

BỘ GIÁO DỤC VÀ ĐÀO TẠO  
ĐẠI HỌC UEH  
KHOA CÔNG NGHỆ THÔNG TIN KINH DOANH



## ĐỒ ÁN MÔN HỌC

### ĐỀ TÀI

# ĐỒ THỊ VÀ ỨNG DỤNG TRONG QUẢN LÝ LỘ TRÌNH ĐƯỜNG ĐI

Học Phần: Cấu Trúc Dữ Liệu & Giải Thuật  
Danh Sách Nhóm:

1. NGUYỄN NGỌC BẢO
2. LÒ SÌN DẬU
3. ĐOÀN TRẦN BÁ ĐẠT
4. NGUYỄN HOÀN THIÊN

Chuyên Ngành: CÔNG NGHỆ PHẦN MỀM  
Khóa: K46

Giảng Viên: TS. Đặng Ngọc Hoàng Thành

Tp. Hồ Chí Minh, Ngày 19 tháng 12 năm 2021

# MỤC LỤC

<b>MỤC LỤC.....</b>	<b>2</b>
<b>CHƯƠNG 1. ĐỒ THỊ.....</b>	<b>3</b>
1.1. Các Khái Niệm Liên Quan .....	3
1.2. Cấu Trúc và Cài Đặt Đồ Thị.....	3
1.3. Các Thuật Toán Trên Đồ Thị.....	4
a) Thuật Toán Tìm kiếm theo chiều sâu - Depth-First Search .....	4
PSEUDOCODE.....	4
b) Thuật Toán Tìm kiếm theo chiều rộng - Breadth First Search .....	5
PSEUDOCODE.....	5
c) Thuật Toán Tìm đường đi ngắn nhất .....	5
PSEUDOCODE.....	5
<b>CHƯƠNG 2. PHÂN TÍCH VÀ THIẾT KẾ LỚP .....</b>	<b>6</b>
2.1. Phân Tích Bài Toán Quản Lý Lộ Trình Đường Đi .....	6
2.2. Sơ Đồ Lớp.....	8
2.3. Cài Đặt Lớp.....	9
<b>CHƯƠNG 3. THIẾT KẾ GIAO DIỆN .....</b>	<b>21</b>
3.1. Giao Diện Menu Chính .....	21
3.2. Chi Tiết Chức Năng.....	21
<b>CHƯƠNG 4. THẢO LUẬN &amp; ĐÁNH GIÁ .....</b>	<b>24</b>
4.1. Các Kết Quả Nhận Được.....	24
4.2. Một Số Tồn Tại .....	24
4.3. Hướng Phát Triển.....	24
<b>PHỤ LỤC.....</b>	<b>24</b>
<b>TÀI LIỆU THAM KHẢO .....</b>	<b>28</b>

# CHƯƠNG 1. ĐỒ THỊ

## 1.1. Các Khái Niệm Liên Quan

- Đồ thị (Graph) là một dạng cấu trúc dữ liệu được cấu tạo từ tập các đỉnh  $V$  (vertex) và tập các cạnh  $E$  (edge):  $G=(V, E)$ .
- Hai đỉnh được nối với nhau tạo thành một cặp đỉnh (pair). Mỗi cặp đỉnh như vậy tạo thành một cạnh. Một cạnh được tạo từ một đỉnh duy nhất gọi là khuyên. Nếu tồn tại hai cạnh trên cùng một cặp điểm, thì cặp cạnh đó gọi là cặp cạnh song song (hoặc cạnh bội).
- Một đồ thị mà trong đó các cặp đỉnh là các bộ sắp thứ tự được gọi Đồ thị có hướng (Digraph). Ngược lại, được gọi là Đồ thị vô hướng.
- Nếu mỗi cạnh của đồ thị được gán với một trọng số (weight), thì đồ thị đó được gọi là Đồ thị có trọng số (hoặc là mạng lưới network).
- Đơn đồ thị là đồ thị (có hướng hoặc vô hướng) KHÔNG chứa khuyên hoặc cạnh bội.
- Để biểu diễn đồ thị: biểu diễn hình học, biểu diễn bằng ma trận liên kề.

## 1.2. Cấu Trúc và Cài Đặt Đồ Thị

Lớp Đỉnh Vertex	
<pre>public class <b>Vertex</b> {     public bool wasVisited;     public string label;     public Vertex(string label)     {         this.label = label;         wasVisited = false;     } }</pre>	
<pre>public class <b>Graph</b> {     private const int NUM_VERTICES = 20;     private Vertex[] vertices;     private int[,] adjMatrix;</pre>	<pre>public void AddVertex(string label) {     vertices[numVerts] = new <b>Vertex</b>(label);     numVerts++; }</pre>

<pre> int numVerts;  public Graph() {     vertices = new <b>Vertex</b>[NUM_VERTICES];     adjMatrix = new int[NUM_VERTICES, NUM_VERTICES];     numVerts = 0;     for (int j = 0; j &lt; NUM_VERTICES; j++)         for (int k = 0; k &lt; NUMVERTICES; k++)             adjMatrix[j, k] = 0; } </pre>	<pre> public void AddEdge(int start, int eend) {     adjMatrix[start, eend] = 1;     adjMatrix[eend, start] = 1; }  public void ShowVertex(int v) {     Console.Write(vertices[v].label + " "); } } </pre>
---	--

### 1.3. Các Thuật Toán Trên **Đồ Thị**

#### a) Thuật Toán Tìm kiếm theo chiều sâu - *Depth-First Search*

<i><b>PSEUDOCODE</b></i>	
<pre> procedure GetAdjUnvisitedVertex(int v):     FOR j = 0 to (numVertices - 1)         IF adjMatrix(v,j) == 1 &amp;&amp;             vertices[j].WasVisited == false THEN             return j         ENDIF     ENDFOR     return -1  procedure DepthFirstSearch():     vertices[0].WasVisited = true     ShowVertex(0)     Stack gStack = new Stack()     gStack.Push(0) </pre>	<pre> WHILE gStack.Count &gt; 0     v = GetAdjUnvisitedVertex(gStack.Peek())     IF v == -1 THEN         gStack.Pop()     ELSE         vertices[v].WasVisited = true         ShowVertex(v)         gStack.Push(v)     ENDIF     FOR j = 0 to (numVertices - 1)         vertices[j].WasVisited = false     ENDFOR </pre>

***b) Thuật Toán Tìm kiếm theo chiều rộng - Breadth First Search***

<b><i>PSEUDOCODE</i></b>	
<pre> procedure BreadthFirstSearch():     gQueue = new Queue()     vertices[0].WasVisited = true     ShowVertex(0)     gQueue.Enqueue(0)     WHILE gQueue.Count &gt; 0         vert1 = gQueue.Dequeue()         vert2 = GetAdjUnvisitedVertex(vert1) </pre>	<pre>         WHILE vert2! = -1             vertices[vert2].WasVisited = true             ShowVertex(vert2)             gQueue.Enqueue(vert2)             vert2 = GetAdjUnvisitedVertex(vert1)         ENDWHILE     ENDWHILE     FOR i = 0 to (numVertices - 1)         vertices[index].WasVisited = false     ENDFOR </pre>

