**BỘ CÔNG THƯƠNG**

**TRƯỜNG ĐẠI HỌC CÔNG THƯƠNG THÀNH PHỐ HỒ CHÍ MINH**

**KHOA CÔNG NGHỆ THÔNG TIN**

-----o0o-----



**TÊN ĐỀ TÀI: TIỂU LUẬN VỀ THUẬT TOÁN DIJKSTRA**

|  |  |
| --- | --- |
| **Nhóm:**  Trưởng nhóm: Nguyễn Ngọc Thùy Trang  Thành viên:  1. Nguyễn Thị Hồng Điệp  2. Võ Minh Phú  3. Nguyễn Hoàng Tuấn  4. Phạm Bá Triết | **Giảng viên hướng dẫn:**  Nguyễn Hải Yến |

# LỜI CẢM ƠN

Nhóm em sẽ không hoàn thành bài tiểu luận nếu không có sự hướng dẫn và chỉ bảo của cô giáoNguyễn Hải Yến. Nhóm em xin chân thành cảm ơn sự hương dẫn của cô.

Lần đầu tiên nhóm em nghiên cứu khoa học chắc chắn đề tài của chúng em không tránh khỏi những thiếu sót, vì vậy em rất mong sự đóng góp ý kiến củ cô để đề tài của chúng em được hoàn thiện hơn. Một lần nữa em xin chân thành cảm ơn công lao dạy dỗ chỉ bảo của cô giáo. Kính chúc cô mạnh khỏe, tiếp tục đạt được nhiều thắng lợi trong nghiên cứu khoa học và sự nghiệp trồng người.

*Em xin chân thành cảm ơn!*

Thành phố Hồ Chí Minh, tháng 11 năm 2023

Nhóm thực hiện

Nhóm 7

# LỜI CAM ĐOAN

Tên nhóm : Nhóm 7

Chúng em xin cam đoan:

1. Đề tài “**Thuật toán Dijkstra**” là kết quả nghiên cứu riêng của nhóm 7, dưới sự hướng dẫn của cô **Nguyễn Hải Yến** và tham khảo một số nguồn tài liệu trên Internet.
2. Bài tiểu luận hoàn toàn không sao chép từ các tài liệu có sẵn
3. Kết quả nghiên cứu không trùng với các tác giả khác.

Nếu sai, nhóm 7 xin hoàn toàn chịu trách nhiệm!

Thành phố Hồ Chí Minh, tháng 11 năm 2023

Người cam đoan

Nhóm 7

**MỤC LỤC**

[LỜI CẢM ƠN 1](#_Toc149934201)

[LỜI CAM ĐOAN 1](#_Toc149934202)

[PHẦN MỞ ĐẦU 3](#_Toc149934203)

[1. Lý do chọn đề tài 3](#_Toc149934204)

[2. Mục đích và nhiệm vụ nghiên cứu: 3](#_Toc149934205)

[3. Đối tượng và phạm vi nghiên cứu 3](#_Toc149934206)

[4. Phương pháp nghiên cứu 3](#_Toc149934207)

[CÁC KHÁI NIỆM CƠ BẢN VỀ LÝ THUYẾT ĐỒ THỊ 4](#_Toc149934208)

[1. Định nghĩa đồ thị 4](#_Toc149934209)

[2. Một số thuật ngữ 4](#_Toc149934212)

[3. Định nghĩa đường đi, chu trình, đồ thị liên thông 5](#_Toc149934226)

[4. Các khái niệm mở đầu về đề tài cần đề cập đến 6](#_Toc149934234)

[GIẢI THUẬT\_LƯU ĐỒ THUẬT TOÁN DIJKSTRA 9](#_Toc149934235)

[**1.** Thuật toán Dijkstra 9](#_Toc149934236)

[2. Độ phức tạp của giải thuật Dijkstra 13](#_Toc149934237)

[3. Lưu đồ thuật toán Dijkstra 14](#_Toc149934238)

[TÀI LIỆU THAM KHẢO 15](#_Toc149934239)

**PHẦN MỞ ĐẦU**

1. **Lý do chọn đề tài**

Lý thuyết đồ thị là một lĩnh vực khoa học đã được phát triển từ lâu và hiện đại hóa với nhiều ứng dụng quan trọng trong các ngành khoa học kỹ thuật như Máy tính, Điện tử, Hóa học, Ngôn ngữ học, Kinh tế học và nhiều lĩnh vực khác. Nhiều khái niệm trong lý thuyết đồ thị xuất phát từ các vấn đề thực tế như tìm đường đi ngắn nhất, chu trình, và các vấn đề tương tự. Điều này đã tạo ra một liên kết mạnh mẽ giữa lý thuyết đồ thị và các ngành khoa học khác, mang lại những giải pháp sáng tạo cho các bài toán phức tạp trong thực tế.

