BỘ GIÁO DỤC VÀ ĐÀO TẠO

**TRƯỜNG ĐẠI HỌC NHA TRANG**

**KHOA CÔNG NGHỆ THÔNG TIN**

****

**THỰC TẬP CƠ SỞ**

**ĐỀ TÀI: Lập trình mô phỏng thuật toán Dijkstra tìm đường đi ngắn nhất.**

**Giảng viên hướng dẫn : Nguyễn Đình Hưng**

**Sinh viên thực hiện : Nguyễn Khánh Duy**

**Lớp : 61-CNTT-1**

**MSSV : 61133540**

Khánh Hòa: 2021-2022

**MỤC LỤC**

[**PHỤ LỤC** 3](#_Toc90585013)

[**LỜI NÓI ĐẦU** 4](#_Toc90585014)

[**I)** **Giới thiệu:** 5](#_Toc90585015)

[**II)** **Thuật toán** 5](#_Toc90585016)

[**1.** **Giới thiệu giải thuật** 5](#_Toc90585017)

[**2.** **Ý tưởng của thuật toán** 5](#_Toc90585018)

[**3.** **Bài toán tổng quát** 6](#_Toc90585019)

[**4.** **Mô tả thuật toán Dijkstra:** 7](#_Toc90585020)

[**a.** **Độ phức tạp thuật toán:** 7](#_Toc90585021)

[**b.** **Các bước trong thuật toán Dijkstra:** 7](#_Toc90585022)

[**c.** **Ví dụ thuật toán Dijkstra:** 7](#_Toc90585023)

[**III)** **Cài đặt bài toán:** 8](#_Toc90585024)

[**1.** **Thiết kế thuật toán Dijkstra:** 8](#_Toc90585025)

[**2.** **Giải thuật thô:** 9](#_Toc90585026)

[**3.** **Minh họa thuật toán:** 9](#_Toc90585027)

# **PHỤ LỤC**

[Hình 1: Bài toán thực tiễn](#h1)

[Hình 2: Mô tả bằng đồ thị](#h2)

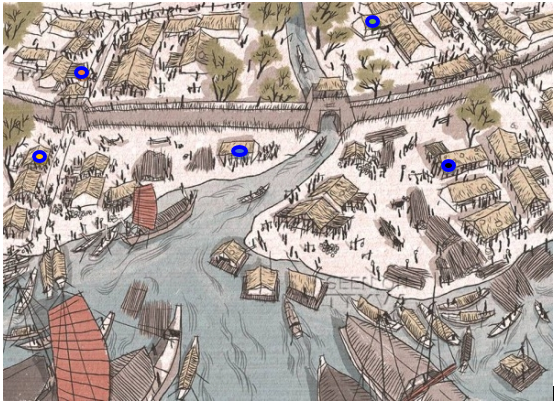
[Hình 3: Ví dụ thuật toán](#h3)

[Hình 4: Đồ thị mô tả](#h4)

[Hình 5: Bước thực hiện](#h5)

# **LỜI NÓI ĐẦU**

Trong đời sống, chúng ta thường gặp những tình huống như sau:để đi từ địa điểm A đến địa điểm B, có nhiều đường đi, nhiều cách đi; có lúc ta chọn đường đi ngắn nhất (theo nghĩa cự ly), có lúc lại cần chọn đường đi nhanh nhất ( theo nghĩa thời gian) và có lúc phải cân nhắc để chọn đường đi rẻ tiền nhất( theo nghĩa chi phí),…

 Hình 1: Bài toán thực tiễn

1. **Giới thiệu:**

Bài toán tìm đường đi ngắn nhất trên đồ thị là một trong những bài toán đa dạng, có nhiều ứng dụng thực tế (như trong Google Maps, hay các bài toán networking, …).

Có ba thuật toán cơ bản của dạng bài tìm đường đi ngắn nhất:

Thuật toán Bellman - Ford.

Thuật toán Dijkstra.

Thuật toán Floyd-Warshall (còn gọi là thuật toán Floyd).

Trong đề tài này thì sẽ nói về thuật toán Dijkstra.

**\*) Khái niệm đồ thị:**

* Đồ thị là một tập các đối tượng gọi là đỉnh nối với nhau bởi các cạnh. Thông thường, đồ thị được vẽ dưới dạng một tập các điểm (đỉnh, nút) nối với nhau bởi các đoạn thẳng (cạnh).
* Đồ thị có hướng G (graph) là một cặp có thứ tự G = (V, A), trong đó:

+ V, tập các đỉnh hoặc nút,

+ A là tập các cặp có thứ tự chứa các đỉnh, được gọi là các cạnh có hướng hoặc cung. Một cạnh a = (x, y) được coi là có hướng từ x tới y; x được gọi là điểm đầu/gốc và y được gọi là điểm cuối/ngọn của cạnh.

