**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC NHA TRANG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**Nguyễn Ngọc Hồng Hân**

**CÀI ĐẶT THUẬT TOÁN**

**TÌM ĐƯỜNG ĐI NGẮN NHẤT VÀ MINH HỌA**

**THỰC TẬP CƠ SỞ**

(Ngành: Công nghệ thông tin)

Nha Trang – 2022

# BỘ GIÁO DỤC VÀ ĐÀO TẠO

**TRƯỜNG ĐẠI HỌC NHA TRANG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**Nguyễn Ngọc Hồng Hân**

**CÀI ĐẶT THUẬT TOÁN**

**TÌM ĐƯỜNG ĐI NGẮN NHẤT VÀ MINH HỌA**

**THỰC TẬP CƠ SỞ**

(Ngành: Công nghệ thông tin)

CÁN BỘ HƯỚNG DẪN: ThS.GV. Nguyễn Thị Hương Lý

Nha Trang – 2022

MỤC LỤC

[TRANG BÌA 1](#_Toc93649969)

[MỤC LỤC 2](#_Toc93649970)

[MỞ ĐẦU 4](#_Toc93649971)

[1. Mục đích, lý do chọn đề tài: 4](#_Toc93649972)

[1.1. Lý do chọn đề tài: 4](#_Toc93649973)

[1.2. Mục đích: 4](#_Toc93649974)

[2. Nhiệm vụ: 4](#_Toc93649975)

[3. Phương pháp nghiên cứu 5](#_Toc93649976)

[3.1. Phương pháp nghiên cứu lý luận: 5](#_Toc93649977)

[3.2. Phương pháp chuyên gia: 5](#_Toc93649978)

[3.3. Phương pháp thực nghiệm: 5](#_Toc93649979)

[4. Đối tượng, phạm vi nghiên cứu: 5](#_Toc93649980)

[5. Ý nghĩa khoa học và thực tiễn: 5](#_Toc93649981)

[6. Cấu trúc đề tài: 6](#_Toc93649982)

[Chương 1: CƠ SỞ LÝ THUYẾT 7](#_Toc93649983)

[1.1. Định nghĩa: 7](#_Toc93649984)

[1.1.1. Định nghĩa đồ thị: 7](#_Toc93649985)

[1.1.2. Phân loại đồ thị: 7](#_Toc93649986)

[1.1.3. Các thuật ngữ cơ bản: 8](#_Toc93649987)

[1.2. Hành trình, đường đi, chu trình, đồ thị liên thông: 10](#_Toc93649988)

[1.3. Ma trận kề, ma trận trọng số: 11](#_Toc93649989)

[1.3.1. Ma trận kề: 11](#_Toc93649990)

[1.3.2. Ma trận trọng số: 12](#_Toc93649991)

[1.4. Một số đồ thị đặc biệt: 12](#_Toc93649992)

[1.4.1. Đồ thị đầy đủ: 12](#_Toc93649993)

[1.4.2. Đồ thị vòng: 13](#_Toc93649994)

[1.4.3. Đồ thị hai phía, đồ thị hai phía đầy đủ: 13](#_Toc93649995)

[Chương 2: BÀI TOÁN DƯỜNG ĐI NGẮN NHẤT 15](#_Toc93649996)

[2.1. Khái niệm: 15](#_Toc93649997)

[2.2. Đường đi ngắn nhất xuất phát từ một đỉnh: 16](#_Toc93649998)

[2.3. Đường đi ngắn nhất trong đồ thị có trọng số cạnh không âm – Thuật toán Dijkstra 17](#_Toc93649999)

[2.3.1. Mô tả thuật toán 17](#_Toc93650000)

[2.3.2. Nội dung thuật toán 19](#_Toc93650001)

[2.4. Độ phức tạp thuật toán: 20](#_Toc93650002)

[Chương 3: XÂY DỰNG CHƯƠNG TRÌNH 21](#_Toc93650003)

[3.1. Cấu tạo: 21](#_Toc93650007)

[3.1.1. Class DuongDi 21](#_Toc93650008)

[3.1.2. Class Dijkstra 21](#_Toc93650009)

[3.1.3. Class Form1 22](#_Toc93650010)

[3.2. Thiết kế giao diện 23](#_Toc93650011)

[KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 31](#_Toc93650012)

[1. Kết luận: 31](#_Toc93650013)

[2. Hướng phát triển: 31](#_Toc93650014)

[TÀI LIỆU THAM KHẢO 32](#_Toc93650015)

# MỞ ĐẦU

1. **Mục đích, lý do chọn đề tài:**
   1. **Lý do chọn đề tài:**

Trong những năm gần đây, các phương pháp tối ưu hoá ngày càng được áp dụng sâu rộng và hiệu quả vào các ngành giao thông vận tải, mạng viễn thông, kinh tế, kỹ thuật, công nghệ thông tin và các ngành khoa học khác. Các phương pháp tối ưu là công cụ đắc lực giúp người làm quyết định có những giải pháp tốt nhất về định lượng và định tính.

Một trong những lớp bài toán tối ưu đầu tiên được nghiên cứu là thuật toán giải bài toán tìm đường đi ngắn nhất có trọng số xác định. Bài toán tìm đường đi ngắn nhất là vấn đề quan trọng trong lý thuyết đồ thị, nó đã được nghiên cứu từ lâu và có nhiều ứng dụng trong nhiều ngành khoa học nói chung, khoa học máy tính và hệ thống thông tin nói riêng như giải thuật Dijkstra đã được phát triển để tìm đường đi ngắn nhất và ngày nay đã được nhiều nhà nghiên cứu nhằm cải tiến xây dựng giải thuật giải bài toán tìm đường đi ngắn nhất với dữ liệu mờ dạng khoảng. Bài toán tìm đường đi ngắn nhất cũng được phát triển rộng rãi và trở thành một chuyên ngành toán học từ những năm 1950. Giải đáp những câu hỏi đặt ra mà tìm đường đi ngắn nhất với các cạnh có trọng số xác định.

Thuật toán Dijkstra là một thuật toán rất phổ biến vì tính hiệu quả của nó. Nó tìm ra đường đi ngắn nhất với chi phí nhỏ nhất. Vì thế nó có một ý nghĩa to lớn trong thực tế.

Chính vì vậy, sau một thời gian tìm hiểu em quyết định chọn đề tài: *“Cài đặt thuật toán tìm đường đi ngắn nhất Dijkstra và xây dựng chương trình minh họa”* làm đề tài của mình.

* 1. **Mục đích:**

Mục đích của đề tài này là nghiên cứu thuật toán Dijkstra để tìm đường đi ngắn nhất.

1. **Nhiệm vụ:**

Đề tài *“Cài đặt thuật toán tìm đường đi ngắn nhất Dijkstra và xây dựng chương trình minh họa”* với các nhiệm vụ chính sau:

* + Phải nêu được các khái niệm cơ bản về lý thuyết đồ thị, thuật toán tìm đường đi ngắn nhất, nhằm mục đích mô phỏng quá trình hiển thị đường đi dựa trên ngôn ngữ Visual C#.
  + Tìm hiểu thuật toán Dijkstra áp dụng trong bài toán tìm đường đi ngắn nhất từ một đỉnh đến tất cả các đỉnh.
* Xây dựng chương trình mô phỏng.
  + Yêu cầu:
    - Mức 1: Tổ chức dữ liệu và cài đặt được thuật toán. Chương trình cần có các chức năng Thêm mới đồ thị (từ bàn phím hoặc từ file), Tìm đường và xuất kết quả dạng văn bản và dạng file văn bản.
    - Mức 2: Thể hiện đồ họa đồ thị và đường đi ngắn nhất.

1. **Phương pháp nghiên cứu**
   1. **Phương pháp nghiên cứu lý luận:**

Nghiên cứu qua việc giáo trình và các tài liệu liên quan nhằm xây dựng cơ sở lý thuyết và các biện pháp cần thiết để giải quyết các vấn đề của đề tài.