Thuật toán Dijkstra, một trong những thuật toán quan trọng trong lý thuyết đồ thị, đã thu hút sự chú ý với tính ngắn gọn và ứng dụng đa dạng của nó. Khả năng tìm đường đi ngắn nhất giữa các đỉnh của đồ thị đã giúp giải quyết nhiều vấn đề thực tế quan trọng như quản lý đường giao thông, tối ưu hóa mạng lưới viễn thông, lập kế hoạch vận tải, và nhiều ứng dụng khác trong cuộc sống hàng ngày.

Với những lý do này, nhóm chúng em đã quyết định chọn đề tài "Thuật toán Dijkstra " để nghiên cứu. Chúng em mong muốn không chỉ hiểu rõ về cách thuật toán này hoạt động một cách chi tiết và chính xác, mà còn muốn khám phá sâu hơn về cách áp dụng nó vào các bài toán thực tế để tối ưu hóa hiệu suất và giải quyết các thách thức trong cuộc sống hàng ngày. Chúng em hy vọng rằng nghiên cứu của chúng em không chỉ đem lại kiến thức chuyên sâu về lý thuyết đồ thị mà còn góp phần vào việc giải quyết các vấn đề thực tế một cách hiệu quả và sáng tạo.

1. **Mục đích và nhiệm vụ nghiên cứu:**

* Trình bày lý thuyết đồ thị và thuật toán Dijkstra:
* Hiểu rõ lý thuyết đồ thị và cách thuật toán Dijkstra hoạt động.
* Khám phá các biến thể của Dijkstra và cách chúng được ứng dụng.
* Nâng cao hiểu biết về ưu điểm và hạn chế của Dijkstra:
* Đánh giá sâu hơn về ưu và nhược điểm của Dijkstra, đặc biệt trong các đồ thị phức tạp.
* Đề xuất cách tối ưu hóa và áp dụng biến thể của Dijkstra để giải quyết các thách thức này.

1. **Đối tượng và phạm vi nghiên cứu**

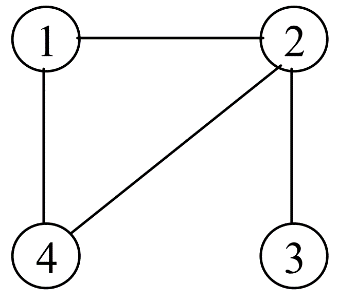
Đối tượng nghiên cứu và phạm vi nghiêm cứu của đề tài là Thuật toán Dijkstra và một số ứng dụng tiêu biểu của thuật toán này trong các lĩnh vực khác nhau.

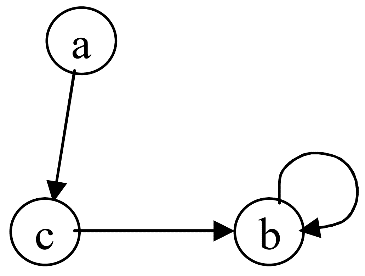
1. **Phương pháp nghiên cứu**

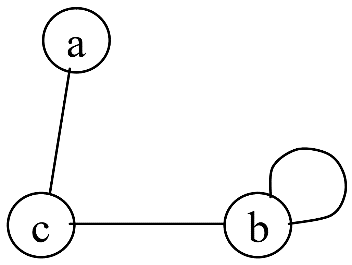
Cơ bản sử dụng phương pháp nghiên cứu tài liệu (sách, báo, các mục trên internet có liên quan đến đề tài) để thu thập thông tin nhằm phân tích, hệ thống lý thuyết, các thuật toán về đường ngắn nhất, ứng dụng phục vụ cho đề tài.

# CÁC KHÁI NIỆM CƠ BẢN VỀ LÝ THUYẾT ĐỒ THỊ

1. **Định nghĩa đồ thị**

Cho đồ thị G = (V,E) là một bộ gồm hai tập V (vertices - tập đỉnh), với V ≠ ∅ và tập E (edges – tập cạnh), với mỗi cạnh tương ứng với hai đỉnh.

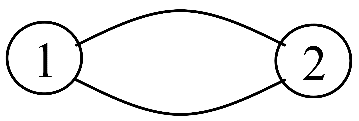
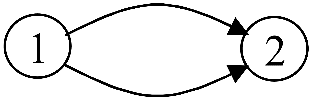
1. *Đơn đồ thị vô hướng* G = (V,E) gồm V là tập đỉnh, E là tập các cạnh không có thứ tự gồm hai phần tử khác nhau của V gọi là các cạnh.
2. *Đa đồ thị vô hướng* G = (V,E) gồm V là tập đỉnh E là tập các cạnh không có thứ tự gồm hai phần tử khác nhau của V gọi là các cạnh.
3. **Một số thuật ngữ**
4. *Khuyên:* Cạnh/cung được gọi là khuyên nếu có đỉnh đầu trùng với đỉnh cuối.
5. ***Khuyên tại đỉnh b của các đồ thị***



Hình 1: Đồ thị vô hướng

Hình 2: Đồ thị có hướng

1. *Cạnh/cung song song:* Hai cạnh/cung được gọi là song song (hay lặp) nếu chúng có 2 đỉnh trùng nhau.



