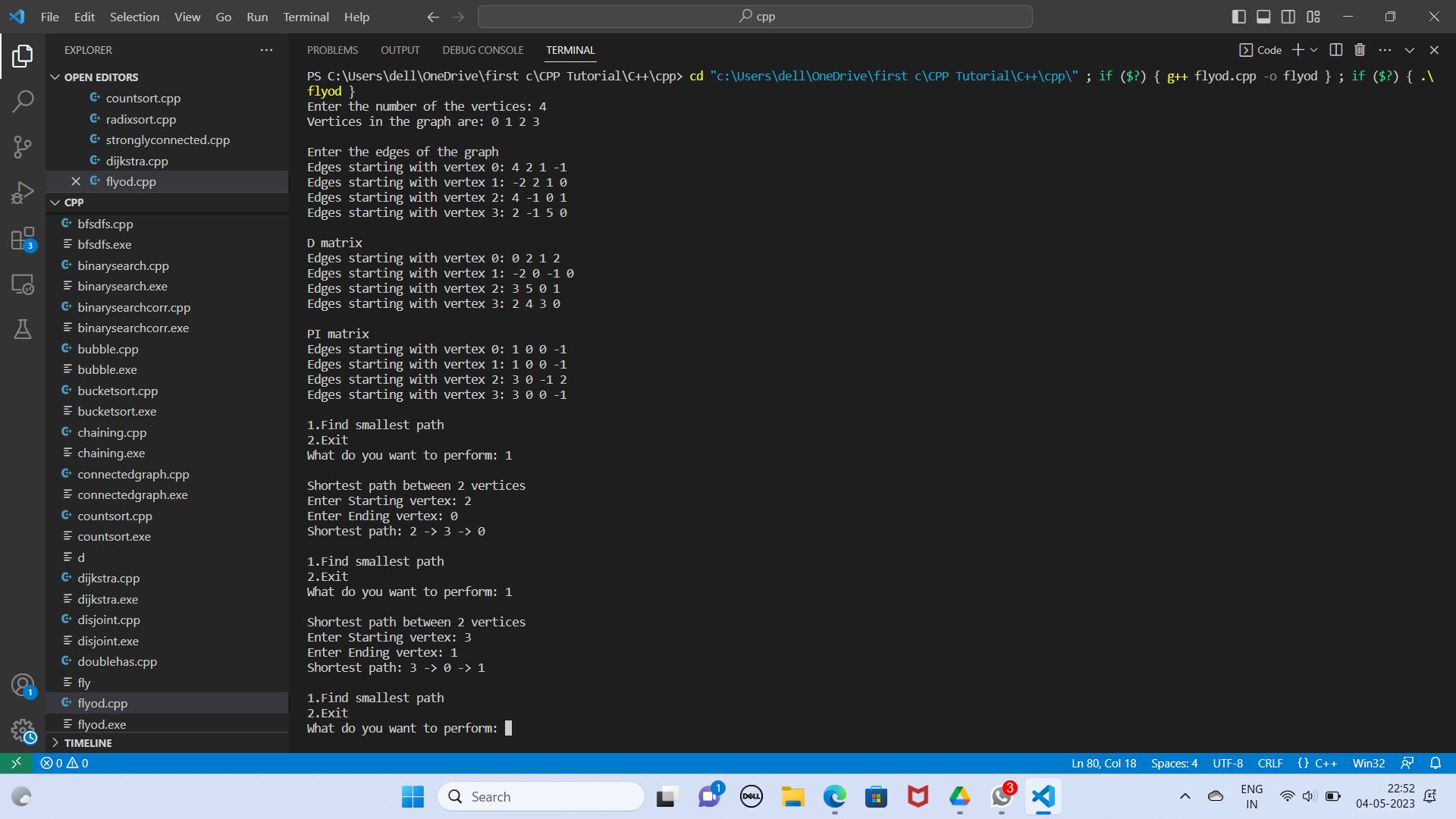
**}**

**}**

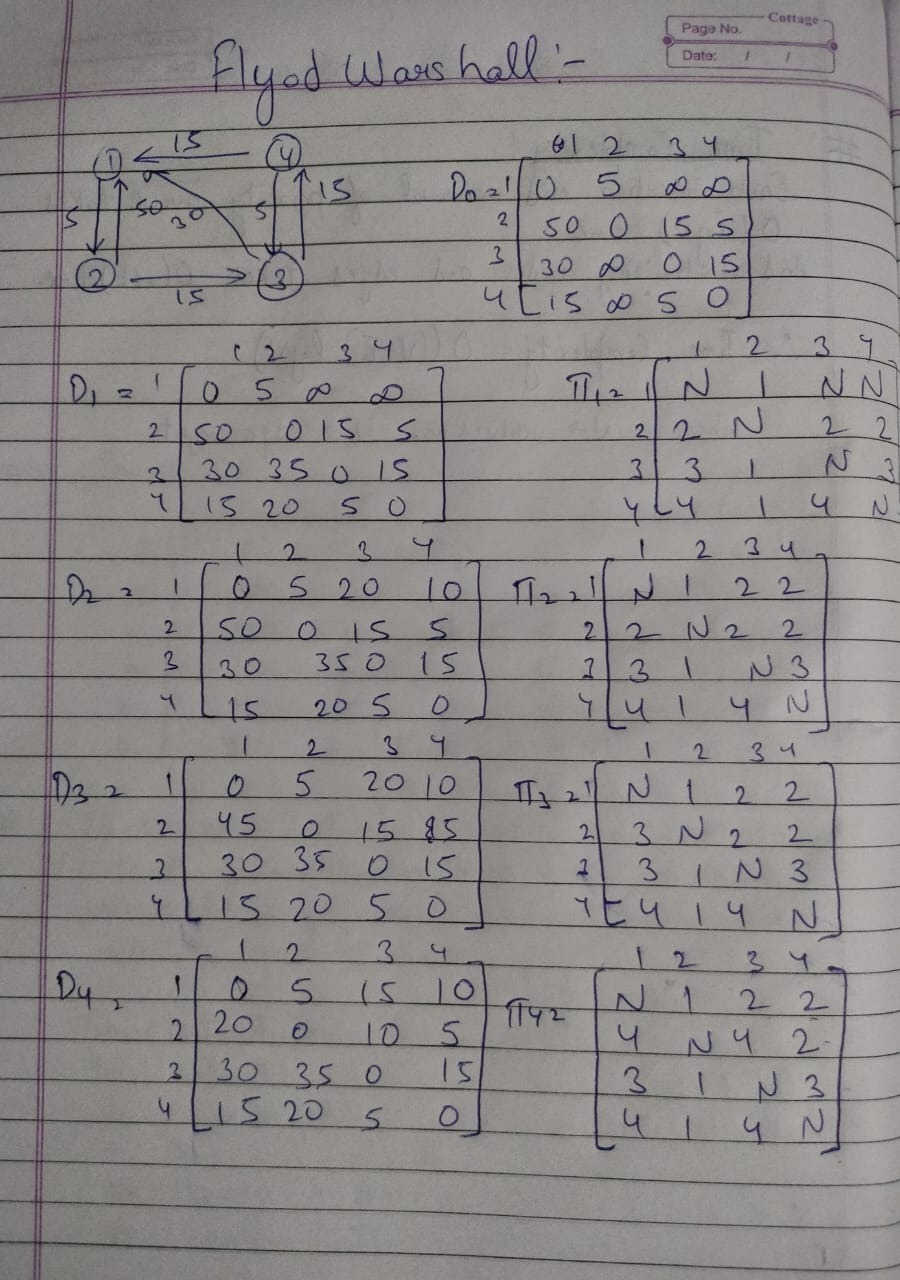
**return 0;**

**}**

**OUTPUT :**



**GRAPH IMPLEMENTATION :**



**TIME COMPLEXITY :**