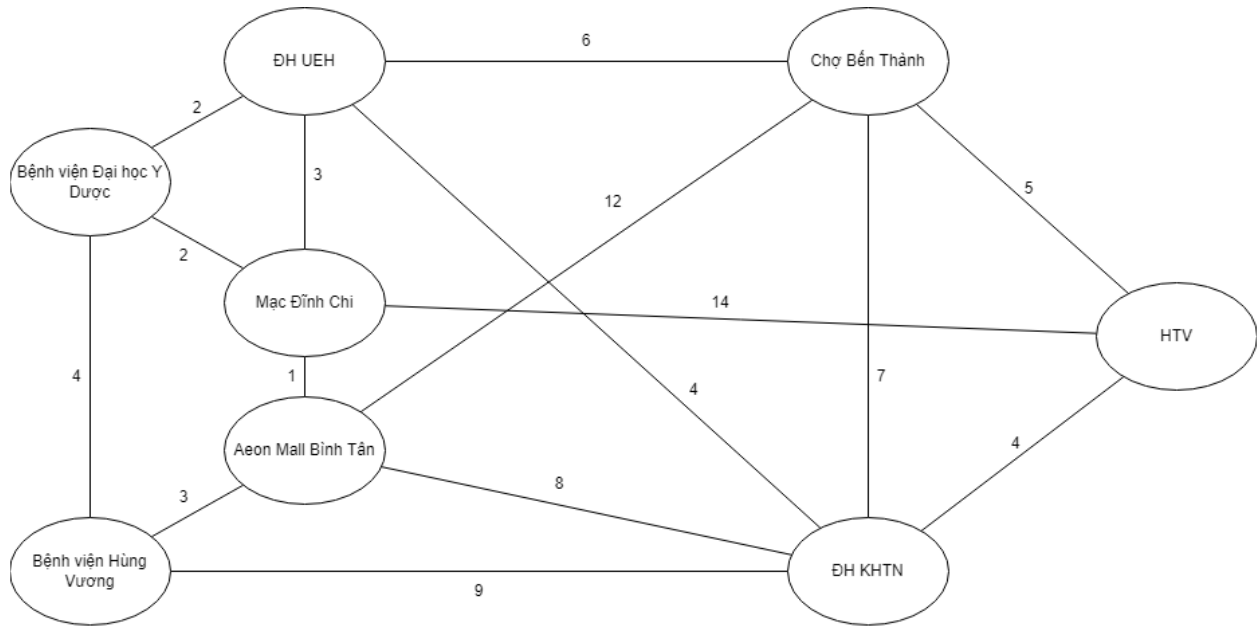
***c) Thuật Toán Tìm đường đi ngắn nhất***

<b><i>PSEUDOCODE</i></b>	
<pre> procedure Path():     startTree = 0     vertexList[startTree].isInTree = true     nTree = 1     FOR j = 0 to nVerts         tempDist = adjMat[startTree, j]         sPath[j] = new DistOriginal(startTree, tempDist)     ENDFOR     WHILE nTree &lt; nVerts         indexMin = GetMin() </pre>	<pre>         minDist = sPath[indexMin].distance         currentVert = indexMin         startToCurrent = sPath[indexMin].distance         vertexList[currentVert].isInTree = true         nTree++         AdjustShortPath()     ENDWHILE     DisplayPaths()     nTree = 0     FOR j = 0 to (nVerts - 1)         vertexList[j].isInTree = false     ENDFOR </pre>

## CHƯƠNG 2. PHÂN TÍCH VÀ THIẾT KẾ LỚP

### 2.1. Phân Tích Bài Toán **Quản Lý Lộ Trình Đường Đi**

- Bài toán Quản lý lộ trình đường đi là một bài toán được thực hiện dựa trên bản đồ. Trước tiên ta cần phải cài đặt và sử dụng cấu trúc Đồ thị (Graph) để có thể biểu diễn bản đồ trong ngôn ngữ lập trình.



- Phần mềm Quản lý lộ trình đường đi phải đáp ứng đủ các nhu cầu của người dùng trong việc tìm kiếm và tham khảo các thông tin cần thiết về **một địa điểm** hoặc **lộ trình** nào đó để đưa ra quyết định.

- Do vậy dựa trên nhu cầu thực tế của người dùng chương trình sẽ có những nhóm chức năng chính cung cấp thông tin **theo địa điểm** và **theo lộ trình**:

+ Nhóm chức năng cung cấp thông tin thực tế theo địa điểm gồm:

- Tìm kiếm tên địa điểm theo từ khóa:

Trong thực tế khi ta tìm kiếm một địa điểm chung chung nào đó như bệnh viện, trường học, đại học, ... thì sẽ có rất nhiều địa điểm có từ khóa tương tự do vậy khi người dùng có nhu cầu tìm kiếm địa điểm theo từ khóa nào đó thì chương trình sẽ cung cấp một danh sách những địa điểm có chứa từ khóa do người dùng nhập.

- Tìm địa điểm gần nhất hoặc những địa điểm xung quanh một địa điểm:

Khi người dùng muốn đến một nơi nào đó thì họ thường sẽ có xu hướng là tìm thêm những địa điểm xung quanh hoặc địa điểm gần đó nhất để có thể tiện đường đi

đến với mục đích vui chơi, hoặc công việc, ... Để có thể thực hiện chức năng này, ta cần viết thuật toán thao tác dựa trên ma trận kề đã lưu các đỉnh và trọng số.

+ Nhóm chức năng cung cấp thông tin thực tế theo lộ trình gồm:

- Tùy chọn được điểm xuất phát và điểm kết thúc:

Đây là chức năng cần thiết nhất để có thể tạo nên một lộ trình đường đi nhất định nào đó.

- Hiện thị quãng đường ngắn nhất:

Dựa trên thuật toán tìm đường đi ngắn nhất Dijkstra ta sẽ tìm đường trọng số nhỏ nhất từ điểm xuất phát đến điểm kết thúc trong đồ thị tức là quãng đường đi ngắn nhất.