Hình 4: Cạnh có hướng song song

Hình 3: Cạnh vô hướng song song

1. *Đỉnh kề:* Nếu (u,v) là một cung của đồ thị có hướng thì đỉnh v được gọi là kề của đỉnh u. Nếu (u,v) là một cạnh của đồ thị vô hướng thì đỉnh v kề với đỉnh u và đỉnh u cũng kề với đỉnh v.
2. *Cạnh kề:* *Hai cạnh/cung được gọi là kề nhau nếu chúng có chung một đỉnh*.
3. *Cạnh liên thuộc:* *Cạnh (u,v) được gọi là liên thuộc (hay kề) với 2 đỉnh u, v*.
4. *Bậc của đỉnh:* Số cạnh liên thuộc với đỉnh v gọi là bậc của v, ký hiệu d(v).
5. Đối với đỉnh có khuyên thì bậc tính là 2 cho mỗi khuyên.
6. *Cung vào/ra:* Cung (u,v) gọi là cung ra khỏi đỉnh u và là cung vào đỉnh v.
7. *Bậc của đỉnh:* Cho đồ thị vô hướng G = (V,E). Bậc của đỉnh v, ký hiệu là deg(v) là số cạnh kề với v, trong đó một khuyên (vòng) tại một đỉnh được đếm hai lần.
   * + - Đỉnh có bậc bằng 0 🡪 Đỉnh cô lập.
       - Đỉnh có bậc bằng 1 🡪 Đỉnh treo.
       - Đồ thị mà mọi đỉnh đều là đỉnh cô lập 🡪 đồ thị rỗng.

### *Bậc vào của đỉnh*:

* + Bậc vào của đỉnh v ký hiệu: degin(v) là số cung có đỉnh cuối là v.
  + Bậc ra của đỉnh v ký hiệu: degout(v) là số cung có đỉnh đầu là v.

**Deg (v) = degin(v) + degout(v)**

## 

Hình 5: Đỉnh c (đỉnh cô lập) và đỉnh e (đỉnh treo)

## Định nghĩa đường đi, chu trình, đồ thị liên thông

1. *Đường đi****:*** Đường đi trong đồ thị là một dãy các đỉnh v1, v2, v3,…, vn với (vi, vi+1) ∈ E, i=1, 2,.., n. Số cạnh của đường đi được gọi là độ dài của đường đi.

Tóm lại: Đường đi là dãy các đỉnh trong đó 2 đỉnh liên tiếp có cạnh nối.

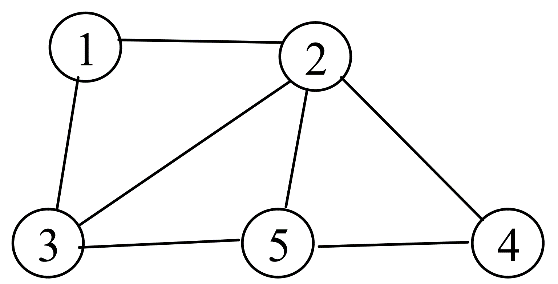
* Đường đi đơn là đường đi mà không có cạnh nào xuất hiện quá một lần.
* Đường đi sơ cấp là đường đi mà không có đỉnh nào xuất hiện quá một lần.

VD: Trong đồ thị dưới đây thì (1, 2, 5, 4, 2, 1) không là đường đi đơn do cạnh (a, b) (b, a) có mặt trong đường đi hai lần. Và cũng không là đường đi sơ cấp vì đỉnh 1 và đỉnh 2 xuất hiện > 1 lần trong đường đi.

1. *Chu trình:*Là đường đi mà có đỉnh đầu trùng với đỉnh cuối.

Một đường đi hay chu trình được gọi là đơn nếu không có cạnh/cung bị lặp lại, gọi là sơ cấp nếu không có đỉnh nào bị lặp lại.

1. *Chu trình đơn****:*** chu trình mà không có cạnh nào xuất hiện quá một lần.
2. *Chu trình sơ cấp:* chu trình mà không có đỉnh nào xuất hiện quá một lần.