* 1. **Phương pháp chuyên gia:**

Tham khảo ý kiến của cô ThS.GV. Nguyễn Thị Hương Lý để có thể thiết kế chương trình phù hợp với yêu cầu thực tiễn.

* 1. **Phương pháp thực nghiệm:**

Thông qua quan sát thực tế, yêu cầu của cơ sở, những lý luận được nghiên cứu và kết quả đạt được qua những phương pháp trên.

1. **Đối tượng, phạm vi nghiên cứu:**

Đối tượng nghiên cứu của đề tài là thuật toán Dijkstra. Trên cơ sở thuật toán Dijkstra, đề tài xây dựng chương trình tìm đường đi ngắn nhất từ một đỉnh đến tất cả các đỉnh trên một đồ thị theo một cách tuần tự.

1. **Ý nghĩa khoa học và thực tiễn:**

Đề tài “Cài đặt thuật toán tìm đường đi ngắn nhất Dijkstra và xây dựng chương trình minh họa” giúp ta hiểu thuật toán Dijkstra và bài toán tìm đường đi ngắn nhất từ một đỉnh đến tất cả các đỉnh còn lại trong một đồ thị, qua đó mô phỏng chương trình một cách cụ thể giúp ta hiểu rõ hơn thuật toán và có thể xây dựng những chương trình ứng dụng cụ thể từ thuật toán Dijkstra.

1. **Cấu trúc đề tài:**

Ngoài phần mở đầu và kết luận, đề tài gồm có 3 chương:

* Chương 1: Cơ sở lý thuyết.
* Chương 2: Bài toán đường đi ngắn nhất.
* Chương 3: Xây dựng chương trình.

# Chương 1: CƠ SỞ LÝ THUYẾT

* 1. **Định nghĩa:**
     1. **Định nghĩa đồ thị:**

***Định nghĩa 1:*** *Đồ thị* là một cấu trúc rời rạc bao gồm hữu hạn các điểm gọi là đỉnh của đồthị, cùng với tập các cạnh (hay cung), mỗi cạnh nối một cặp đỉnh nào đó.

***Ký hiệu:***

* là đồ thị, trong đó: V là tập các đỉnh, E là tập các cạnh.
* là số đỉnh, là số cạnh của G.

***Khái niệm:***

* Khuyên: cạnh có hai đầu trùng nhau (cùng một đỉnh).
* Đỉnh cô lập: đỉnh có bậc 0.
* Đỉnh treo: đỉnh có bậc 1.

***Ví dụ 1:*** Xét đồ thị trên Hình 1.1. Ta có:

=5, =5

Đỉnh cô lập:

Đỉnh treo:

Cạnh khuyên:

a

b

c

d

e

Hình 1.1 Đồ thị G = (V,E)

* + 1. **Phân loại đồ thị:**

***Định nghĩa 2:*** *Đơn đồ thị* là đồ thị không cho phép đa cạnh (và không cho phép khuyên nếu là đồ thị có hướng).

***Định nghĩa 3:*** *Đa đồ thị* là đồ thị cho phép đa cạnh (và đôi khi cả khuyên).

***Định nghĩa 4:*** *Giả đồ thị* được dùng để hàm ý cả đa cạnh và khuyên đều được phép.

Đơn đồ thị

Đa đồ thị

Giả đồ thị

Hình 1.2

***Ví dụ thực tế về đồ thị:***

* + Đồ thị biểu diễn hệ thống giao thông là *giả đồ thị*.
  + Đồ thị biểu diễn sơ đồ cấp điện hoặc sơ đồ hệ thống cấp nước *không thể là giả đồ thị*.
  + Đồ thị biểu diễn một bản đồ là một *giả đồ thị*, trong đó các cạnh của đồ thị là các đường biên giới giữa hai nước, các đỉnh là điểm giao giữa các đường biên đó.
    1. **Các thuật ngữ cơ bản:**
* Trong mục này, chúng ta sẽ trình bày một số thuật ngữ cơ bản của lý thuyết đồ thị. Trước tiên, ta xét các thuật ngữ mô tả các đỉnh và cạnh của đồ thị vô hướng. Để có thể biết có bao nhiêu cạnh liên thuộc với một đỉnh, ta có định nghĩa sau:

***Định nghĩa 5:*** Trong một đồ thị, *bậc của một đỉnh* là số cạnh xuất phát từ đỉnh đó. Bậc của đỉnh v được ký hiệu là deg(v).

***Ví dụ 2:*** Xét đồ thị trên Hình 1.3, ta có:

a

e

b

c

f

d

deg(a) = 3,

deg(c) = 1,

deg(d) = 0 .

Hình 1.3

Bậc của đỉnh có tính chất sau:

***Định lý 1:*** (Định lý Bắt tay). Trong một đồ thị, tổng bậc của tất cả các đỉnh bằng hai lần số cạnh:

.

***Ví dụ 3:*** Một đồ thị có 7 đỉnh và mỗi đỉnh đều có bậc là 6. Đồ thị này có bao nhiêu cạnh? Giải. Gọi m, n lần lượt là số cạnh và số đỉnh của đồ thị đã cho. Theo Định lý 1 ta có 2m = 6n. Do đó m = 3n = 3 \* 7 = 21.

***Hệ quả:*** Trong một đồ thị, số đỉnh bậc lẻ là một số chẵn.

***Định lý 2:*** Trong đồ thị n đỉnhbao giờ cũng có ít nhất hai đỉnh cùng bậc.

* Ta xét các thuật ngữ tương tự cho đồ thị có hướng:

***Định nghĩa 6:*** Đồ thị có các cạnh được định hướng (được gọi là *cung*). Mỗi đỉnh có khái niệm *bán bậc ra, bán bậc vào*. Ký kiệu: .

***Ví dụ 4:*** Hình 1.4 là đồ thị có hướng, (a, b), (b, c), (f, e) là các cung, .

►

►

►

►

►

►

►

a

b

c

d

e

f

Hình 1.4 Đồ thị có hướng

***Định lý 3:*** Giả sử là đồ thị có hướng, khi đó:

* 1. **Hành trình, đường đi, chu trình, đồ thị liên thông:**

***Định nghĩa 7***: Một *hành trình* trong đồ thị là một dãy cạnh nối tiếp.

Hành trình , và là *đỉnh đầu* và *đỉnh cuối*. Số được gọi là *độ dài* của W.

Hành trình được gọi là *khép kính* nếu .

***Ví dụ 5:*** Xét đồ thị Hình 1.5. Hành trình

a

b

c

d

e

f

g

* *(a, b, c, f)* có độ dài 3, trong đó *a* là đỉnh đầu, *f* là đỉnh cuối.
* *(a, b, c, d, e, b, c, f)* có độ dài 7, hành trình này này đi qua đỉnh *b* và đỉnh *c* hai lần.
* *(a, b, c, d, e, b, g)* có độ dài 6 đi qua đỉnh *b* hai lần.
* *(a, b, c, d, e, b, a)* là hành trình khép kín đi qua cạnh *(a, b)* hai lần.

Hình 1.5

***Định nghĩa 8***: Trong một đồ thị, hành trình được gọi là một *đường đi* nếu các đỉnh của nó là khác nhau.

Một hành trình khép kín , trong đó là một đường đi được gọi là *chu trình*.

***Ví dụ 6:*** Xét đồ thị Hình 1.5, ta có:

* (a, b, c, f) là đường đi có độ dài 3.
* (b, c, d, e, b) là một chu trình độ dài 4.

