BỘ GIÁO DỤC VÀ ĐÀO TẠO

**TRƯỜNG ĐẠI HỌC NHA TRANG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**THỰC TẬP CƠ SỞ**

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

**Giảng viên hướng dẫn: Nguyễn Thị Hương Lý**

**Sinh viên thực hiện: Nguyễn Ngọc Hồng Hân**

**Mã số sinh viên: 61132924**

Khánh Hòa – 2022

# MỤC LỤC

Contents

[MỤC LỤC 2](#_Toc92887027)

[MÔ TẢ YÊU CẦU 3](#_Toc92887028)

[CÁC BÀI TOÁN VỀ ĐƯỜNG ĐI NGẮN NHẤT TRÊN ĐỒ THỊ 4](#_Toc92887029)

[**1.** **Đường đi ngắn nhất trong đồ thị có trọng số cạnh không âm – Thuật toán Dijkstra** 4](#_Toc92887030)

[**2.** **Đường đi ngắn nhất giữa các cặp đỉnh – Thuật toán Floyd** 6](#_Toc92887031)

[CHƯƠNG TRÌNH 8](#_Toc92887032)

[**1.** **Class DuongDi** 8](#_Toc92887033)

[**2.** **Class Dijkstra** 8](#_Toc92887034)

[**3.** **Class Form1** 10](#_Toc92887035)

# MÔ TẢ YÊU CẦU

1. Sinh viên bắt buộc phải tạo tài khoản trên Github để tổ chức lưu trữ và quản

lý phiên bản mã nguồn, push kết quả định kỳ ½ ngày nếu có thay đổi mã

nguồn.

1. 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

1. Mức 2: Thể hiện đồ họa đồ thị và đường đi ngắn nhất.

# CÁC BÀI TOÁN VỀ ĐƯỜNG ĐI NGẮN NHẤT TRÊN ĐỒ THỊ

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

**Bài toán.** Cho đồ thị đơn vô hướng liên thông G có trọng số dương. Tìm đường từ đỉnh *s* đến *t* 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.

d[v]

d[u]

a[u, v]

u

v

s

**Ví dụ 1.1:** Tìm đường đi ngắn nhất từ đỉnh s đến các đỉnh còn lại của đồ thị trên hình sau:

(6)

b

a

(4)

(4)

(3)

(2)

(10)

(2)

(1)

(5)

d

c

t

s

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 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 1.2:** Tìm đường đi ngắn nhất từ đỉnh a đến các đỉnh còn lại của đồ thị có hướng sau đây.

(7)

c

b f

(5**)** (1)

(1)(2)(1)(1)

(4)

a (2) d (3) e

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 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 |

**Ví dụ 1.3:** Bài toán lập lịch thi công công trình. Việc thi công một công trình lớn được chia thành n công đoạn đánh số từ 1 đến n. Có một số công đoạn mà việc thực hiện nó chỉ được tiến hành sau khi một số công đoạn nào đó đã hoàn thành. Đối với mỗi công đoạn i biết t[i] là thời gian cần thiết để hoàn thành nó (i = 1,n ). Hãy xác định lịch trình thi công công trình (chỉ rõ mỗi công đoạn phải được bắt đầu thực hiện vào thời điểm nào) sao cho công trình hoàn thành xong sớm nhất có thể được.

|  |  |  |
| --- | --- | --- |
| Công đoạn | Thời gian t[i] ngày | Các công đoạn phải hoàn thành trước |
| 1 | 15 |  |
| 2 | 30 | 1 |
| 3 | 80 |  |
| 4 | 45 | 2,3 |
| 5 | 124 | 4 |
| 6 | 15 | 2,3 |
| 7 | 15 | 5,6 |
| 8 | 19 | 5 |

**Thuật toán được mô tả như sau:**

* 𝐼𝑛𝑝𝑢𝑡: Đồ 𝑡ℎị 𝐺 = (𝑉, 𝐸) 𝑐ó 𝑡𝑟ọ𝑛𝑔 𝑠ố, 𝑠 𝑙à đỉ𝑛ℎ 𝑥𝑢ấ𝑡 𝑝ℎá𝑡
* 𝑂𝑢𝑡𝑝𝑢𝑡:𝐾ℎ𝑜ả𝑛𝑔 𝑐á𝑐ℎ 𝑛𝑔ắ𝑛 𝑛ℎấ𝑡 𝑡ừ 𝑠 đế𝑛 𝑐á𝑐 đỉ𝑛ℎ 𝑐ò𝑛 𝑙ạ𝑖.

procedure Dijkstra;

for v ∈ V do

begin

d[v] = a[s, v];

truoc[v]=s;

end

d[s] = 0; H= V\{s};

while H ≠ ∅ do

begin

d[u]=min{d[z]:z ∈ 𝐻};

H=H\{u};

for v ∈ 𝑉 𝑑𝑜

if d[v] > d[u] + a[u,v] then

begin

d[v]=d[u] + a[u,v];

truoc[v]=u;

end

end

end

1. **Đường đi ngắn nhất giữa các cặp đỉnh – Thuật toán Floyd**

Xét đồ thị *G = (V,E), |V| = n.* Gọi *A = (aij) (i, j = )* là ma trận trọng số của G. Thuật toán Floyd cho kết quả là ma trận, ma trận đối *xứng D = (Dij)* *(i, j = ),* trong đó *dij* cho độ dài đường đi ngắn nhất từ *i* đến *j* và ma trận P = (p*ij*) *(i, j = )* với pij ghi nhận đỉnh đi trước đỉnh *j* trong đường đi ngắn nhất từ *i* đến *j* . Thuật toán được mô tả như sau.

