**TRƯỜNG ĐẠI HỌC BÁCH KHOA**

**KHOA CÔNG NGHỆ THÔNG TIN**



**ĐỒ ÁN LẬP TRÌNH TÍNH TOÁN**

**BÀI TOÁN ĐƯỜNG ĐI NGẮN NHẤT**

Người hướng dẫn**: TS. NGUYỄN VĂN HIỆU**

Sinh viên thực hiện**:**

**Trương Công Hoàng Long LỚP: 21TCLC\_Nhat1 NHÓM: 11**

**Trần Lê Minh LỚP: 21TCLC\_Nhat1 NHÓM: 11**

**Đà Nẵng, 07/2022**

MỤC LỤC

[MỤC LỤC 1](#_Toc103108891)

[DANH MỤC HÌNH VẼ 2](#_Toc103108892)

[MỞ ĐẦU{font: TimeNew Roman, bold, size: 14, căn lề: center} 1](#_Toc103108893)

[1. TỔNG QUAN ĐỀ TÀI 1](#_Toc103108894)

[2. CƠ SỞ LÝ THUYẾT 1](#_Toc103108895)

[2.1. Ý tưởng 1](#_Toc103108896)

[2.2. Cơ sở lý thuyết 1](#_Toc103108897)

[3. TỔ CHỨC CẤU TRÚC DỮ LIỆU VÀ THUẬT TOÁN 2](#_Toc103108898)

[3.1. Phát biểu bài toán 2](#_Toc103108899)

[3.2. Cấu trúc dữ liệu 2](#_Toc103108900)

[3.3. Thuật toán 2](#_Toc103108901)

[4. CHƯƠNG TRÌNH VÀ KẾT QUẢ 4](#_Toc103108902)

[4.1. Tổ chức chương trình 4](#_Toc103108903)

[4.2. Ngôn ngữ cài đặt 4](#_Toc103108904)

[4.3. Kết quả 4](#_Toc103108905)

[4.3.1. Giao điện chính của chương trình 4](#_Toc103108906)

[4.3.2. Kết quả thực thi của chương trình 4](#_Toc103108907)

[4.3.3. Nhận xét đánh giá 4](#_Toc103108908)

[5. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 4](#_Toc103108909)

[5.1. Kết luận 4](#_Toc103108910)

[5.2. Hướng phát triển 4](#_Toc103108911)

[TÀI LIỆU THAM KHẢO 5](#_Toc103108912)

[PHỤ LỤC 6](#_Toc103108913)

DANH MỤC HÌNH VẼ

MỞ ĐẦU{font: TimeNew Roman, bold, size: 14, căn lề: center}

{Để 2 dòng trống}

*{Font: Time New Roman; thường; cỡ chữ: 13; dãn dòng: 1,3; căn lề: justified}*

{Trong phần này, cần trình bày về: Mục đích thực hiện đề tài, mục tiêu đề tài, phạm vi và đối tượng nghiên cứu, phương pháp nghiên cứu, cấu trúc của đồ môn học}

Mục đích thực hiện đề tài: Trình bày một số kiến thức cơ bản của lí thuyết đồ thị liên quan đến bài toán đường đi ngắn nhất, giới thiệu bài toán và áp dụng một số thuật toán để giải quyết vấn đề.

Mục tiêu đề tài:

Phạm vi và đối tượng nghiên cứu: Tập trung trình bày hai thuật toán Dijkstra và Ford-Bellman, đồng thời tìm hiểu và trình bày một số ứng dụng của bài toán tìm đường đi ngắn nhất trong thực tế.

Phương pháp nghiên cứu: Tìm hiểu sách tham khảo, video, bài giảng, các mẫu báo cáo trên mạng.

Cấu trúc của đồ môn học:

# TỔNG QUAN ĐỀ TÀI

{Nội dung A {Font: Time New Roman; thường; cỡ chữ: 13; dãn dòng: 1,3; căn lề: justified}

Tìm đường đi ngắn nhất giữa các điểm trên bản đồ nhằm tối ưu chi phí nhất có thể đã trở nên quen thuộc trong thời đại ngày nay, khi mà việc giao thông đi lại toàn cầu phát triển mạnh mẽ. Có nhiều hướng kết nối hai vị trí lại với nhau nhưng dựa vào các yếu tố chẳng hạn quãng đường, địa hình, phương tiện, giao thông … mà ta quyết định đường đi ngắn nhất, tối ưu chi phí nhất. Chẳng hạn tìm đi ngắn nhất giữa hai thành phố, tìm tuyến xe buýt…

Mở rộng ra có thể kể đến ứng dụng trong việc truyền các gói tin trên mạng internet, tìm kiếm và kết nối bạn bè trên mạng xã hội …

# CƠ SỞ LÝ THUYẾT

## Ý tưởng

## Cơ sở lý thuyết

Đồ thị G=(V,E): V là các đỉnh, E là các cung có trọng số

Đồ thị trọng số (weighted graph) là đồ thị có gắn một số (số nguyên hay số thực) cho mỗi cạnh hoặc mỗi cung.