***Định lý 4:*** Giả sử *đồ thị G* có *n* đỉnh và bậc của mỗi đỉnh đều lớn hơn hoặc bằng 2. Khi đó trong G bao giờ cũng có một *chu trình*.

a

b

c

d

e

f

g

a

b

c

d

e

f

g

(G)

(G)’

Hình 1.6

***Định nghĩa 9:*** Trong một đồ thị,

* Hai đỉnh liên thông nhau nếu tồn tại một đường đi nào đó nối chúng.
* Đồ thị liên thông nếu mọi cặp đỉnh của đồ thị đều liên thông nhau.

a

b

c

d

e

f

H1

H2

(G)’’

(H)

Hình 1.7 Đồ thị liên thông và không liên thông

* 1. **Ma trận kề, ma trận trọng số:**
     1. **Ma trận kề:**

Xét đồ thị vô hướng G = (V, E) với tập đỉnh và tập cạnh E . Khi đó, ma trận kề của G là ma trận vuông trong đó

Ma trận kề là đối xứng .

Từ ma trận kề ta có thể xác định được các yếu tố cơ bản của đồ thị: số cạnh, danh sách cạnh, bậc của đỉnh.

Ma trận kề của đồ thị có hướng không phải là ma trận đối xứng.

* + 1. **Ma trận trọng số:**

Trong các ứng dụng của lý thuyết đồ thị, mỗi cạnh (hay cung) của đồ thị thường được gán một con số nào đó để chỉ tính chất của cạnh (cung) đó, chẳng hạn như độ dài, thời gian, … Các số này được gọi là trọng số cạnh (cung) của đồ thị. Ký hiệu *c(u, v)* là trọng số của cạnh (cung) (u, v) .

Ma trận trọng số , trong đó nhận giá trị khi hai đỉnh *i* và *j* không kề nhau hoặc nhận giá trị là trọng số của cạnh (cung) (i, j).

Trong cài đặt, tùy từng trường hợp cụ thể của bài toán, các giá trị có thể chọn là một số đủ lớn (đủ nhỏ).

* 1. **Một số đồ thị đặc biệt:**
     1. **Đồ thị đầy đủ:**

***Định nghĩa 10:*** Một đồ thị mà mọi cặp đỉnh của nó đều kề nhau.

K1

K2

K3

Hình 1.8 Đồ thị đầy đủ

Đồ thị đầy đủ là đơn đồ thị có nhiều cạnh nhất trong các đơn đồ thị có cùng số đỉnh. Ký hiệu *Kn* là đồ thị đầy đủ *n* đỉnh. Dễ thấy số cạnh của *Kn* là

* + 1. **Đồ thị vòng:**

Ký hiệu *Cn* là đồ thị vòng n đỉnh.

C1

C2

C3

Hình 1.9 Đồ thị vòng

* + 1. **Đồ thị hai phía, đồ thị hai phía đầy đủ:**

***Định nghĩa 11:*** Một đồ thị *G = (V, E)* là *đồ thị hai phía* nếu các đỉnh của nó được phân thành hai tập con *V1* và *V2* để mỗi cạnh của *G* nối một đỉnh của *V1* với một đỉnh của *V2*.

***Định nghĩa 12:*** Đồ thị hai phía *G = (V, E )* mà tất cả các đỉnh trong tập *V1* được nối với tất cả các đỉnh trong *V2* được gọi là *đồ thị hai phía đầy đủ*.

Ký hiệu là , trong đó , là đồ thị hai phía đầy đủ.

***Ví dụ 7:*** Đồ thị hai phía G1, đồ thị K4 là hai phía đầy đủ.

A

B

C

A

B

C

a

a

b

b

c

c

(G1)

(K4)

Hình 1.10

# Chương 2: BÀI TOÁN DƯỜNG ĐI NGẮN NHẤT

* 1. **Khái niệm:**

Xét đồ thị có hướng với các cung được gán trọng số, nghĩa là, mỗi cung của nó được đặt tương ứng với một số thực gọi là trọng số của nó. Chúng ta sẽ đặt nếu .

Nếu dãy là một đường đi trên G, thì độ dài của nó được định nghĩa là tổng sau:

tức là, độ dài của đường đi chính là tổng các trọng số trên các cung của nó.

Bài toán tìm đường đi ngắn nhất trên đồ thị dưới dạng tổng quát có thể phát biểu như sau:

Tìm đường đi có độ dài nhỏ nhất từ một đỉnh xuất phát s∈V đến đỉnh cuối (đích) t ∈V. Đường đi như vậy ta sẽ gọi là đường đi ngắn nhất từ s đến t còn độ dài của nó ta kí hiệu là d(s, t) và còn gọi là khoảng cách từ s đến t (khoảng cách định nghĩa như vậy có thể là số âm). Nếu như không tồn tại đường đi từ s đến t thì ta sẽ đặt d(s, t) = ∞. Rõ ràng, nếu như mỗi chu trình trong đồ thị đều có độ dài dương, thì đường đi ngắn nhất không có đỉnh nào bị lặp lại (đường đi không có đỉnh nào lặp lại sẽ được gọi là đường đi cơ bản). Mặt khác, nếu trong đồ thị có độ dài âm (chu trình như vậy, để ngắn gọn, ta sẽ gọi là chu trình âm) thì khoảng cách giữa một số cặp đỉnh nào đó của đồ thị có thể là không xác định, bởi vì, bằng cách đi vòng theo chu trình này một số đủ lớn lần, ta có thể chỉ ra đường đi giữa các đỉnh này có độ dài nhỏ hơn bất cứ số thực cho trước nào. Trong những trường hợp như vậy, có thể đặt vấn đề tìm đường đi cơ bản ngắn nhất, tuy nhiên bài toán đặt ra sẽ trở nên phức tạp hơn rất nhiều, bởi vì nó chứa bài toán xét sự tồn tại đường đi Hamilton trong đồ thị như là một trường hợp riêng.

Một tính chất của đường đi ngắn nhất nữa có thể phát biểu một cách không hình thức như sau: Mọi đoạn đường con của đường đi ngắn nhất cũng là đường đi ngắn nhất. Tính chất tuy rất hiển nhiên này nhưng lại hàm chứa một nội dung rất sâu sắc, và người ta thường gọi nó là nguyên lý tối ưu. Việc chứng minh tính đúng đắn của hầu hết các thuật toán tìm đường đi ngắn nhất đều được xây dựng dựa vào nguyên lý này. Trước hết cần chú ý rằng nếu biết khoảng cách từ s đến t, thì dường đi ngắn nhất từ s đến t, trong trường hợp trọng số không âm, có thể tìm được một cách dễ dàng. Để tìm đường đi, chỉ cần để ý là đối với cặp đỉnh s, t ∈V tùy ý (s ≠ t) luôn tìm được đỉnh v sao cho: *d(s,t) = d(s,v) + a(v,t).*

Thực vậy, đỉnh v như vậy chính là đỉnh đi trước đỉnh t trong đường đi ngắn nhất từ s đến t. Tiếp theo ta lại có thể tìm được đỉnh u sao cho d(s,v)=d(s,u)+a(u,v),... Từ giả thiết về tính không âm của các trọng số dễ dàng suy rằng t, v, u, … không chứa đỉnh lặp lại và kết thúc ở đỉnh s. Rõ ràng dãy thu được xác định (nếu lật ngược thứ tự các đỉnh trong nó) đường đi ngắn nhất từ s đến t. Từ đó ta có thuật toán sau đây để tìm đường đi ngắn nhất từ s đến t khi biết độ dài của nó.

Cũng cần lưu ý thêm trong trường hợp trọng số trên các cạnh là không âm, bài toán tìm đường đi ngắn nhất trên đồ thị vô hướng có thể dẫn về bài toán trên đồ thị có hướng, bằng cách thay mỗi cạnh của nó bởi 2 cung có hướng ngược chiều nhau với cùng trọng số là trọng số của cạnh tương ứng. Tuy nhiên, trong trường hợp có trọng số âm, việc thay như vậy có thể dẫn đến chu trình âm.