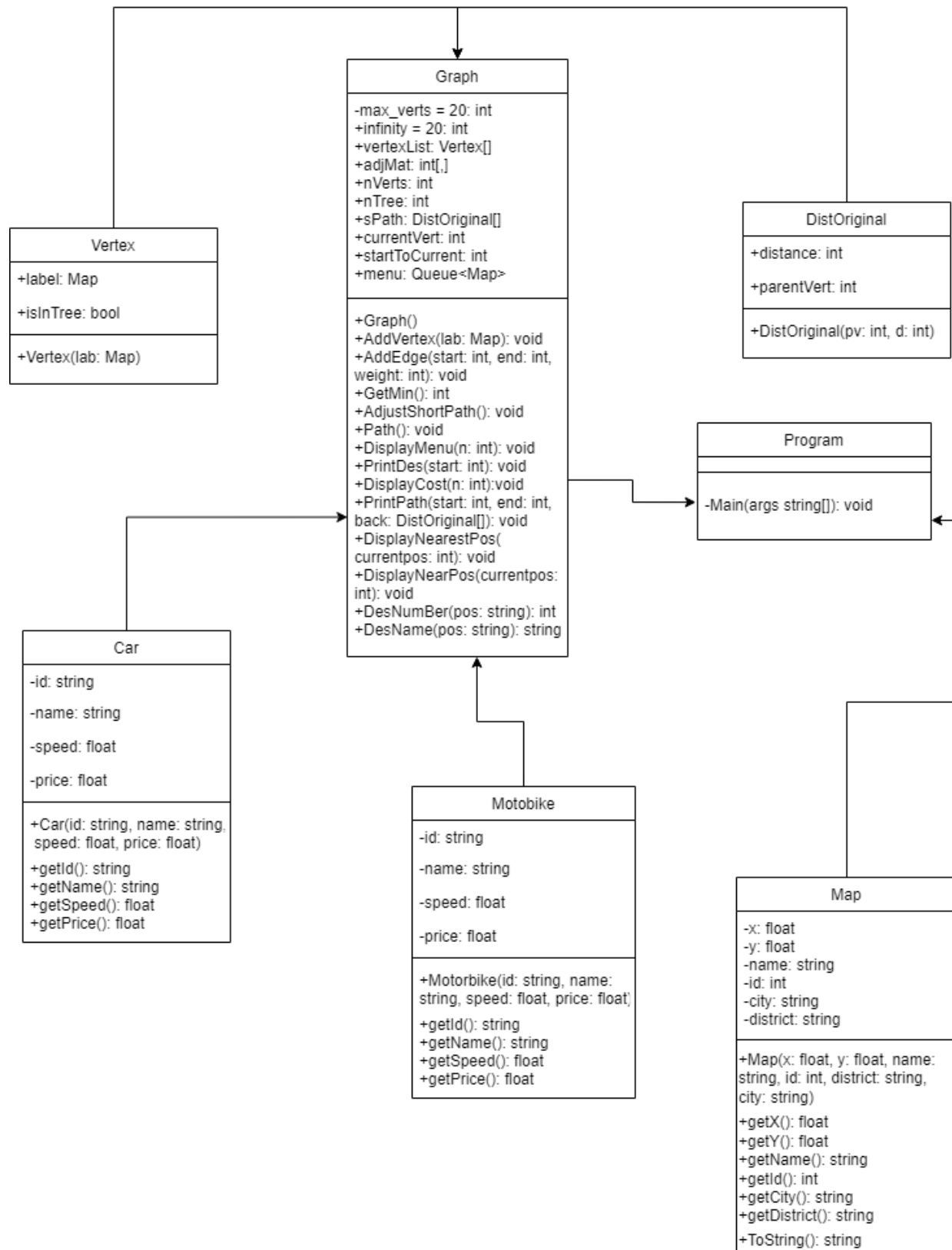
- Hiện thị tuyến đường ngắn nhất:

Chức năng này sẽ cung cấp thông tin về lộ trình đường đi một cách trực quan nhất bằng cách in ra một tuyến đường gồm các địa điểm mà người dùng phải đi qua khi đi từ điểm xuất phát đến điểm kết thúc.

- Hiện thị giá theo phương tiện/ hãng xe:

Trong thực tế khi người dùng tìm kiếm những thông tin liên quan đến lộ trình thì thường sẽ là những người dùng có mong muốn đi xe ô tô công nghệ. Chính vì vậy việc cung cấp thông tin chi phí của lộ trình theo từng loại phương tiện/ hãng xe sẽ rất là hữu ích. Để làm được việc đó ta sẽ thiết kế nên hai class phương tiện với object sẽ là hãng xe. Rồi ta quy ước một mức giá và một khoản thời gian hợp lý tính trên km tùy vào loại phương tiện/ hãng xe.

## 2.2. Sơ Đồ Lớp





## 2.3. Cài Đặt Lớp

Class Vertex	Class DistOriginal
<pre>public class Vertex {     public Map label;     public bool isInTree;     public Vertex(Map lab)     {         label = lab;         isInTree = false;     } }</pre>	<pre>public class DistOriginal {     public int distance;     public int parentVert;     public DistOriginal(int pv, int d)     {         distance = d;         parentVert = pv;     } }</pre>
Class Car	Class Motorbike
<pre>namespace baithi {     public class Car     {         private string id;         private string name;         private float speed;         private float price;         public string getId()         {             return id;         }         public string getName()         {             return name;         }         public float getSpeed()</pre>	<pre>namespace baithi {     public class Motorbike     {         private string id;         private string name;         private float speed;         private float price;         public string getId()         {             return id;         }         public string getName()         {             return name;         }         public float getSpeed()</pre>

```

        {

            return speed;

        }

        public float getPrice()

        {

            return price;

        }

        public Car(string id, string name,
float speed, float price)

        {

            this.id = id;

            this.name = name;

            this.speed = speed;

            this.price = price;

        }

    }

}

```

```

        {

            return speed;

        }

        public float getPrice()

        {

            return price;

        }

        public Motorbike(string id, string
name, float speed, float price)

        {

            this.id = id;

            this.name = name;

            this.speed = speed;

            this.price = price;

        }

    }

}

```

### Class Map

```

public class Map
{
    private float x;
    private float y;
    private string name;
    private int id;
    private string city;
    private string district;
    public float getX()
    {
        return x;
    }
    public float getY()
    {
        return y;
    }
    public string getName()
    {
        return name;
    }
}

```

```

public int getId()
{
    return id;
}
public string getCity()
{
    return city;
}
public string getDistrict()
{
    return district;
}
public Map(float x, float y, string name, int id, string district, string city)
{
    this.x = x;
    this.y = y;
    this.name = name;
    this.id = id;
    this.city = city;
    this.district = district;
}
public override string ToString()
{
    return name + " (" + x + "; " + y + "), " + district + ", " + city + " ";
}
}

```

### Class Graph

```

using System;
using System.Collections.Generic;
namespace baithi
{
    public class Graph
    {
        private const int max_verts = 20;
        int infinity = 1000000;
        Vertex[] vertexList;
        int[,] adjMat;
        int nVerts;
        int nTree;
        DistOriginal[] sPath;
        int currentVert;
        int startToCurrent;
        Queue<Map> menu = new Queue<Map>();

        public Graph() //Cài đặt giá trị ban đầu
        {