Số nguyên hay số thực cho mỗi cạnh đại điện cho: cự ly, thời gian, chi phí, tốc độ…

Danh sách kề: Đồ thị G = (V,E) V={v1,...,vn} có thể được biểu điễn bằng n danh sách liên kết, trong đó mỗi danh sách liên kết thứ i sẽ biểu điễn các đỉnh kề với đỉnh vi. Mỗi phần tử trong danh sách liên kết thứ i chứa dữ liệu là đỉnh có liên kết với đỉnh vi và trọng số của liên kết đó.

Diagram

Description automatically generated

# TỔ CHỨC CẤU TRÚC DỮ LIỆU VÀ THUẬT TOÁN

## Phát biểu bài toán

Mô tả đầu vào (Input) và đầu ra (Output)

Input: Tệp chứa ma trận trọng số của đồ thị, dữ liệu có thể tạo ngẫu nhiên hoặc nhập từ bàn phím. Dòng đầu của tệp chứa số đỉnh đồ thị, các dòng còn lại là của ma trận trọng số.

Output: Xuất ra màn hình kết quả của 2 truy vấn: tìm đường đi ngắn nhất giữa hai đỉnh trên đồ thị và tìm các đỉnh có đường đi ngắn nhất đến một đỉnh có định không lớn hơn một giá trị cụ thể.

## Cấu trúc dữ liệu

Sử dụng danh sách kề để lưu đồ thị. Đối với thuật toán Dijkstra sử dụng cấu trúc dữ liệu HEAP trong quá trình tính toán.

## Thuật toán

Trình bày các thuật toán và phân tích độ phức tạp của các thuật toán.

**a.** Thuật toán Ford-Bellman có thể phát biểu rất đơn giản: Với đỉnh xuất phát S. Gọi d[v] là khoảng cách từ S tới v. Ban đầu d[v] được khởi gán bằng c[S, v]. Sau đó ta tối ưu hoá dần các d[v] như sau: Xét mọi cặp đỉnh u, v của dồ thị, nếu có một cặp đỉnh u, v mà d[v] > d[u]+ c[u, v] thì ta đặt lại d[v] := d[u] + c[u, v]. Tức là nếu độ dài đường đi từ S tới v lại lớn hơn tổng độ dài đuờng đi từ S tới u cộng với chi phí đi từ u tới v thì ta sẽ huỷ bỏ đuờng đi từ S tới v đang có và coi đuờng đi từ S tới v chính là đường đi từ S tới u sau đó đi tiếp từ u tới v. Chú ý rằng ta đặt c[u, v] = +∞ nếu (u, v) không là cung. Thuật toán sẽ kết thúc khi không thể tối ưu thêm bất kỳ một nhãn d[v] nào nữa.

Tính đúng của thuật toán:

• Tại buớc lặp 1: Bước khởi tạo d[v] = c[S, v]: thì dãy d[v] chính là độ dài ngắn nhất của đường đi từ S tới v qua không quá 1 cạnh.

• Giả sử tại buớc lặp thứ i (i ≥ 1), d[v] bằng độ dài đuờng đi ngắn nhất từ S tới v qua không quá i cạnh, thì do tính chất: đuờng đi từ S tới v qua không quá i + 1 cạnh sẽ phải thành lập bằng cách: lấy một đuờng đi từ S tới một đỉnh u nào đó qua không quá i cạnh, rồi đi tiếp tới v bằng cung (u,v). Nên độ dài đuờng đi ngắn nhất từ S tới v qua không quá i + 1 cạnh sẽ được tính bằng giá trị nhỏ nhất trong các giá trị: (Nguyên lý tối ưu Bellman)

♦ Ðộ dài đường đi ngắn nhất từ S tới v qua không quá i cạnh

♦ Ðộ dài đường đi ngắn nhất từ S tới u qua không quá i cạnh cộng với trọng số cạnh (u, v)(∀u)

Vì vậy, sau buớc lặp tối ưu các d[v] bằng công thức

d[v]buớc i+1 = min(d[v]buớc i, d[u]buớc i + c[u, v]) (∀u)

thì các d[v] sẽ bằng độ dài đuờng đi ngắn nhất từ S tới v qua không quá i + 1 cạnh.

Sau buớc lặp tối ưu thứ n - 2, ta có d[v] = độ dài đường đi ngắn nhất từ S tới v qua không quá n – 1 cạnh. Vì đồ thị không có chu trình âm nên sẽ có một đuờng đi ngắn nhất từ S tới v là đuờng đi cơ bản (qua không quá n - 1 cạnh). Tức là d[v] sẽ là độ dài đuờng đi ngắn nhất từ S tới v.

Vậy thì số buớc lặp tối ưu hoá sẽ không quá n - 2 buớc.

Trong khi cài đặt chương trình, nếu mỗi buớc ta mô tả dưới dạng:

for u := 1 to n do

for v := 1 to n do

d[v] := min(d[v], d[u] + c[u, v]);

Thì do sự tối ưu bắc cầu (dùng d[u] tối ưu d[v] rồi lại có thể dùng d[v] tối ưu d[w] nữa...) nên chỉ làm tốc độ tối ưu nhãn d[v] tăng nhanh lên chứ không thể giảm đi được.