* 1. **Đường đi ngắn nhất xuất phát từ một đỉnh:**

Phần lớn các thuật toán tìm khoảng cách giữa 2 đỉnh s và t được xây dựng nhờ kĩ thuật tính toán mà ta có thể mô tả đại thể như sau: từ ma trận trọng số, ta tính cận trên d[v] của khoảng cách từ s đến tất cả các đỉnh v∈V. Mỗi khi phát hiện:

(2.1)

Cận trên d[v] sẽ thành: d[v] := d[u] + a[u,v].

Quá trình đó sẽ kết thúc khi nào chúng ta không làm tốt thêm được bất cứ cận trên nào. Khi đó, rõ ràng giá trị của mỗi d[v] sẽ cho ta khoảng cách từ s đến đỉnh v. Khi thể hiện kĩ thuật tính toán ở trên máy tính, cận trên d[v] sẽ được gọi là nhãn của đỉnh v, còn việc tính lại các cận trên này sẽ gọi là phép gán nhãn cho đồ thị và toàn bộ thủ tục thường gọi là thủ tục gán nhãn. Nhận thấy rằng để tính khoảng cách từ s đến t, ở đây, ta phải tính khoảng cách từ s đến tất cả các đỉnh còn lại của đồ thị. Hiện nay vẫn chưa biết thuật toán nào cho phép tìm đường đi ngắn nhất giữa 2 đỉnh làm việc thực sự hiệu quả hơn những thuật toán tìm đường đi ngắn nhất từ một đỉnh đến tất cả các đỉnh còn lại.

Sơ đồ tính toán mà ta vừa mô tả còn chưa là xác định, bởi vì còn phải chỉ ra thứ tự chọn các đỉnh u và v để kiểm tra điều kiện (2.1). Thứ tự chọn này còn ảnh hưởng rất lớn đến hiệu quả của thuật toán.

## Đường đi ngắn nhất trong đồ thị có trọng số cạnh không âm – Thuật toán Dijkstra

* + 1. **Mô tả thuật toán**

***Bài toán.*** Cho đồ thị đơn vô hướng liên thông G có trọng số dương. Tìm đường từ đỉnh *s* đến *v* của G sao cho tổng trọng số của đường đi này là nhỏ nhất.

Dijkstra đề xuất thuật toán tổng quát để tìm đường đi ngắn nhất từ một đỉnh đến tất các các đỉnh còn lại trong đồ thị vô hướng có trọng số dương. Chỉ cần cần lưu ý đến hướng của cung là có thể giải được bài toán đường đi ngắn nhất trong đồ thị có hướng. Đặc điểm của phương pháp này là *cực tiểu hóa nhãn* cho các đỉnh.

Ký hiệu nhãn của đỉnh *v* là d[v] là độ dài ngắn nhất (tổng các trọng số) của đường đi từ *s* đến *v*. Ký hiệu a[u,v] là trọng số của cạnh (u, v). Giả sử tại đỉnh *u* đã được cực tiểu hóa nhãn, khi đó đỉnh v kề với u sẽ được gán nhãn là *d[v]:=min{d[v],d[u]+a[u,v]}*. Trong trường hợp này, đỉnh *u* được gọi là đỉnh đi trước đỉnh *v* trên đường đi ngắn nhất từ s đến *v* qua *u*. Thuật toán kết thúc khi mọi đỉnh được cực tiểu hoá.

s

u

v

a[u,v]

d[u]

d[v]

Hình 2.1

***Ví dụ 8:*** Tìm đường đi ngắn nhất từ đỉnh s đến các đỉnh còn lại của đồ thị trên hình sau:

a

b

c

d

s

t

(6)

(4)

(3)

(2)

(5)

(10)

(2)

(4)

(1)

Hình 2.2

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Lặp | s | a | b | c | d | t | V |
| 0 | [s,0] | [s,4] | [s,] | [s,] | [s,]\* | [s,] | s |
| 1 |  | [d,]\* | [d,] | [d,] | - | [s,] | d |
| 2 |  | - | [d,]\* | [d,] |  | [s,] | a |
| 3 |  |  | - | [b,9]\* |  | [b,11] | b |
| 4 |  |  |  | - |  | [b,11]\* | c |
| 5 |  |  |  |  |  | - | t |

***Ví dụ 9:*** Tìm đường đi ngắn nhất từ đỉnh a đến các đỉnh còn lại của đồ thị có hướng sau đây.

a

b

c

d

e

f

►

►

►

►

►

►

►

►

►

►

(1)

(2)

(2)

(7)

(5)

(1)

(1)

(4)

(3)

(1)

Hình 2.3

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Lặp | a | b | c | d | e | f | V |
|  | [a,0] | [a,1]\* | [a,] | [a,] | [a,] | [a,] | a |
|  |  | - | [b,6] | [b,3]\* | [a,] | [b,8] | b |
|  |  |  | [d,4]\* | - | [d,7] | [b,8] | d |
|  |  |  | - |  | [d,7] | [c,5]\* | c |
|  |  |  |  |  | [f,6] | - | f |
|  |  |  |  |  | - |  | e |

* + 1. **Nội dung thuật toán**

Giải thuật Dijkstra là một giải thuật để giải bài toán đường đi ngắn nhất nguồn đơn trên một đồ thị có trọng số cạnh mà tất cả các trọng số đều không âm. Nó xác định đường đi ngắn nhất giữa hai đỉnh cho trước, từ đỉnh *a* đến đỉnh *b*.

Ở mỗi đỉnh v, giải thuật Dijkstra xác định 3 thông tin: *kv, dv* và *pv*.

*kv*: mang giá trị boolean xác định trạng thái được chọn của đỉnh *v*.

Ban đầu ta khởi tạo tất cả các đỉnh *v* chưa được chọn, nghĩa là:

*dv*: là chiều dài đường đi mà ta tìm thấy cho đến thời điểm đang xét từ *a* đến *v*.

Khởi tạo:

*pv*: là đỉnh trước của đỉnh v trên đường đi ngắn nhất từ a đến b. Đường đi ngắn nhất từ a đến b có dạng . Khởi tạo .

Sau đây là các bước của giải thuật Dijkstra:

***Bước 1:*** Khởi tạo: Đặt .

***Bước 2:*** Chọn sao cho và . Nếu thì kết thúc, không tồn tại đường đi từ a đến b.

***Bước 3:*** Đánh dấu đỉnh v, .

***Bước 4:*** Nếu thì kết thúc và là độ dài ngắn nhất từ *a* đến *b*. Ngược lại nếu sang ***Bước 5***.

***Bước 5:*** Với mỗi đỉnh u kề với v mà , kiểm tra. Nếu thì . Ghi nhớ đỉnh *v*: . Quay lại ***Bước 2.***

* 1. **Độ phức tạp thuật toán:**

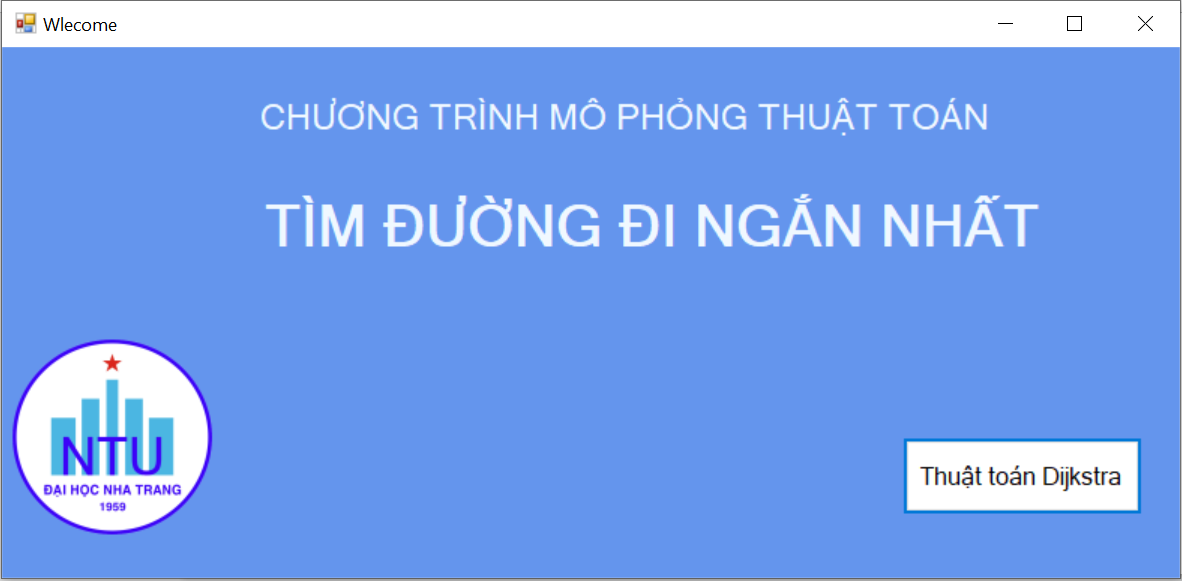
***Mệnh đề:*** Thuật toán Dijkstra tìm đường đi ngắn nhất từ một đỉnh cho trước đến một đỉnh tuỳ ý trong đơn đồ thị vô hướng liên thông có trọng số có độ phức tạp là O(n2).