```

```

vertexList = new Vertex[max_verts];
adjMat = new int[max_verts, max_verts];
nVerts = 0; nTree = 0;
for (int j = 0; j <= max_verts - 1; j++)
    for (int k = 0; k <= max_verts - 1; k++)
        adjMat[j, k] = infinity;
sPath = new DistOriginal[max_verts];
}

public void AddVertex(Map lab)    //Thêm đỉnh
{
    vertexList[nVerts] = new Vertex(lab);
    menu.Enqueue(lab);
    nVerts++;
}

public void AddEdge(int start, int end, int weight)    //Thêm cạnh
{
    adjMat[start, end] = weight;
    adjMat[end, start] = weight;
}

public int GetMin()    // Tìm vị trí gần với đỉnh cha nhất
{
    int minDist = infinity;
    int indexMin = 0;
    for (int j = 0; j <= nVerts - 1; j++)
        if (!(vertexList[j].isInTree) && sPath[j].distance < minDist)
        {
            minDist = sPath[j].distance; indexMin = j;
        }
    return indexMin;
}

public void AdjustShortPath()    //Thêm đường đi ngắn nhất từ vị trí hiện tại đến hết
{
    int column = 1;
    while (column < nVerts)
        if (vertexList[column].isInTree) column++;
        else
        {
            int currentToFringe = adjMat[currentVert, column];
            int startToFringe = startToCurrent + currentToFringe;
            int sPathDist = sPath[column].distance;
            if (startToFringe < sPathDist)
            {
                sPath[column].parentVert = currentVert;
                sPath[column].distance = startToFringe;
            }
            column++;
        }
}

```

```

    }
    public void Path(string startPos, string endPos)
    {
        int start = DesNumber(startPos);
        int end = DesNumber(endPos);
        int startTree = start;
        vertexList[startTree].isInTree = true;
        nTree = 1;
        // Lưu trọng số từ điểm bắt đầu tới tất cả vị trí
        for (int j = 0; j <= nVerts; j++)
        {
            int tempDist = adjMat[startTree, j];
            sPath[j] = new DistOriginal(startTree, tempDist);
        }
        while (nTree < nVerts)
        {
            int indexMin = GetMin();
            int minDist = sPath[indexMin].distance;
            currentVert = indexMin;
            startToCurrent = sPath[indexMin].distance;
            vertexList[currentVert].isInTree = true;
            nTree++;
            AdjustShortPath();
        }
        nTree = 0;
        for (int j = 0; j <= nVerts - 1; j++)
            vertexList[j].isInTree = false;
    }
    public void DisplayMenu(string startPos, string endPos)
    {
        Begin:
        Console.Clear();
        int m = DesNumber(startPos);
        int n = DesNumber(endPos);
        Path(startPos, endPos);
        System.Console.WriteLine("Điểm xuất phát: " + DesName(startPos));
        System.Console.WriteLine("Điểm đến: " + DesName(endPos));
        System.Console.WriteLine("-----Những thông tin hữu ích về lộ trình bạn cần biết-----");
        System.Console.WriteLine("Quãng đường ngắn nhất: 1");
        System.Console.WriteLine("Tuyến đường ngắn nhất: 2");
        System.Console.WriteLine("Hiển thị giá tiền theo phương tiện/hãng xe: 3");

        Error1:

        System.Console.Write("Nhập lựa chọn của bạn: ");
        int choose = int.Parse(Console.ReadLine());
    }

```

```

        switch (choose)
        {
            case 1:
                System.Console.WriteLine("-----");
                System.Console.WriteLine("Quãng đường ngắn nhất từ {0} đến {1} là: {2}km",
vertexList[m].label.getName(), vertexList[n].label.getName(), sPath[n].distance);
                break;

            case 2:
                System.Console.WriteLine("-----");
                System.Console.WriteLine("Tuyến đường ngắn nhất phải đi là: ");
                PrintPath(m, n, sPath);
                Console.WriteLine();
                break;

            case 3:
                Console.Clear();
                DisplayCost(n);
                break;

            default:
                System.Console.WriteLine("Yêu cầu bạn nhập không đúng vui lòng nhập lại! (1 -
5)");

                goto Error1;
        }
    Error2:
        System.Console.Write("Để xem thêm thông tin về lộ trình ấn phím 1, để tiếp tục ấn
phím 2: ");
        int exit = Int32.Parse(Console.ReadLine());
        switch (exit)
        {
            case 1:
                goto Begin;
            case 2:
                Console.Clear();
                break;
            default:
                System.Console.WriteLine("Yêu cầu bạn nhập không đúng vui lòng nhập lại! (1 -
2)");

                goto Error2;
        }
    }
    public void PrintDes(int start)
    {
        Queue<Map> clone = new Queue<Map>(menu);
        System.Console.WriteLine("-----Các địa điểm-----");
        int button = 1;

```

```

while (clone.Count != 0)
{
    if (start == clone.Peek().getId())
    {
        clone.Dequeue();
    }
    System.Console.WriteLine("{0}: Phím {1}", clone.Dequeue().getName(), button);
    button++;
}
}

public void DisplayCost(int n)
{
    System.Console.WriteLine("-----Các loại phương tiện-----");
    System.Console.WriteLine("Xe taxi: Phím 1");
    System.Console.WriteLine("Xe máy: Phím 2");

Error:
    System.Console.Write("Nhập lựa chọn của bạn: ");
    int numberChoose = Int32.Parse(Console.ReadLine());
    switch (numberChoose)
    {
        case 1:
            List<Car> listCar = new List<Car>();
            Car grabCar = new Car("13203", "Taxi Grab", 45f, 5000f);
            listCar.Add(grabCar);
            Car uberCar = new Car("12131", "Taxi Uber", 42f, 4800f);
            listCar.Add(uberCar);
            Car mailinhCar = new Car("14205", "Taxi Mai Linh", 50f, 5500f);
            listCar.Add(mailinhCar);
            System.Console.WriteLine("-----Các hãng xe taxi công nghệ-----");

            System.Console.WriteLine("Grab (5000đ/1km): 0");
            System.Console.WriteLine("Uber (4800đ/1km): 1");
            System.Console.WriteLine("Taxi Mai Linh (5500đ/1km): 2");

Error1:
            System.Console.Write("Nhập lựa chọn của bạn: ");
            int carChoose = Int32.Parse(Console.ReadLine());
            System.Console.WriteLine("-----");
            Random random1 = new Random();
            switch (carChoose)
            {
                case 0:
                    System.Console.Write("Tài xế Grab gần bạn nhất là: ");
                    System.Console.WriteLine(random1.Next(0, 1) + "," + random1.Next(1,
9) + "km");