1. **Thuật toán**
2. **Giới thiệu giải thuật**

Thuật toán Dijkstra, mang tên của nhà khoa học máy tính người Hà Lan Edsger Dijkstra vào năm 1956 và ấn bản năm 1959, là một thuật toán giải quyết bài toán đường đi ngắn nhất từ một đỉnh đến các đỉnh còn lại của đồ thị có hướng không có cạnh mang trọng số không âm. Thuật toán thường được sử dụng trong định tuyến với một chương trình con trong các thuật toán đồ thị hay trong công nghệ ”Hệ thống định vị toàn cầu (GPS)”.

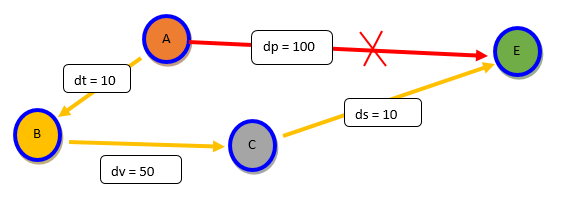
1. **Ý tưởng của thuật toán**

Ví dụ:

Với 4 đỉnh A, B, C, E.

Đường đi từ A đến E với độ dài là dp, đường đi từ A đến B với độ dài là dt, đường đi từ B đến C với độ dài là dv, đường đi từ C đến E với độ dài là ds.

Đường đi từ A đến B, từ B đến C và từ C đến E có thể chọn để đi so với đường đi trực tiếp từ A đến E vì dp > dt + dv + ds.



Hình 2: Ví dụ thuật toán

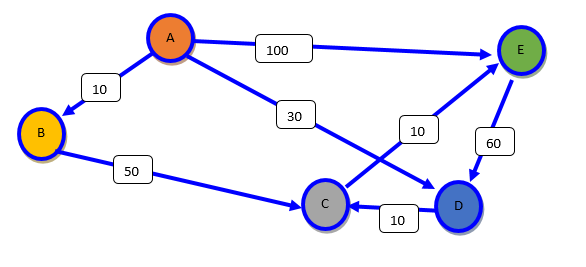
1. **Bài toán tổng quát**

Cho một đồ thị có hướng G= (V, A), một hàm trọng số w: A → [0, ∞) và một đỉnh nguồn x. Cần tính toán được đường đi ngắn nhất từ đỉnh nguồn x đến mỗi đỉnh của đồ thị.

Ví dụ:

Chúng ta có các điểm A, B, C, D, E và khoảng cách là:

Từ A đến B là 10, từ A đến E là 100, từ A đến D là 30, từ B đến C là 50, từ C đến E là 10, từ D đến C là 10, từ E đến D là 60.



Hình 3: Mô tả bằng đồ thị

Chúng ta dùng các đỉnh của đồ thị để mô hình các thành phố và các cạnh để mô hình các đường nối giữa chúng. Khi đó trọng số các cạnh có thể xem như độ dài của các con đường (và độ dài đó là không âm). Chúng ta cần vận chuyển từ thành phố A đến thành phố E.

1. **Mô tả thuật toán Dijkstra:**
   1. **Độ phức tạp thuật toán:**

Thuật toán Dijkstra bình thường sẽ có độ phức tạp là O(n2 + m). Tuy nhiên ta có thể sử dụng kết hợp với cấu trúc heap, khi đó độ phức tạp sẽ là O((m + n)log (n)), nếu dùng Fibonacci Heap thì độ phức tạp giảm xuống còn O(m + n log (n)). Trong đó m là số cạnh, n là số đỉnh của đồ thị đang xét.

* 1. **Các bước trong thuật toán Dijkstra:**

Tập các đỉnh của đồ thị là V, gọi S là tập các đỉnh lấy từ V

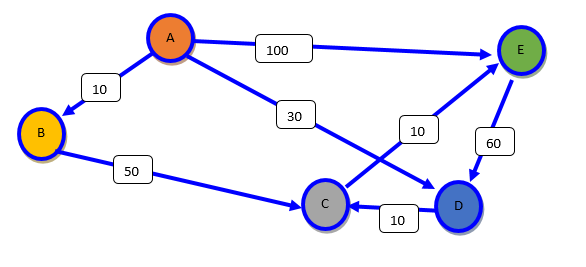
Bước 1:

Chọn một đỉnh nguồn x0 có trong tập V và thêm vào tập S ( S = x0 ).

Bước 2:

Chọn đỉnh w thuộc tập V nhưng chưa có trong tập S sao cho độ dài đường đi ngắn nhất từ x0 đến w là ngắn nhất. Thêm w vào tập S.