procedure Floyd;

begin

for i=1 to n do

for j=1 to n do

begin

d[i,j]=a[i,j];

p[i, j]=i;

end

for k=1 to n do

for i=1 to n do

for j= 1 to n do

if d[i, j] > d[i ,k] + d[k, j] then

begin

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

p[i, j] = p[k, j];

end

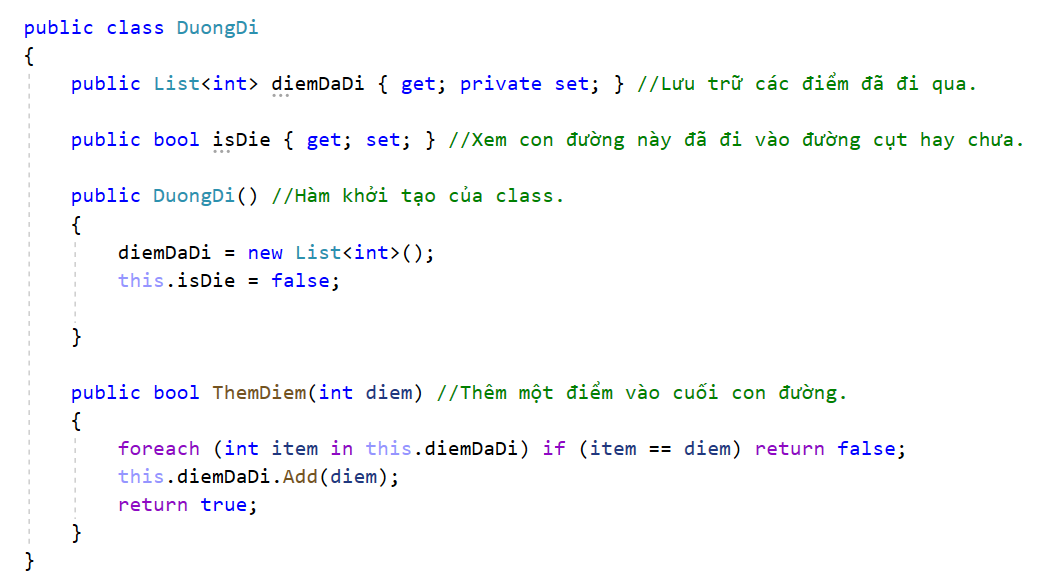
end

# CHƯƠNG TRÌNH

**Cấu tạo:**

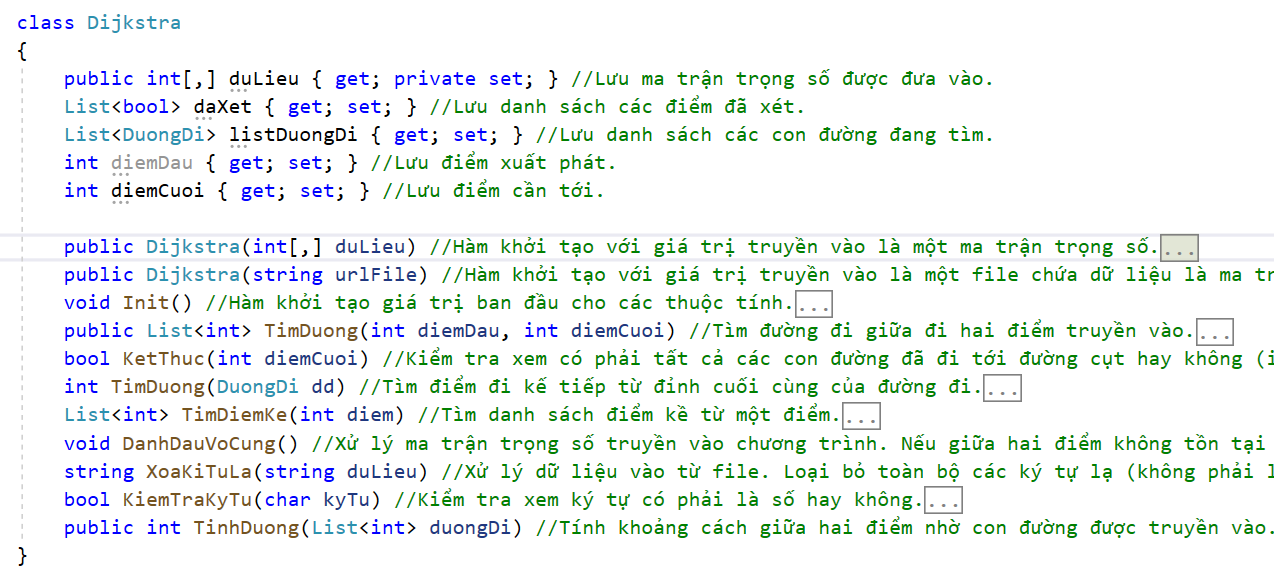
* Chương trình được cấu tạo bởi 3 class chính:
  + Class DuongDi lưu trữ các con đường đang tìm kiếm.
  + Class Dijkstra thực hiện chức năng tìm đường đi ngắn nhất giữa hai điểm và lưu trữ ma trận trọng số được load vào chương trình.
  + Class Form1 tạo giao diện chương trình, xử lý các yêu cầu mà người dùng đưa vào. In kết quả ra cho người dùng xem.
  + Class fThongTin tạo giao diện in thông tin về tác giả.

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**