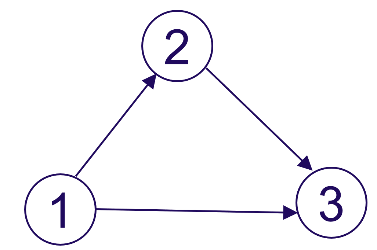
****

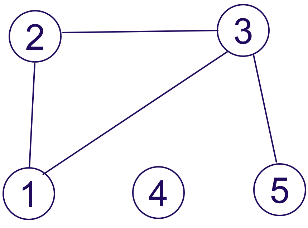
Hình 6: Đồ thị vô hướng

Một số đường đi, chu trình và đối chu trình trên đồ thị như sau:

* 1, 3, 5, 2, 3, 5, 4 (đường đi không đơn vì có cạnh (3,5) bị lặp lại).
* 1, 2, 3, 5, 2, 4 (đường đi đơn vì không có cạnh nào lặp lại nhưng không sơ cấp vì có đỉnh 2 bị lặp lại).
* 1, 2, 3, 5, 4 (đường đi sơ cấp).
* 4, 5, 2, 3, 5, 2, 4 (Chu trình không đơn vì có cạnh (5,2) bị lặp lại).

1. *Đồ thị liên thông****:*** Một đồ thị được gọi là liên thông nếu hai đỉnh bất kỳ luôn có đường đi.

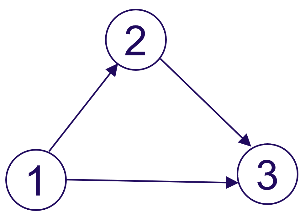
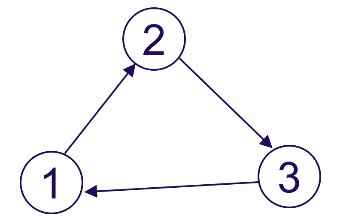




Hình 8: Không liên thông

Hình 7: Không liên thông

1. *Liên thông yếu****:*** là đồ thị có hướng không liên thông, nhưng đồ thị vô hướng tương ứng liên thông.
2. *Liên thông mạnh****:*** là đồ thị có hướng nếu 2 đỉnh bất kì u, v luôn có đường đi từ u tới v.



Hình 9: Liên thông mạnh

Hình 10: Liên thông yếu

## 4. Các khái niệm mở đầu về đề tài cần đề cập đến

1. ***Đồ thị có trọng số***

* Đồ thị G =(V,E) gọi là đồ thị có trọng số nếu mỗi cạnh (cung) được gán với một số thực w(e). Ta gọi w(e) là trọng lượng của e. w(e) =∞ nếu e ∉ E.
* Độ dài của đường đi từ u đến v bằng tổng độ dài các cạnh mà đường đi qua.
* Khoảng các giữa hai đỉnh u, v là độ dài ngắn nhất của các đường đi từ u đến v

1. ***Tìm đường đi ngắn nhất xuất phát từ một đỉnh***

- **Bài toán tìm đường đi ngắn nhất** trên đồ thị dưới dạng tổng quát có thể được phát biểu dưới dạng sau: Tìm đường đi ngắn nhất từ một đỉnh xuất phát s∈V(đỉnh nguồn) đến đỉnh cuối t∈V(đỉnh đích). Đường đi như vậy được gọi là đường đi ngắn nhất từ s đến t, độ dài của đường đi d(s,t) được gọi là khoảng cách ngắn nhất từ s đến t (trong trường hợp tổng quát d(s,t) có thể âm). Nếu như không tồn tại đường đi từ s đến t thì độ dài đường đi d(s,t)=∞. Nếu như mỗi chu trình trong đồ thị đều có độ dài dương thì trong đường đi ngắn nhất sẽ không có đỉnh nào bị lặp lại, đường đi như vậy được gọi là đường đi cơ bản. Nếu như đồ thị tồn tại một chu trình nào đó có độ dài âm, thì đường đi ngắn nhất có thể không xác định, vì ta có thể đi qua chu trình âm đó một số lần đủ lớn để độ dài của nó nhỏ hơn bất kỳ một số thực cho trước nào.

1. ***Thuật toán Dijkstra\_ Bài toán và ví dụ cụ thể***

Thuật toán tìm đường đi ngắn nhất từ đỉnh s đến các đỉnh còn lại được Dijkstra đề nghị áp dụng cho trường hợp đồ thị có hướng với trọng số không âm. Thuật toán được thực hiện trên cơ sở gán tạm thời cho các đỉnh. Nhãn của mỗi đỉnh cho biết cận trên của độ dài đường đi ngắn nhất tới đỉnh đó. Các nhãn này sẽ được biến đổi (tính lại) nhờ một thủ tục lặp, mà ở mỗi bước lặp một số đỉnh sẽ có nhãn không thay đổi, nhãn đó chính là độ dài đường đi ngắn nhất từ s đến đỉnh đó.