***Chứng minh***: Thuật toán dùng không quá n−1 bước lặp. Trong mỗi bước lặp, dùng không hơn 2(n−1) phép cộng và phép so sánh để sửa đổi nhãn của các đỉnh. Ngoài ra, một đỉnh thuộc Sk có nhãn nhỏ nhất nhờ không quá n−1 phép so sánh. Do đó thuật toán có độ phức tạp O(n2 ).

# Chương 3: XÂY DỰNG CHƯƠNG TRÌNH

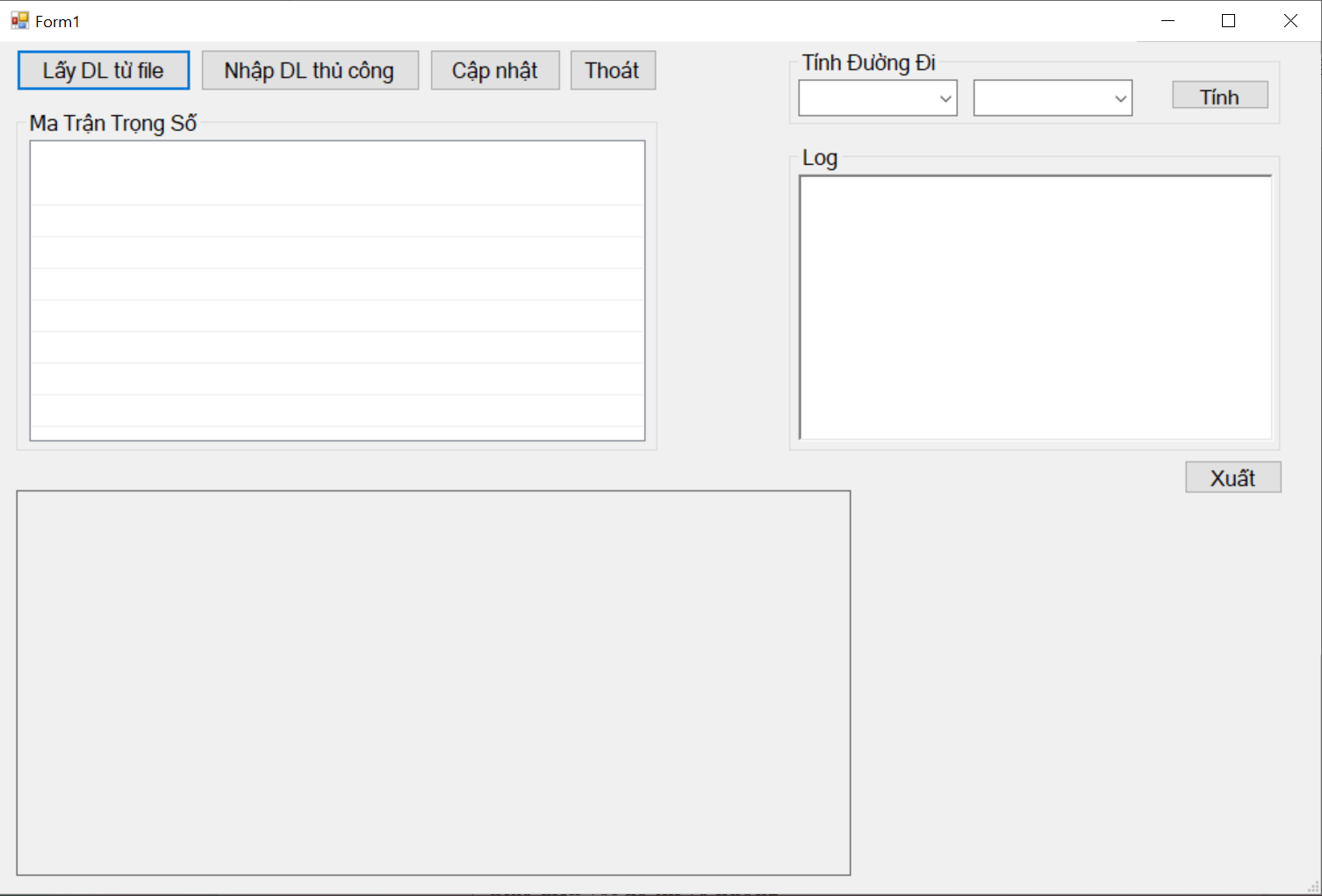
3. 1. **Cấu tạo:**

* Chương trình được cấu tạo bởi 3 class chính:
  + Class DuongDi lưu trữ các con đường đang tìm kiếm.
  + Class Dijkstra thực hiện chức năng tìm đường đi ngắn nhất giữa hai điểm và lưu trữ ma trận trọng số được load vào chương trình.
  + Class Form1 tạo giao diện chương trình, xử lý các yêu cầu mà người dùng đưa vào. In kết quả ra cho người dùng xem.
    1. **Class DuongDi**
* List<int> diemDaDi: Lưu trữ các điểm đã đi qua.
* bool isDie: Xem con đường này đã đi vào đường cụt hay chưa.
* DuongDi(): Hàm khởi tạo của class.
* bool ThemDiem(int diem): Thêm một điểm vào cuối con đường.
  + 1. **Class Dijkstra**
* int[,] duLieu: Lưu ma trận trọng số được đưa vào.
* List<bool> daXet: Lưu danh sách các điểm đã xét.
* List<DuongDi> listDuongDi: Lưu danh sách các con đường đang tìm.
* int diemDau: Lưu điểm xuất phát.
* int diemCuoi: Lưu điểm cần tới.
* Dijkstra(int[,] duLieu): Hàm khởi tạo với giá trị truyền vào là một ma trận trọng số.
* Dijkstra(string urlFile): Hàm khởi tạo với giá trị truyền vào là một file chứa dữ liệu là ma trận trọng số.
* Init(): Hàm khởi tạo giá trị ban đầu cho các thuộc tính.
* List<int> TimDuong(int diemDau,int diemCuoi): Tim đường đi giữa đi hai điểm truyền vào.
* bool KetThuc(int diemCuoi): Kiểm tra xem có phải tất cả các con đường đã đi tới đường cụt hay không (isDie=true). Nếu tất cả đã tới đường cụt thì không tồn tại đường đi giữa hai điểm truyền vào hoặc đã tìm được đường đi giữa hai điểm.
* int TimDuong(DuongDi dd): Tìm điểm đi kế tiếp từ đỉnh cuối cùng của đường đi.
* List<int> TimDiemKe(int diem): Tìm danh sách điểm kề từ một điểm.
* void DanhDauVoCung(): Xử lý ma trận trọng số truyền vào chương trình. Nếu giữa hai điểm không tồn tại đường đi thì đánh dấu trong ma trận trọng số là -1.
* string XoaKiTuLa(string duLieu): Xử lý dữ liệu vào từ file. Loại bỏ toàn bộ các ký tự lạ (không phải là số, không phải là dấu cách, không phải là dấu xuống hàng).
* bool KiemTraKyTu(char kyTu): Kiểm tra xem ký tự có phải là số hay không.
* public int TinhDuong(List<int> duongDi): Tính khoảng cách giữa hai điểm nhờ con đường được truyền vào.
  + 1. **Class Form1**
* fDijkstra(): Khởi tạo các đối tượng trong form.
* string file : Đường dẫn file đang mở hiện tại.
* TimDuongDijkstra td: Khai báo biến tìm đường.
* void LoadDuLieuLenListView(): Load ma trận trọng số lên listview.
* string ChuyenDoi(int so): Nếu số đưa vào là -1 thì chuyển thành hiển thị là "∞".
* void LoadDuLieuLenComboBox(): Load danh sách đỉnh vào combobox.
* void btMoFile\_Click(object sender, EventArgs e): Bắt sự kiện người dùng ấn vào nút mở file, và cho người dùng chọn file dữ liệu muốn mở.
* void btTinhDuongDi\_Click(object sender, EventArgs e): Bắt sự kiên người dùng ấn vào nút Tính đường đi và tính toán đường đi dựa vào hai điểm người dùng muốn tính.
* void xóaDữLiệuToolStripMenuItem\_Click(object sender, EventArgs e): Xóa dữ liệu load lên từ file và kết quả tính toán khi người dùng chọn Xóa Dữ Liệu trong Menu.
* void thôngTinToolStripMenuItem\_Click(object sender, EventArgs e): Show thông tin tác giả.
* void thoátToolStripMenuItem\_Click(object sender, EventArgs e): Thoát khỏi chương trình.
  1. **Thiết kế giao diện**
* Giao diện welcome