```

```

        float grabCarCost = sPath[n].distance *
listCar[carChoose].getPrice();
        double grabCarTime = Math.Round((sPath[n].distance /
listCar[carChoose].getSpeed()) * 60);
        System.Console.WriteLine("Tiền đi xe hơi của hãng " +
listCar[carChoose].getName() + " là: " + grabCarCost + "đ");
        System.Console.WriteLine("Thời gian đi xe hơi của hãng " +
listCar[carChoose].getName() + " là: " + grabCarTime + " phút");
        break;
    case 1:
        System.Console.Write("Tài xế Uber gần bạn nhất là: ");
        System.Console.WriteLine(random1.Next(0, 1) + "," + random1.Next(1,
9) + "km");

        float uberCarCost = sPath[n].distance *
listCar[carChoose].getPrice();
        double uberCarTime = Math.Round((sPath[n].distance /
listCar[carChoose].getSpeed()) * 60);
        System.Console.WriteLine("Tiền đi xe hơi của hãng " +
listCar[carChoose].getName() + " là: " + uberCarCost + "đ");
        System.Console.WriteLine("Thời gian đi xe hơi của hãng " +
listCar[carChoose].getName() + " là: " + uberCarTime + " phút");
        break;
    case 2:
        System.Console.Write("Tài xế Mai Linh gần bạn nhất là: ");
        System.Console.WriteLine(random1.Next(0, 1) + "," + random1.Next(1,
9) + "km");

        float mailinhCarCost = sPath[n].distance *
listCar[carChoose].getPrice();
        double mailinhCarTime = Math.Round((sPath[n].distance /
listCar[carChoose].getSpeed()) * 60);
        System.Console.WriteLine("Tiền đi xe hơi của hãng " +
listCar[carChoose].getName() + " là: " + mailinhCarCost + "đ");
        System.Console.WriteLine("Thời gian đi xe hơi của hãng " +
listCar[carChoose].getName() + " là: " + mailinhCarTime + " phút");
        break;
    default:
        System.Console.WriteLine("-----");
        System.Console.WriteLine("Yêu cầu bạn nhập không đúng vui lòng nhập
lại! (0 - 2)");

        goto Error1;
    }
    break;
case 2:
    List<Motorbike> listMotorbike = new List<Motorbike>();
    Motorbike grabMotorbike = new Motorbike("21023", "Xe máy Grab", 30f, 2500f);
    listMotorbike.Add(grabMotorbike);
    Motorbike uberMotorbike = new Motorbike("25037", "Xe máy Uber", 32f, 2600f);

```



```

        listMotorbike.Add(uberMotorbike);
        Motorbike beMotorBike = new Motorbike("24306", "Xe máy BE", 29f, 2400f);
        listMotorbike.Add(beMotorBike);
        System.Console.WriteLine("-----Các hãng xe máy công nghệ-----
");

        System.Console.WriteLine("Grab (2500đ/1km): Phím 0");
        System.Console.WriteLine("Uber (2600đ/1km): Phím 1");
        System.Console.WriteLine("BE (2400đ/1km): Phím 2");

        Error2:
        System.Console.Write("Nhập lựa chọn của bạn: ");
        int motorBikeChoose = Int32.Parse(Console.ReadLine());
        System.Console.WriteLine("-----");
        Random random2 = new Random();
        switch (motorBikeChoose)
        {
            case 0:
                System.Console.Write("Tài xế Grab gần bạn nhất cách: ");
                System.Console.WriteLine(0 + "," + random2.Next(1, 9) + "km");
                float grabMotorbikeCost = sPath[n].distance *
listMotorbike[motorBikeChoose].getPrice();
                double grabMotorbikeTime = Math.Round((sPath[n].distance /
listMotorbike[motorBikeChoose].getSpeed()) * 60);
                System.Console.WriteLine("Tiền đi xe máy của hãng " +
listMotorbike[motorBikeChoose].getName() + " là: " + grabMotorbikeCost + "đ");
                System.Console.WriteLine("Thời gian đi xe của hãng " +
listMotorbike[motorBikeChoose].getName() + " là: " + grabMotorbikeTime + " phút");
                break;
            case 1:
                System.Console.Write("Tài xế Uber gần bạn nhất cách: ");
                System.Console.WriteLine(0 + "," + random2.Next(1, 9) + "km");
                float uberMotorbikeCost = sPath[n].distance *
listMotorbike[motorBikeChoose].getPrice();
                double uberMotorbikeTime = Math.Round((sPath[n].distance /
listMotorbike[motorBikeChoose].getSpeed()) * 60);
                System.Console.WriteLine("Tiền đi xe máy của hãng " +
listMotorbike[motorBikeChoose].getName() + " là: " + uberMotorbikeCost + "đ");
                System.Console.WriteLine("Thời gian đi xe của hãng " +
listMotorbike[motorBikeChoose].getName() + " là: " + uberMotorbikeTime + " phút");
                break;
            case 2:
                System.Console.Write("Tài xế Be gần bạn nhất cách: ");
                System.Console.WriteLine(0 + "," + random2.Next(1, 9) + "km");
                float beMotorbikeCost = sPath[n].distance *
listMotorbike[motorBikeChoose].getPrice();
                double beMotorbikeTime = Math.Round((sPath[n].distance /
listMotorbike[motorBikeChoose].getSpeed()) * 60);

```

```

        System.Console.WriteLine("Tiền đi xe máy của hãng " +
listMotorbike[motorBikeChoose].getName() + " là: " + beMotorbikeCost + "đ");
        System.Console.WriteLine("Thời gian đi xe của hãng " +
listMotorbike[motorBikeChoose].getName() + " là: " + beMotorbikeTime + " phút");
        break;
    default:
        System.Console.WriteLine("-----");
        System.Console.WriteLine("Yêu cầu bạn nhập không đúng vui lòng nhập
lại! (0 - 2)");
        goto Error2;
    }
    break;
default:
    System.Console.WriteLine("-----");
    System.Console.WriteLine("Yêu cầu bạn nhập không đúng vui lòng nhập lại! (1 -
2)");
    goto Error;
}

}

public void PrintPath(int start, int end, DistOriginal[] back)
{
    if (start == end)
    {
        System.Console.Write(vertexList[start].label.getName());
    }
    else
    {
        PrintPath(start, back[end].parentVert, back);
        System.Console.Write(" -> ");
        System.Console.Write(vertexList[end].label.getName());
    }
}

public void DisplayNearestPos(int currentpos)
{
    int min = infinity;
    int nearestpos = 0;

    System.Console.WriteLine("Địa điểm gần {0} nhất là: ",
vertexList[currentpos].label.getName());
    for (int i = 0; i < adjMat.GetLength(1); i++)
    {
        if (adjMat[currentpos, i] == min && adjMat[currentpos, i] != infinity)
        {
            System.Console.WriteLine(vertexList[nearestpos].label.getName());
        }
        if (adjMat[currentpos, i] <= min && adjMat[currentpos, i] != 0)

```

```

        {
            min = adjMat[currentpos, i];
            nearestpos = i;
        }
    }
    System.Console.WriteLine(vertexList[nearestpos].label.getName());
}
public void DisplayNearPos(int currentpos)
{
    Console.WriteLine("Những địa điểm xung quanh {0} là: ",
vertexList[currentpos].label.getName());
    for (int i = 0; i < adjMat.GetLength(1); i++)
    {
        if (adjMat[currentpos, i] != 0 && adjMat[currentpos, i] < infinity)
        {
            Console.WriteLine(vertexList[i].label.getName());
        }
    }
}
public int DesNumber(string pos)
{
    switch (pos.ToLower())
    {
        case "bệnh viện đại học y dược": return 0;
        case "trường đại học kinh tế tp.hcm": return 1;
        case "chợ bến thành": return 2;
        case "bệnh viện hùng vương": return 3;
        case "trường thpt mạc đình chi": return 4;
        case "trung tâm mua sắm aeon mall bình tân": return 5;
        case "đại học khoa học tự nhiên": return 6;
        case "đài truyền hình htv": return 7;
        default: return -1;
    }
}
public string DesName(string pos)
{
    switch (pos.ToLower())
    {
        case "bệnh viện đại học y dược": return "Bệnh viện đại học Y Dược";
        case "trường đại học kinh tế tp.hcm": return "Trường Đại học kinh tế TP.HCM";
        case "chợ bến thành": return "Chợ Bến Thành";
        case "bệnh viện hùng vương": return "Bệnh viện Hùng Vương";
        case "trường thpt mạc đình chi": return "Trường THPT Mạc Đình Chi";
        case "trung tâm mua sắm aeon mall bình tân": return "Trung tâm mua sắm AEON Mall
Bình Tân";
        case "đại học khoa học tự nhiên": return "Đại học Khoa học tự nhiên TPHCM";
        case "đài truyền hình htv": return "Đài truyền hình HTV";
    }
}