Thuật toán có thể được mô tả bằng thủ tục Dijkstra như sau:

void Dijkstra()

{

int d[n]; // Mảng chứa khoảng cách tạm thời từ điểm bắt đầu đến các điểm khác

int truoc[n]; // Mảng chứa đỉnh trước đỉnh hiện tại trong đường đi ngắn nhất

int T[n]; // Tập đỉnh có nhãn tạm thời

int u, minDist, i, j;

// Khởi tạo nhãn tạm thời cho các đỉnh

for (i = 0; i < n; i++)

{

d[i] = INF; // Đặt khoảng cách tạm thời từ s đến các đỉnh là vô cùng lớn

truoc[i] = -1; // Đỉnh trước của các đỉnh là không xác định ban đầu

T[i] = 1; // Khởi tạo tập đỉnh có nhãn tạm thời, 1 là true, tức là các đỉnh đều thuộc T ban đầu

}

int s = 0;

d[s] = 0; // Khoảng cách từ s đến chính nó là 0

// Bước lặp

while (1)

{

minDist = INF;

// Tìm đỉnh u thuộc T sao cho d[u] là nhỏ nhất

for (i = 0; i < n; i++)

{

if (T[i] && d[i] < minDist)

{

minDist = d[i];

u = i;

}

}

if (minDist == INF)

{

// Nếu không có đỉnh nào còn lại trong T, thoát khỏi vòng lặp

break;

}

T[u] = 0; // Đánh dấu u đã được xử lý

// Cập nhật nhãn tạm thời của các đỉnh kề với u

for (j = 0; j < n; j++)

{

if (T[j] && (d[u] + cost[u][j] < d[j])) {

d[j] = d[u] + cost[u][j];

truoc[j] = u;

}

}

}

// In ra khoảng cách ngắn nhất từ đỉnh s đến các đỉnh còn lại

printf("Khoang cach ngan nhat tu dinh %d:\n", s);

for (i = 0; i < n; i++)

{

printf ("%d -> %d: %d\n", s, i, d[i]);

}

}

**Định lý 1**. Thuật toán Dijkstra tìm được đường đi ngắn nhất trên đồ thị sau thời gian cỡ O(n2).

**Chứng minh.**

Trước hết ta chứng minh là thuật toán tìm được đường đi ngắn nhất từ đỉnh s đến các đỉnh còn lại của đồ thị. Giả sử ở một bước lặp nào đó các nhãn cố định cho ta độ dài các đường đi ngắn nhất từ s đến các đỉnh có nhãn cố định, ta sẽ chứng minh rằng ở lần gặp tiếp theo nếu đỉnh u\* thu được nhãn cố định d(u\*) chính là độ dài đường đi ngắn nhất từ s đến u\*.

Ký hiệu S1 là tập hợp các đỉnh có nhãn cố định còn S2 là tập các đỉnh có nhãn tạm thời ở bước lặp đang xét. Kết thúc mỗi bước lặp nhãn tạm thời d(u\*) cho ta độ dài của đường đi ngắn nhất từ s đến u\* không nằm trọng trong tập S1, tức là nó đi qua ít nhất một đỉnh của tập S2. Gọi z ∈ S2 là đỉnh đầu tiên như vậy trên đường đi này. Do trọng số trên các cung là không âm, nên đoạn đường từ z đến u\* có độ dài L>0 và

d(z) < d(u\*) – L < d(u\*).

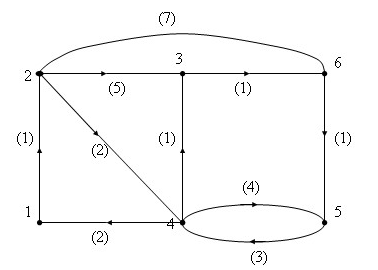
Bất đẳng thức này là mâu thuẫn với cách xác định đỉnh u\* là đỉnh có nhãn tạm thời nhỏ nhất. Vậy đường đi ngắn nhất từ s đến u\* phải nằm trọn trong S1, và vì thế, d[u\*] là độ dài của nó. Do ở lần lặp đầu tiên S1 = { s} và sau mỗi lần lặp ta chỉ thêm vào một đỉnh u\* nên giả thiết là d(v) cho độ dài đường đi ngắn nhất từ s đến v với mọi v ∈ S1 là đúng với bước lặp đầu tiên. Theo qui nạp suy ra thuật toán cho ta đường đi ngắn nhất từ s đến mọi đỉnh của đồ thị.

Bây giờ ta sẽ đánh giá số phép toán cần thực hiện theo thuật toán. Ở mỗi bước lặp để tìm ra đỉnh u cần phải thực hiện O(n) phép toán, và để gán nhãn lại cũng cần thực hiện một số lượng phép toán cũng là O(n). thuật toán phải thực hiện n-1 bước lặp, vì vậy thời gian tính toán của thuận toán cỡ O(n2).

Định lý được chứng minh.