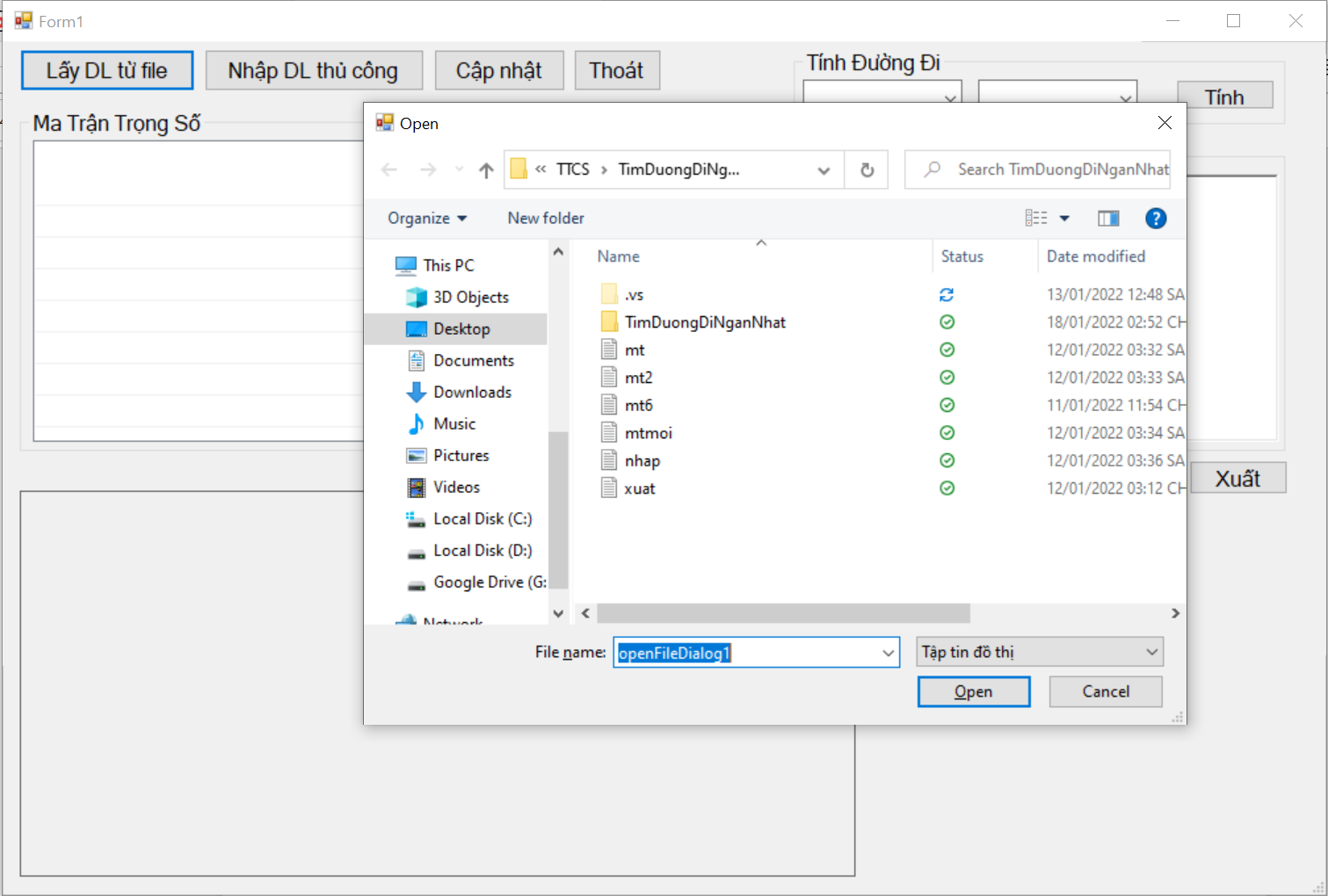
Hình 3.1 Form giao diện welcome

* Giao diện chính



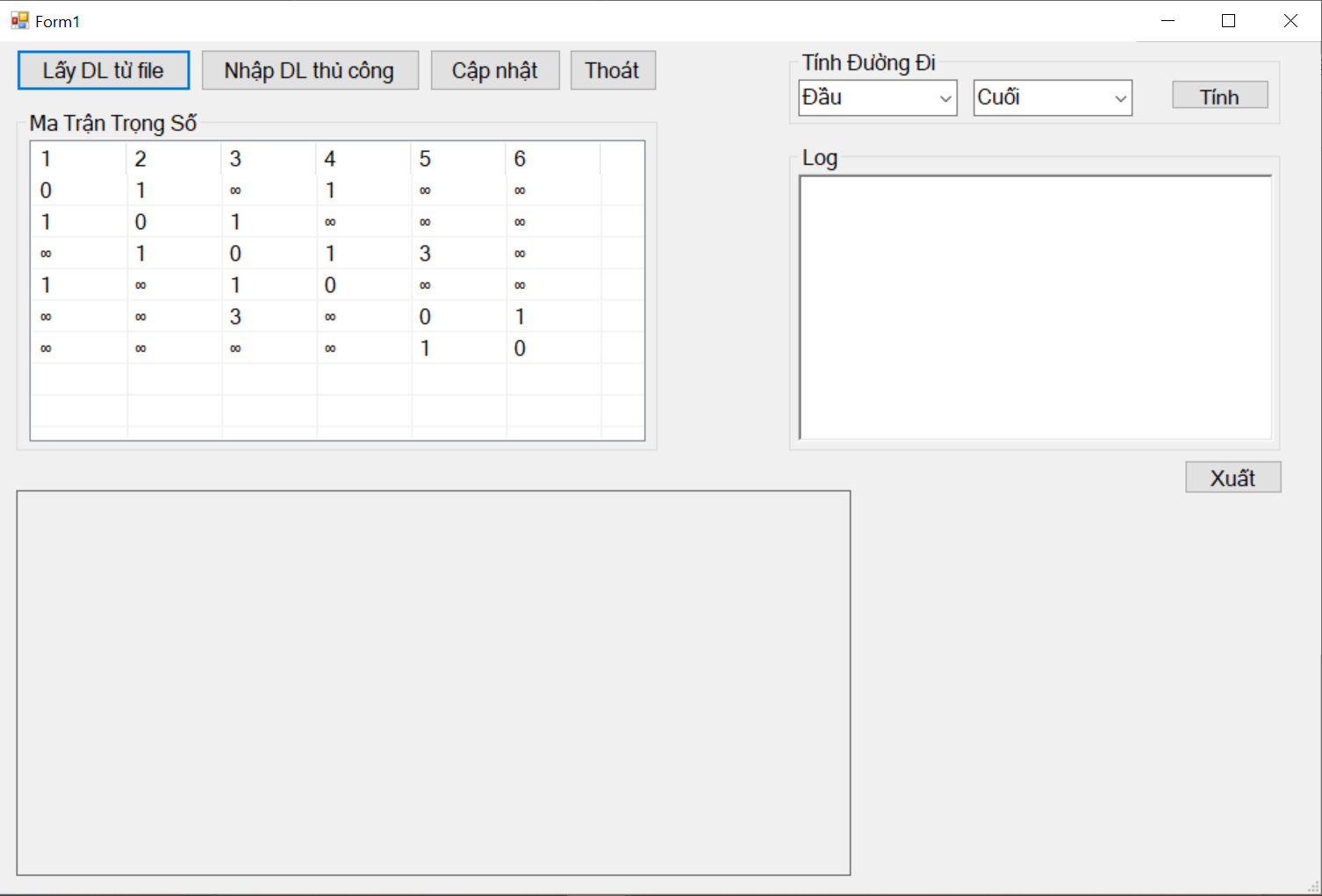
Hình 3.2 Form giao diện chính

* Giao diện khi nhấn nút Lấy DL từ file



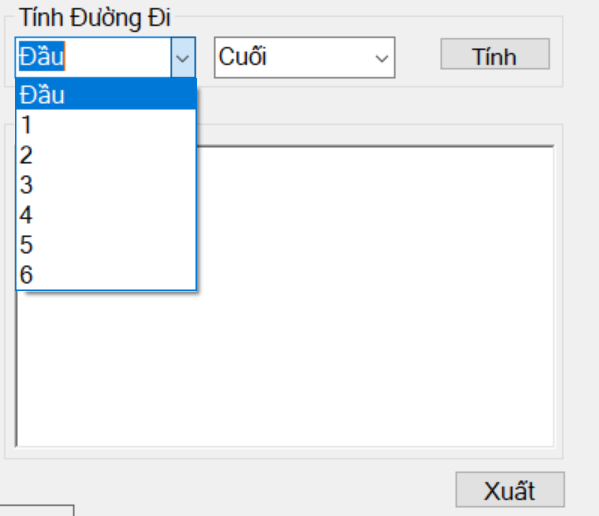
Hình 3.3 Hiện ra form để lấy dữ liệu từ file