Lặp lại bước 2 (n-1) lần và chúng ta có thể rìm được đường đi ngắn nhất từ v0 đến các đỉnh còn lại mà chỉ đi qua các đỉnh đã tồn tại trong S.

* 1. **Ví dụ thuật toán Dijkstra:**

Hình 4: Đồ thị mô tả

Cho tập V gồm 5 đỉnh A, B, C, D, E và có hướng ( giống như đường 1 chiều) và có các trọng số từ đỉnh A đến B là 10, đỉnh A đến D là 30, đỉnh A đến E là 100, đỉnh B đến C là 50, đỉnh C đến E là 10, đỉnh D đến C là 10, đỉnh E đến D là 60.

Còn các đỉnh không nối trực tiếp thì trọng số là vô cùng (∞).

Bước 1:

Giả sử đỉnh nguồn là A 🡪 S={A} và độ dài từ đỉnh A đến các đỉnh còn lại là {10, ∞, 30, 100}

Bước 2:

Chọn ra độ dài nhỏ nhất là 10 tương ứng là đỉnh B vào tập S

Cập nhật lại S={A,B} và độ dài từ các đỉnh còn lại là {10, 60, 30, 100} vì lúc này đỉnh A qua B và đến C nên cập nhật lại là 60.

Chọn độ dài nhỏ nhất là 30 tương ứng với đỉnh D vào tập S

Cập nhật lại S={A,B,D} và độ dài từ A đến các đỉnh còn lại là {10, 40, 30, 90} vì lúc này đỉnh A qua D và đến C là 40 và A qua D đến E là 90.

Chọn độ dài nhỏ nhất là 40 tương ứng với đỉnh C vào tập S

Cập nhật lại S={A,B,D,C} và độ dài từ A đến các đỉnh còn lại là {10, 40, 30, 50} vì lúc này đỉnh A qua D và qua C đến E là 50.

Chọn độ dài nhỏ nhất là 50 tương ứng với đỉnh E vào tập S

Cập nhật lại S={A,B,D,C,E} và độ dài từ A đến các đỉnh còn lại là {10, 40, 30, 50} vì đã đi hết các đỉnh nên dừng lại.

1. **Cài đặt bài toán:**
   1. **Thiết kế thuật toán Dijkstra:**

Tạo ma trận A lưu độ dài các cạnh (số thực), tức là A[x,y] là độ dài của cạnh (x,y), nếu không có cạnh (x,y) thì A[x,y] = ∞.

Mảng S[n] trong đó n là số nguyên lưu các số 0 và 1, đây là một vecto biểu diễn tập hợp S.

Nếu S[x]==1 là x thuộc S, nếu S[x] == 0 là thuộc tập V-S.

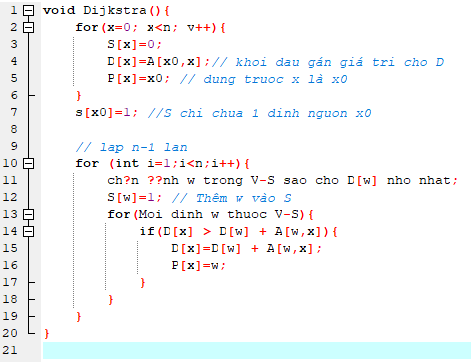
Khởi đầu cho S[x0]=1 nghĩa là x0 thuộc tập S, và các S[x]=0 nghĩa là các x còn lại nằm ngoại tập hợp S. Tại mỗi bước nếu w(đường đi ngắn nhất) được đưa vào S => đặt S[w]=1.

Mảng D[n] để lưu độ dài của đường đi ngắn nhất từ x0 đến mỗi đỉnh của đồ thị.

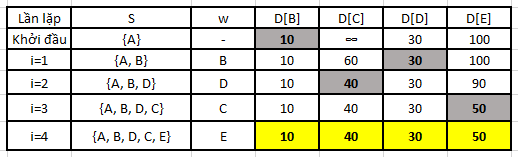
Khởi đầu cho D[x] bằng độ dài của x0 đến x thì D[x]=A[x0,x], nếu không có cạnh từ x0 đến x thì D[x]=∞ .

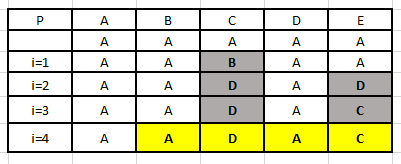
Tại mỗi bước, sẽ cập nhật lại D[x] để lưu độ dài đường đi ngắn nhất từ đỉnh x0 tới đỉnh x, đường đi này chỉ đi qua các đỉnh đã có trong S.

* 1. **Giải thuật thô:**



* 1. **Minh họa thuật toán Dijkstra:**





Hình 5: Bước thực hiện