Khi tìm được độ dài của đường đi ngắn nhất d[v] thì đường đi này có thể tìm dựa vào nhãn Truoc[v], v∈ V, theo qui tắc giống như chúng ta đã xét trong chương 3.

**Ví dụ.** Tìm đường đi ngắn nhất từ 1 đến các đỉnh còn lại của đồ thị ở *hình 11*.



**Hình 2.** Minh hoạ thuật toán Dijkstra

Kết quả tính toán theo thuật toán được trình bày theo bảng dưới đây. Qui ước viết hai thành phần của nhãn theo thứ tự: d[v]. Đỉnh được đánh dấu \* là đỉnh được chọn để cố định nhãn ở bước lặp đang xét, nhãn của nó không biến đổi ở các bước tiếp theo, vì thế ta đánh dấu -.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Bước lặp | Đỉnh 1 | Đỉnh 2 | Đỉnh 3 | Đỉnh 4 | Đỉnh 5 | Đỉnh 6 |
| Khởi tạo | 0,1 | 1,1\* | ∞,1 | ∞,1 | ∞,1 | ∞,1 |
| 1 | - | - | 6,2 | 3,2\* | ∞,1 | 8,2 |
| 2 | - | - | 4,4\* | - | 7,4 | 8,2 |
| 3 | - | - |  | - | 7,4 | 5,3\* |
| 4 | - | - |  | - | 6,6\* | - |
| 5 |  |  |  |  |  |  |

Chú ý:

Nếu chỉ cần tìm đường đi ngắn nhất từ s đến một đỉnh t nào đó thì có thể kết thúc thuật toán khi đỉnh t trở thành có nhãn cố định.

**GIẢI THUẬT\_LƯU ĐỒ THUẬT TOÁN DIJKSTRA**

1. **Thuật toán Dijkstra**
   1. ***Phát biểu bài toán***

* Xét đồ thị G=<V, E>:
* Với mỗi cạnh (u, v) **∈ E, ta đặt tương ứng với nó một A[u][v] được gọi là trọng số của cạnh.**
* Ta sẽ đặt A[u,v] = ∞ nếu (u, v) ∉ E. Nếu dãy v0, v1,....,vk là một đường đi trên G thì độ dài của đường đi của nó là.
* **Bài toán dạng tông quát:**
* Tìm đường đi ngắn nhất tại một đỉnh xuất phát s**∈V (Đỉnh nguồn) đến đỉnh cuối t∈V (đỉnh nghịch).**
* **Đường đi như vậy được gọi là đường đi ngắn nhất từ s đến t.**
* **Độ dài của đường đi d(s,t) được gọi khoảng cách ngắn nhất từ s đến t (trong trường hợp tổng quát (s,t) có thể âm).**
* **Nếu như không tồn tại từ s đến t thì độ dài đường đi d(s,t) =** ∞.
  1. ***Ý tưởng của thuật toán***

**Bước 1** : Từ đỉnh gốc khởi tạo khoảng cách tới chính nó là 0 , khởi tạo khoảng cách nhỏ nhất ban đầu tới các đỉnh khác là vô cực. Ta được danh sách khoảng cách tới các đỉnh từ đỉnh gốc .

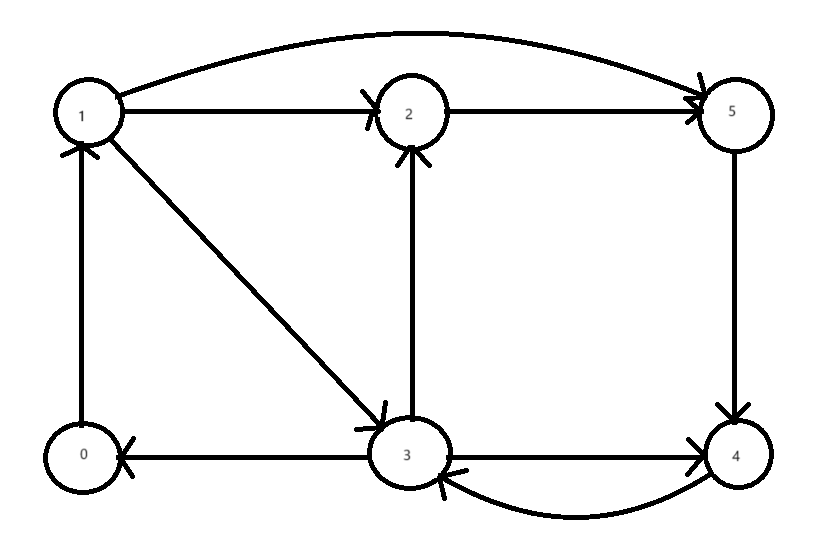
**Bước 2** : Chọn đỉnh v có khoảng cách nhỏ nhất trong danh sách này và ghi nhận . Các lần sau sẽ không xét tới đỉnh này nữa .