* Giao diện sau khi mở từ file thư mục

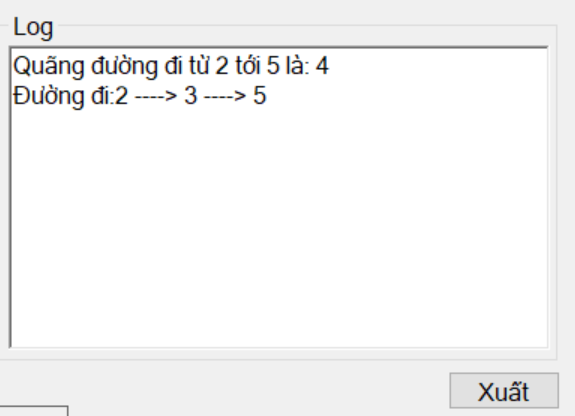


Hình 3.4 Form giao diện sau khi mở từ file thư mục

* Giao diện tính đường đi

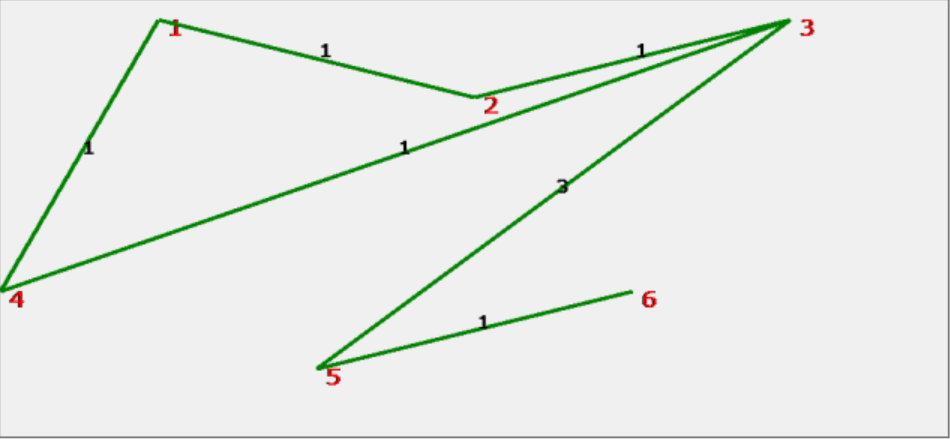


Hình 3.5 Form chọn điểm đầu, chọn điểm cuối, tính, in và xuất kết quả

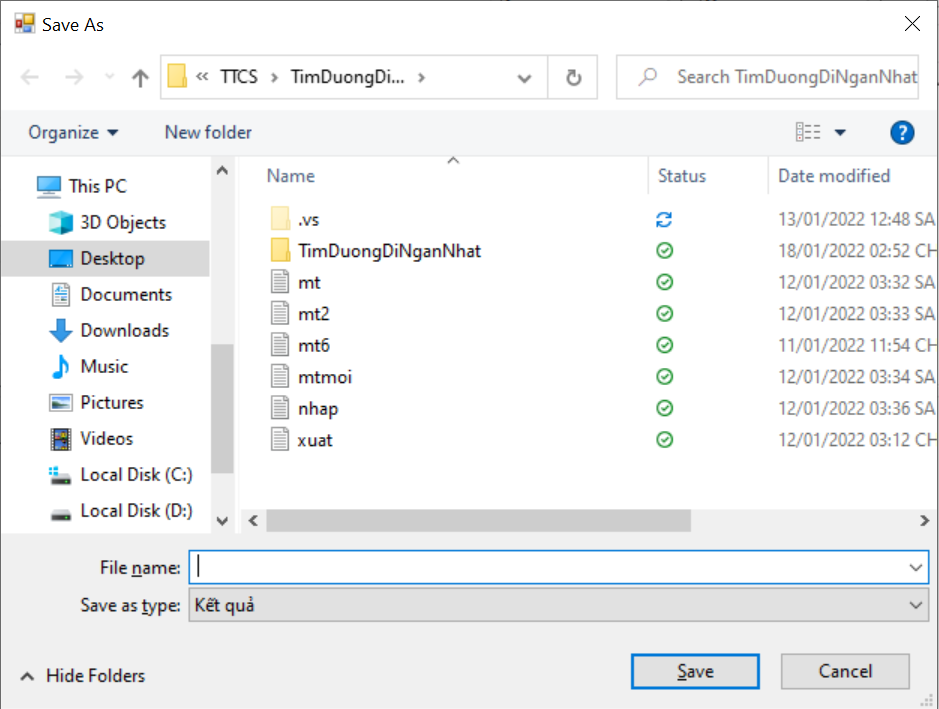


Hình 3.6 Form kết quả hiển thị sau khi tính

* Giao diện thể hiện minh họa đồ thị

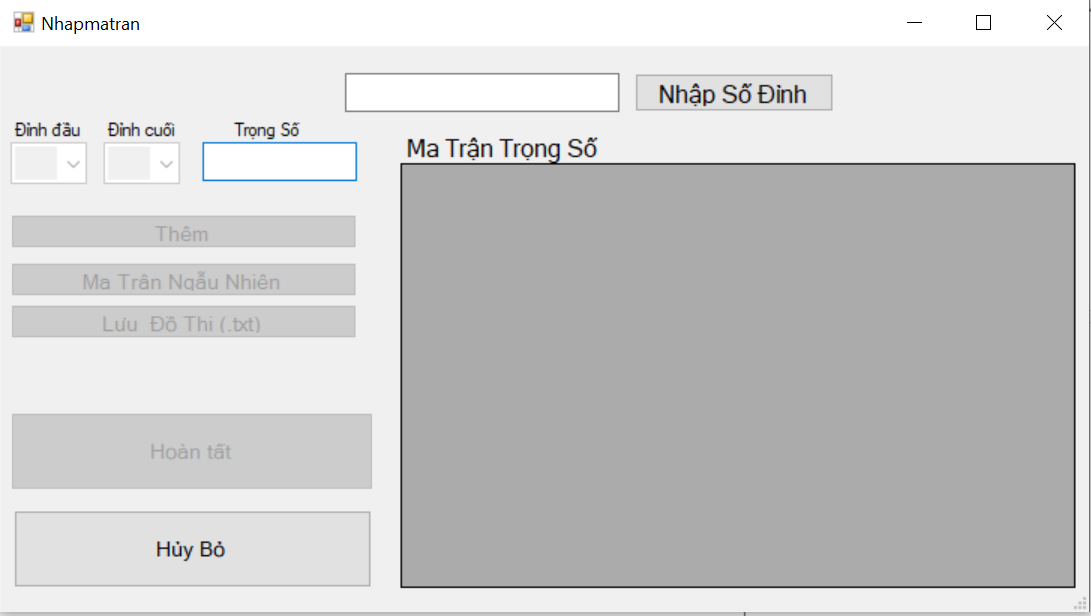


Hình 3.7 Form thể hiện minh họa đồ thị

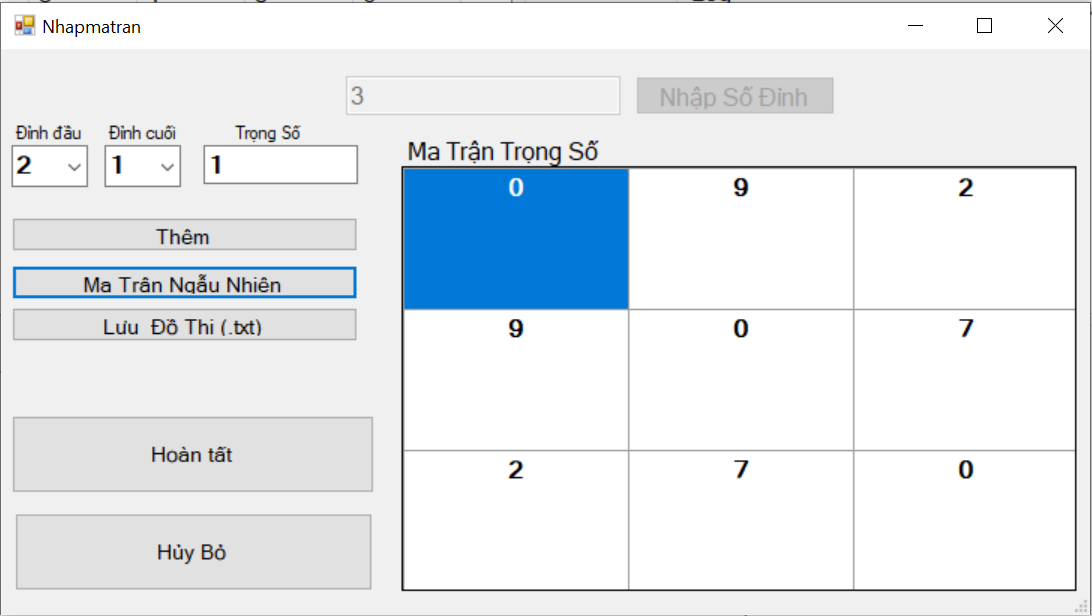


Hình 3.8 Form lưu kết quả

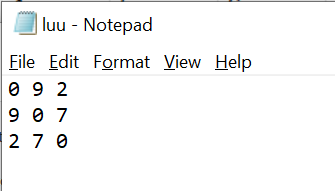
* Giao diện nhập ma trận thủ công



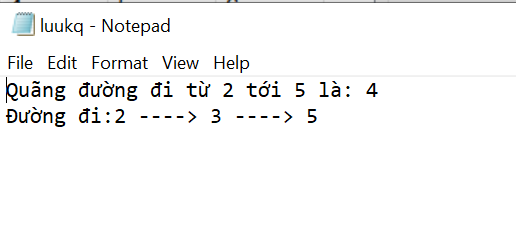
Hình 3.9 Form nhập thủ công



Hình 3.10 Form sau khi nhập ma trận hoặc chọn ma trận ngẫu nhiên



Hình 3.11 Ma trận sau khi lưu



Hình 3.12 Kết quả sau khi xuất

# KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

1. **Kết luận:**

Đề tài thực tập cơ sở *“Cài đặt thuật toán tìm đường đi ngắn nhất Dijkstra và xây dựng chương trình minh họa”* đã trình bày một số khái niệm cơ bản của lý thuyết đồ thị và trình bày thuật toán Dijktra để giải quyết bài toán tìm đường đi ngắn nhất từ một đỉnh đến tất cả các đỉnh còn lại trong đồ thị theo một cách tuần tự. Chương trình minh họa sử dụng ngôn ngữ lập trình C# đã minh họa trực quan được thuật toán Dijkstra và cho phép chúng ta thao tác trực tiếp trên giao diện các công việc sau:

* Thêm mới đồ thị (từ bàn phím hoặc từ file).