```

```
        default: return "";  
    }  
}  
}
```

## CHƯƠNG 3. THIẾT KẾ GIAO DIỆN

### 3.1. Giao Diện Menu Chính

```
=====ĐỒ ÁN MÔN HỌC CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT=====
  ĐỀ TÀI: ĐỒ THỊ VÀ ỨNG DỤNG TRONG QUẢN LÝ LỘ TRÌNH ĐƯỜNG ĐI
    NHÓM THỰC HIỆN: BẢO, DẬU, ĐẠT, THIÊN
      *****
-----PHẦN MỀM QUẢN LÝ LỘ TRÌNH ĐƯỜNG ĐI-----
      _____Các địa điểm trên bản đồ_____
1. Bệnh viện đại học Y Dược
2. Trường đại học Kinh tế TP.HCM
3. Chợ Bến Thành
4. Bệnh viện Hùng Vương
5. Trường THPT Mạc Đĩnh Chi
6. Trung tâm mua sắm AEON Mall Bình Tân
7. Đại học Khoa học tự nhiên TP.HCM
8. Đài truyền hình HTV

=====
Tìm địa điểm theo từ khóa: Phím 1
Tìm địa điểm gần nhất: Phím 2
Tìm địa điểm xung quanh: Phím 3
Tìm thông tin theo lộ trình: Phím 4
Nhập lựa chọn của bạn: █
```

### 3.2. Chi Tiết Chức Năng

#### 1. Tìm địa điểm theo từ khóa

```
Nhập từ khóa cần tìm kiếm: Bệnh viện
-----
Địa điểm bạn cần tìm là:
Bệnh viện Đại học Y dược
Bệnh viện Hùng Vương
Để tiếp tục sử dụng chương trình ấn phím 1, để thoát chương trình ấn phím 2: █
```

```
-----Các địa điểm-----
Bệnh viện Đại học Y dược: 1
Đại học kinh tế TP.HCM CSB: 2
Chợ Bến Thành: 3
Bệnh viện Hùng Vương: 4
Trường THPT Mạc Đĩnh Chi: 5
Trung tâm mua sắm AEON Mall Bình Tân: 6
Đại học Khoa học tự nhiên TPHCM: 7
Đài truyền hình HTV: 8
Nhập địa điểm bạn muốn: 3
-----
Địa điểm gần Chợ Bến Thành nhất là: Đài truyền hình HTV
Để tiếp tục sử dụng chương trình ấn phím 1, để thoát chương trình ấn phím 2: █
```

### 3. Tìm địa điểm xung quanh

```
-----Các địa điểm-----  
Bệnh viện Đại học Y dược: 1  
Đại học kinh tế TP.HCM CSB: 2  
Chợ Bến Thành: 3  
Bệnh viện Hùng Vương: 4  
Trường THPT Mạc Đĩnh Chi: 5  
Trung tâm mua sắm AEON Mall Bình Tân: 6  
Đại học Khoa học tự nhiên TP HCM: 7  
Đài truyền hình HTV: 8  
Nhập địa điểm bạn muốn: 7  
-----  
Những địa điểm xung quanh Đại học Khoa học tự nhiên TP HCM là:  
Đại học kinh tế TP.HCM CSB  
Chợ Bến Thành  
Bệnh viện Hùng Vương  
Trung tâm mua sắm AEON Mall Bình Tân  
Đài truyền hình HTV  
Để tiếp tục sử dụng chương trình ấn phím 1, để thoát chương trình ấn phím 2: █
```

### 4. Tìm thông tin theo lộ trình

Chọn địa điểm xuất phát và kết thúc:

```
Nhập địa điểm xuất phát: Bệnh viện hùng vương  
Nhập điểm đến: trường đại học kinh tế tp.hcm █
```

Các chức năng:

```
Điểm xuất phát: Bệnh viện Hùng Vương  
Điểm đến: Trường Đại học kinh tế TP.HCM  
-----Những thông tin hữu ích về lộ trình bạn cần biết-----  
Quãng đường ngắn nhất: 1  
Tuyến đường ngắn nhất: 2  
Hiển thị giá tiền theo phương tiện/hãng xe: 3  
Nhập lựa chọn của bạn: █
```

#### a. Quãng đường ngắn nhất

```
-----  
Quãng đường ngắn nhất từ Bệnh viện Hùng Vương đến Đại học kinh tế TP.HCM CSB là: 6km  
Để xem thêm thông tin về lộ trình ấn phím 1, để tiếp tục ấn phím 2: █
```

#### b. Tuyến đường ngắn nhất

```
-----  
Tuyến đường ngắn nhất phải đi là:  
Bệnh viện Hùng Vương -> Bệnh viện Đại học Y dược -> Đại học kinh tế TP.HCM CSB  
Để xem thêm thông tin về lộ trình ấn phím 1, để tiếp tục ấn phím 2: █
```

#### c. Hiển thị giá theo phương tiện/ hãng xe

```
-----Các loại phương tiện-----
Xe taxi: Phím 1
Xe máy: Phím 2
Nhập lựa chọn của bạn: 1
-----Các hãng xe taxi công nghệ-----
Grab (5000đ/1km): 0
Uber (4800đ/1km): 1
Taxi Mai Linh (5500đ/1km): 2
Nhập lựa chọn của bạn: 0
-----
Tài xế Grab gần bạn nhất là: 0,7km
Tiền đi xe hơi của hãng Taxi Grab là: 30000đ
Thời gian đi xe hơi của hãng Taxi Grab là: 8 phút
Để xem thêm thông tin về lộ trình ấn phím 1, để tiếp tục ấn phím 2: █
```

## CHƯƠNG 4. THẢO LUẬN & ĐÁNH GIÁ

### 4.1. Các Kết Quả Nhận Được

- Người dùng có thể tự nhập địa điểm cần tìm hoặc có thể tìm kiếm theo một vài từ khóa đã được gợi ý sẵn.
- Người dùng có thể biết được địa điểm gần nhất với vị trí mà người dùng đã chọn.
- Đồng thời chương trình cũng có thể cung cấp thông tin những địa điểm xung quanh địa điểm mà người dùng chọn.
- Chương trình cũng cho phép người dùng có thể chọn điểm bắt đầu và điểm kết thúc để tùy chọn lộ trình đường đi theo ý muốn. Những thông tin mà người dùng có thể tham khảo trên lộ trình đường đi:
  - + Đường đi ngắn nhất đi từ điểm xuất phát đến điểm kết thúc.
  - + Tuyến đường gồm các địa điểm cần phải đi qua.
  - + Hiện thị chi phí theo phương tiện/hãng.