**Bước 3** : Lần lượt xét các đỉnh kề của v. Nếu khoảng cách từ đỉnh gốc tới đỉnh kề của v nhỏ hơn khoảng cách hiện tại đang được ghi nhận thì cập nhật lại khoảng cách từ đỉnh gốc tới đỉnh kề của v .

**Bước 4** : Sau khi xét tất cả đỉnh kề của v, ta sẽ được danh sách khoảng cách tới các đỉnh đã được cập nhật. Quay lại bước 2 với danh sách này. Thuật toán sẽ kết thúc khi chọn được khoảng cách nhỏ nhất từ đỉnh nguồn tới đỉnh đích .

* 1. ***Minh họa thuật toán***

Ta sẽ minh họa thuật toán bằng đồ thị như sau :



#define maxn 1000

#define INF 1000000000

//n: số đỉnh , m: số cạnh, s: đỉnh bắt đầu , e : đỉnh đến

//t : loại đồ thị ( 0 : đồ thị vô hướng , 1 : đồ thị có hướng)

int n, m, s, e, t;

int a[maxn][maxn], check[maxn], d[maxn], pre[maxn];

Mảng d[] lưu chi phí đi tử đỉnh nguồn đến đỉnh v đã tìm được .

Mảng check[] nhận 2 giá trị 0, 1 cho biết đỉnh v đã được tối ưu hay chưa .

Mảng a[][] lưu trọng số các cạnh của đồ thị.

Mảng pre[] lưu đường đi đã được tối ưu trước khi đi đến đỉnh đích.

Ta chọn đỉnh nguồn là s = 0, đỉnh đích là e = 4 thì:

Ban đầu khỏi tạo d = [inf, inf, inf, inf, inf, inf ] và check = check[0,0,0,0,0,0]

d[s] = 0 ( d = [0,inf,inf,inf,inf,inf] )

**Bước 1** : Thuật toán sẽ chọn đỉnh 0 vì d[0] = 0 nhỏ nhất và thỏa mãn check[0] = 0. Tiến hành tối ưu cạnh đi ra :

Cạnh (0,1) : d[1] = min(d[1],d[0] + a[0][1]) = (inf , 0 + 1) = 1

Sau bước này : d= [0,1,inf, inf, inf, inf] ,check = [1, 0, 0, 0, 0, 0]

**Bước 2** : Thuật toán sẽ chọn đỉnh 1 vì d[1] = 1 nhỏ nhất và thoả mãn check[1] = 0 .

Tiến hành tối ưu các cạnh đi ra :

Cạnh (1,2) : d[2] = min(d[2],d[1] + a[1][2]) = (inf , 1 + 5 ) = 6

Cạnh (1,3) : d[3] = min(d[3],d[1] + a[1][3]) = (inf , 1 + 2 ) = 3

Cạnh (1,5) : d[5] = min(d[5],d[1] + a[1][5]) = (inf , 1 + 7 ) = 8

Sau bước này : d= [0,1,6, 3, inf, 8] ,check = [1, 1, 0, 0, 0, 0]

**Bước 3** : Thuật toán sẽ chọn đỉnh 3 vì d[3] = 3 nhỏ nhất và thỏa mãn check[3] = 0 .

Tiến hành tối ưu các cạnh đi ra :

Cạnh (3,2) : d[2] = min(d[2],d[3] + a[3][2]) = (inf , 3 + 1 ) = 4

Cạnh (3,4) : d[4] = min(d[4],d[3] + a[3][4]) = (inf , 3 + 4 ) = 7

Sau bước này : d=[0, 1, 4, 3, 7, 8] , check[1, 1, 0, 1, 0, 0]

**Bước 4**  : Thuật toán sẽ chọn đỉnh 2 vì d[2] = 4 nhỏ nhất và thỏa mãn check[2] = 0 .

Sau bước này : d=[0, 1, 4, 3, 7, 8] , check[1, 1, 1, 1, 0, 0]

**Bước 5**  : Thuật toán sẽ chọn đỉnh 4 vì d[4] = 7 nhỏ nhất và thỏa mãn check[4] = 0

Sau bước này : d=[0, 1, 4, 3, 7, 8] , check[1, 1, 1, 1, 1, 0] và đỉnh v đã trùng với đỉnh đích là e = 4 nên thuật toán sẽ dừng ở đây và ta sẽ tìm được đường đi từ 1 🡪 4 có chi phí là 7 .