* Tìm đường và xuất kết quả dạng văn bản và dạng file văn bản.
* Thể hiện đồ họa đồ thị và đường đi ngắn nhất (đã thể hiện được đồ họa đồ thị nhưng chưa thể hiện được đường đi ngắn nhất).

Qua thời gian thực hiện đề tài, em đã thu được nhiều kiến thức bổ ích. Tuy nhiên do thời gian có hạn và kiến thức vẫn còn hạn chế nên kết quả đề tài của em vẫn không tránh khỏi thiếu sót và có nhiều điểm chưa phù hợp.

Em xin chân thành cảm ơn cô ThS.GV. Nguyễn Thị Hương Lý đã tận tình chỉ bảo và hướng dẫn những điểm còn chưa phù hợp, giúp em hoàn thiện hơn trong phần trình bày báo cáo và demo chương trình.

1. **Hướng phát triển:**

Đề tài thực tập cơ sở *“Cài đặt thuật toán tìm đường đi ngắn nhất Dijkstra và xây dựng chương trình minh họa”* trình bày bài toán tìm đường đi ngắn nhất bằng thuật toán Dijkstra theo một cách tuần tự, giải thuật có độ phức tạp O(n2) khi n tăng lên quá lớn, thì thời gian xử lý sẽ chậm đi đáng kể điều này sẽ không đáp ứng được với đòi hỏi thời gian nhanh hơn. Vì vậy, trong thời gian tới em sẽ tìm cách xử lý thuật toán Dijkstra nhằm giúp tiết kiệm được thời gian và đáp ứng được với các bài toán lớn. Ngoài ra, cần xây dựng lại giao diện chương trình để có giao diện đẹp hơn và thân thiện với người dùng.

# TÀI LIỆU THAM KHẢO

1. Đỗ Như An (2012), *Toán Rời Rạc*, Giáo trình bài giảng trường Đại học Nha Trang.
2. Phạm Thị Kim Ngoan (2015), Lập trình hướng đối tượng, Đại học Nha Trang, Nha Trang.
3. Tham khảo trên <http://vi.wikipedia.org>.