### 4.2. Một Số Tồn Tại

- Không thể nhập quá nhiều địa điểm, bởi việc nhập nhiều sẽ khiến dữ liệu bị nhiễu, dẫn đến việc tốn nhiều dòng code cũng như dung lượng hơn để thực hiện các chức năng.
- Vì việc cài đặt đồ thị trong hàm Main ở program.cs là hoàn toàn thủ công nên nếu mở rộng đồ thị thêm sẽ khiến cho số lượng dòng code trở nên rất lớn dẫn đến khó bảo trì và tái sử dụng.

### 4.3. Hướng Phát Triển

- Chúng ta có thể mở rộng thêm nhiều địa điểm hơn giữa các tỉnh thành hoặc giữa các quốc gia trong tương lai.
- Đồng thời cũng có thể phát triển thêm chức năng tìm đường đi ngắn thứ hai nếu đường đi ngắn nhất ban đầu bị gián đoạn (chướng ngại vật, tắc đường, ...).



# PHỤ LỤC

- Đưa Link Toàn Bộ Mã Nguồn Lên GitHub và Đưa Link Vào Đây

[bao201102/Doan\\_CTDL \(github.com\)](https://github.com/bao201102/Doan_CTDL)

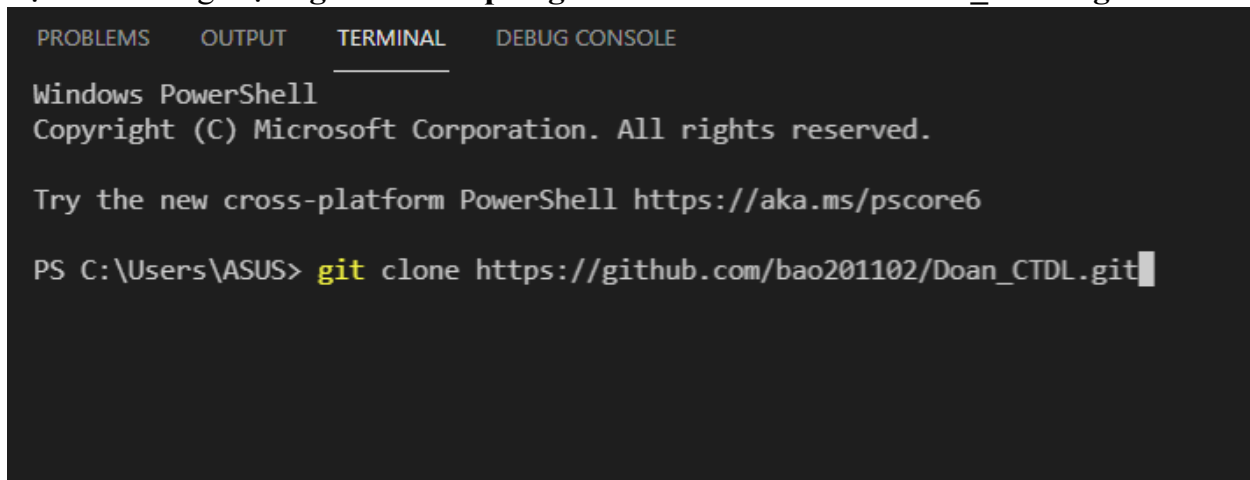
- Viết Hướng Dẫn Cách Cài Đặt Để Chạy

Bước 1: Tải git về máy tính tại đây (chọn theo hệ điều hành phù hợp) và cài đặt.

Bước 2:

- **Đối với Visual Studio Code**

Tại Terminal gõ lệnh **git clone https://github.com/bao201102/Doan\_CTDL.git**



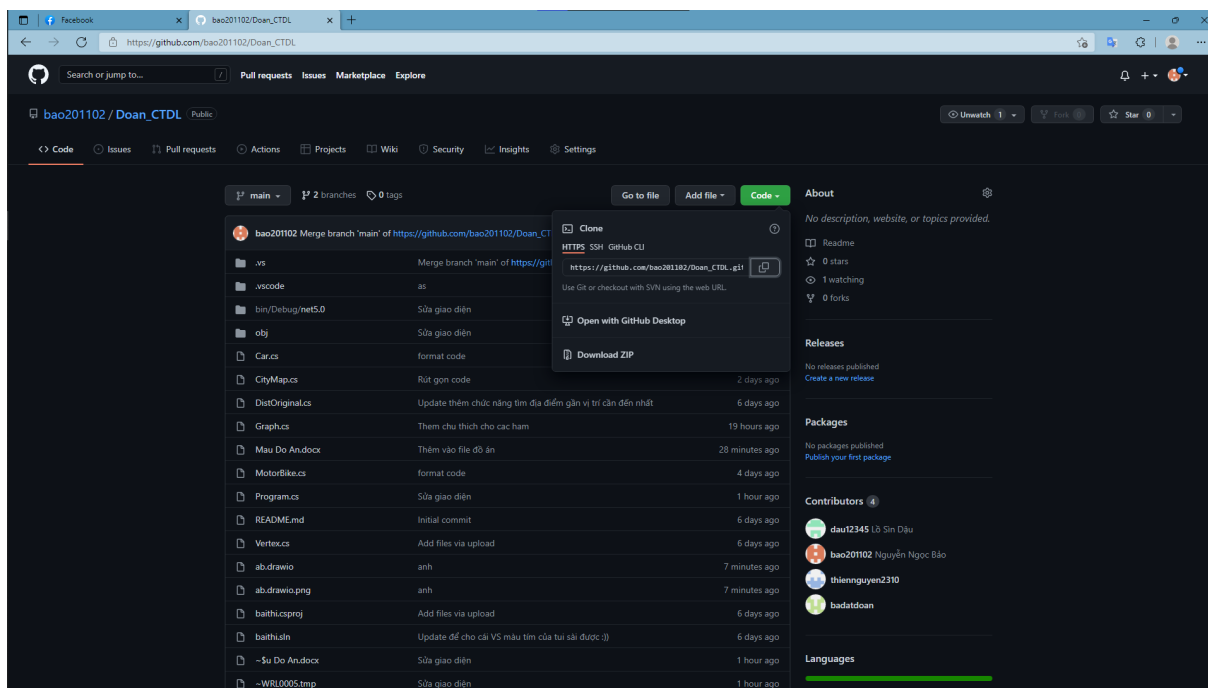
```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

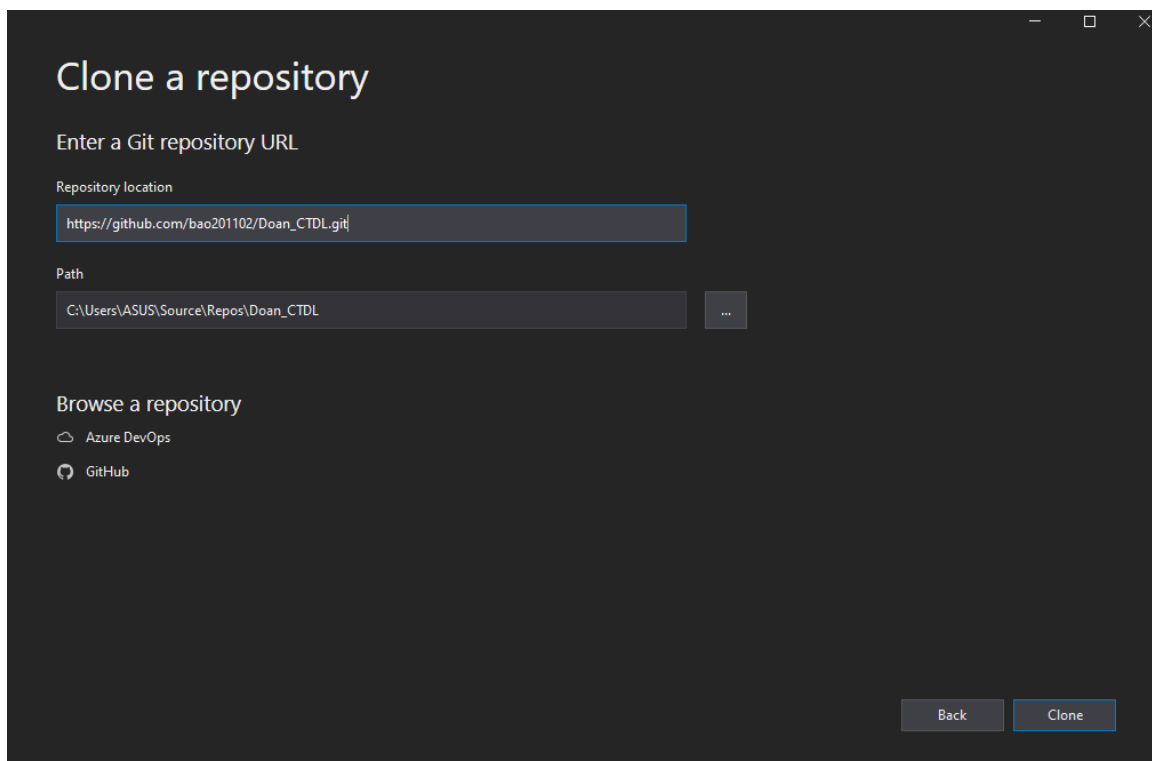
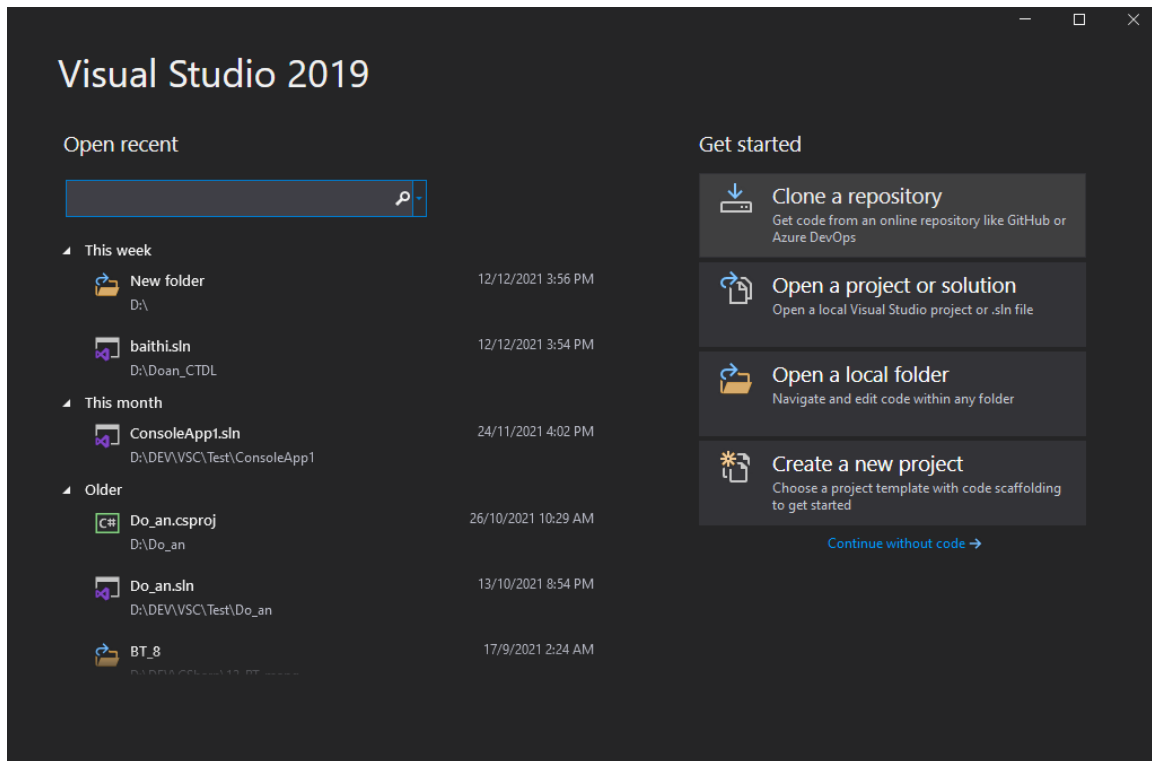
PS C:\Users\ASUS> git clone https://github.com/bao201102/Doan_CTDL.git
```

(Lấy link theo như hình).



- **Đổi với Visual Studio**

Nhập vào **Clone a repository** sau đó lấy link như trên



Bước 3: Sau khi đã clone source code thì bắt đầu chạy chương trình

- Phân Công Công Việc

Thành Viên	Nhiệm Vụ
Nguyễn Ngọc Bảo	PrintPath Giao diện (program.cs) Thiết kế Class Diagram Chương 2 bản báo cáo
Lồ Sìn Dậu	DisplayNearestPos DisplayNearPos Tối ưu code Chương 1 bản báo cáo
Đoàn Trần Bá Đạt	DisplayCost (Class Car, Motorbike, Map) Cài đặt cấu trúc đồ thị Tùy chọn lộ trình (điểm xuất phát, điểm kết thúc) Đường đi ngắn nhất Chương 3 bản báo cáo
Nguyễn Hoàn Thiện	DisplayMenu PrintDes Tìm kiếm theo từ khóa Chương 4 bản báo cáo

**Lưu ý:** nhiệm vụ cần phân công rõ ràng, ko có kiểu là “cùng nhau” làm cái này !

## TÀI LIỆU THAM KHẢO

1. <https://www.howkteam.vn/>
2. <https://xuanthulab.net/>
3. Tìm hiểu về cách thiết kế Class Diagram (viblo.asia)