Độ phức tạp tính toán: t dùng 3 vòng lặp để duyệt các đỉnh nên độ phức tạp O(n3)

**b.** Thuật toán Dijkstra hoạt động hiệu quả hơn nhiều so với thuật toán Ford-Bellman trong trường hợp trọng số trên các cung không âm. Với đỉnh v ∈ V, Gọi d[v] là độ dài đường đi ngắn nhất từ S tới v. Thuật toán Ford-Bellman khởi tạo d[v] = c[S, v]. Sau đó tối ưu hoá dần các nhãn d[v] bằng cách sửa nhãn theo công thức: d[v] :=min(d[v], d[u] + c[u, v]) với ∀u, v ∈ V. Như vậy nếu như ta dùng đỉnh u sửa nhãn đỉnh v, sau đó nếu ta lại tối ưu được d[u] thêm nữa thì ta cung phải sửa lại nhãn d[v] dẫn tới việc d[v] có thể phải chỉnh đi chỉnh lại rất nhiều lần. Vậy nên chăng, tại mỗi buớc không phải ta xét mọi cặp đỉnh (u,v) dể dùng đỉnh u sửa nhãn đỉnh v mà sẽ chọn đỉnh u là đỉnh mà không thể tối ưu nhãn d[u] thêm được nữa.

-Buớc 1: Khởi tạo

Với đỉnh v ∈ V, gọi nhãn d[v] là độ dài đường đi ngắn nhất từ S tới v. Ta sẽ tính các d[v]. Ban đầu d[v] được khởi gán bằng c[S, v]. Nhãn của mỗi đỉnh có hai trạng thái tự do hay cố định, nhãn tự do có nghĩa là có thể còn tối ưu hơn được nữa và nhãn cố định tức là d[v] đã bằng độ dài đường đi ngắn nhất từ S tới v nên không thể tối ưu thêm. Ðể làm điều này ta có thể sử dụng kỹ thuật đánh dấu: Free[v] = TRUE hay FALSE tuỳ theo d[v] tự do hay cố định. Ban đầu các nhãn đều tự do.

-Buớc 2: Lặp

Buớc lặp gồm có hai thao tác:

1. Cố định nhãn: Chọn trong các đỉnh có nhãn tự do, lấy ra đỉnh u là đỉnh có d[u] nhỏ

nhất, và cố định nhãn đỉnh u.

2. Sửa nhãn: Dùng đỉnh u, xét tất cả những đỉnh v và sửa lại các d[v] theo công thức:

d[v] := min(d[v], d[u] + c[u, v])

Buớc lặp sẽ kết thúc khi mà đỉnh dích F được cố định nhãn (tìm được đường đi ngắn nhất từ S tới F); hoặc tại thao tác cố định nhãn, tất cả các đỉnh tự do dều có nhãn là +∞ (không tồn tại đường đi). Có thể đặt câu hỏi, ở thao tác 1, tại sao đỉnh u như vậy được cố định nhãn, giả sử d[u] còn có thể tối ưu thêm được nữa thì tất phải có một đỉnh t mang nhãn tự do sao cho d[u] > d[t] + c[t, u]. Do trọng số c[t, u] không âm nên d[u] > d[t], trái với cách chọn d[u] là nhỏ nhất. Tất nhiên trong lần lặp đầu tiên thì S là đỉnh được cố định nhãn do d[S] = 0.

-Buớc 3: Kết hợp với việc lưu vết đường đi trên từng buớc sửa nhãn, thông báo đường đi ngắn nhất tìm được hoặc cho biết không tồn tại đường đi (d[F] = +∞).

Độ phức tạp tính toán: Mỗi bước lặp sẽ chọn ra một đỉnh trong n đỉnh để tối ưu, sử dụng cấu trúc dữ liệu HEAP thì chi phí để chọn đỉnh đó là O(1), chi phí để điều chỉnh lại HEAP sau mỗi bước là O(logn). Vậy độ phức tạp của thuật toán Dijkstra là O(nlogn).

# CHƯƠNG TRÌNH VÀ KẾT QUẢ

## Tổ chức chương trình

## Ngôn ngữ cài đặt: C

## Kết quả

### Giao điện chính của chương trình

### Kết quả thực thi của chương trình

Mô tả kết quả thực hiện chương trình.

### Nhận xét đánh giá

# KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

## Kết luận

## Hướng phát triển

TÀI LIỆU THAM KHẢO

“Lí thuyết đồ thị” -Lê Minh Hoàng

Thuật toán sắp sếp Selection Sort, Quick Sort và các thuật toán liên quan đến HEAP tham khảo trên <https://www.geeksforgeeks.org/>

PHỤ LỤC

Sinh viên bỏ Code từng phần vào đây.{*Font: Time New Roman; thường; cỡ chữ: 12; dãn dòng: 1,3; căn lề: justified}*