* 1. ***Cài đặt***

void Dijkstra()

{

for (int i = 1; i <= n; i++)

{

d[i] = inf;

check[i] = 0;

pre[i] = 0;

}

d[s] = 0;

int v = s, dmin;

while (v != e)

{

dmin = inf;

for (int i = 1; i <= n; i++)

{

if (check[i] == 0 && dmin > d[i])

{

dmin = d[i];

v = i;

}

}

if (dmin == inf)

break;

// Đánh dấu đỉnh ngắn nhất

check[v] = 1;

for (int i = 1; i <= n; i++)

{

if (a[v][i] != inf && d[i] > d[v] + a[v][i])

{

d[i] = d[v] + a[v][i];

pre[i] = v;

}

}

}

}

* **Tìm đường đi ngắn nhất :**

if (d[e] == inf)

cout << "Khong ton tai duong di";

else

{

cout << "Chi phi : " << d[e] << "\n";

//Dùng mảng lưu đường đi

int path[maxn], q = 0;

path[q] = e;

while (pre[e] != 0)

{

q++;

path[q] = pre[e];

e = pre[e];

}

for (int i = q; i >= 0; i--)

{

cout << " " << path[i];

}

}

## 2. Độ phức tạp của giải thuật Dijkstra

***a) Trường hợp sử dụng ma trận kề:***

Gọi f(n) là số lần giải thuật Dijkstra khảo sát một cạnh của đồ thị G trong trường hợp xấu

nhất. Khi đó ta có:

f(n) < O(|V|2 )

**Chứng minh**: Cho n = |V|, **B5** là vòng lặp chứa các bước **B2 -> B5**, vòng lặp được thực

hiện đến khi v = b.Vì ở mỗi vòng lặp ta rút ra một đỉnh của V và khởi đầu V có n phần tử, nên

vòng lặp được xử lý nhiều nhất là n lần.

Ở **B2** số đỉnh tối đa được khảo sát là n - 1 đỉnh

Ở **B5** số đỉnh kề tối đa được khảo sát là n -1 đỉnh

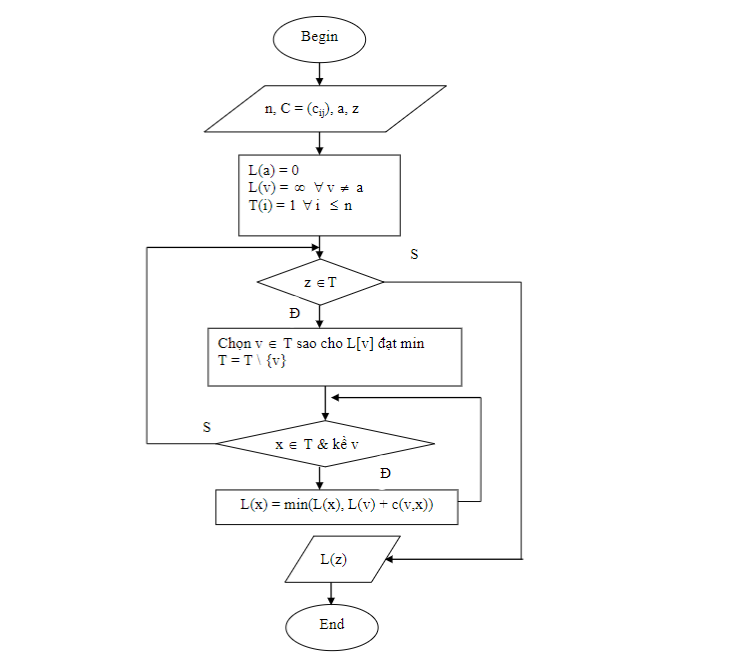
Do đó: f(n)  2(n-1)n < O(|V|2 )

=> Vậy độ phức tạp của giải thuật Dijkstra là O(|V|2 )

***b) Trường hợp sử dụng danh sách kề***

Độ phức tạp của giải thuật Dijkstra là O((|V| + |E|)lg|V|).

## 3. Lưu đồ thuật toán Dijkstra



# TÀI LIỆU THAM KHẢO

1. https://www.youtube.com/watch?v=8Ls1RqHCOPw

2. https://brilliant.org/wiki/dijkstras-short-path-finder/

3. https://www.youtube.com/watch?v=e1xl35UpHb8&t=1357s

4. https://www.youtube.com/watch?v=JqOPBodZmLk&t=610s

5. https://en.wikipedia.org/wiki/Dijkstra%27s\_algorithm

|  |  |  |  |
| --- | --- | --- | --- |
| **TRANG DANH SÁCH NHÓM** | | | |
| **STT** | **Họ và tên** | **MSSV** | **Mức độ hoàn thành (%)** |
| 1. | Nguyễn Thị Hồng Điệp | 2001220976 | 100% |
| 2. | Nguyễn Ngọc Thuỳ Trang | 2001225405 | 100% |
| 3. | Võ Minh Phú | 2033223692 | 100% |
| 4. | Nguyễn Hoàng Tuấn | 2001224555 | 100% |
| 5. | Phạm Bá Triết | 2001210336 | 100% |