**Thuật toán Dijkstra**

# Giới thiệu Thuật toán Dijkstra.

Thuật toán Dijkstra được thiết kế để tìm đường đi ngắn nhất giữa các nút trong đồ thị. Nó được thiết kế bởi một nhà khoa học máy tính người Hà Lan, Edsger Wybe Dijkstra, vào năm 1956, khi ông đang tìm con đường ngắn nhất từ ​​Rotterdam đến Groningen. Thuật toán Dijkstras được công bố 3 năm sau đó.

A person wearing glasses

Description automatically generated with medium confidence

Thuật toán Dijkstras có nhiều thay đổi từ khi được công bố và có nhiều biến thể khác nhau ra đời. Ban đầu thuật toán được sử dụng để tính đường đi ngắn nhất giữa 2 đỉnh trong đồ thị. Sau đó được cải tiến để tính đường đi ngắn nhất của đỉnh bắt đầu đến mọi đỉnh còn lại trong đồ thị, điều này tạo ra nhược điểm của thuật toán Dijkstras là không phù hợp với các loại đồ thị lớn.

# Mục đích và ứng dụng Thuật toán Dijkstra.

Thuật toán Dijkstra Tìm đường đi ngắn nhất giữa một đỉnh nguồn và một đỉnh đích.

Thuật toán này được sử dụng trong các thiết bị GPS để tìm đường đi ngắn nhất giữa vị trí hiện tại và điểm đến. Nó có các ứng dụng rộng rãi trong công nghiệp, đặc biệt trong các lĩnh vực yêu cầu modeling networks.

Ứng dụng:

* Google map: tìm đường ngắn nhất giữa 2 địa điểm, vì có nhiều tuyến đường, lối đi khác nhau kết nối chúng nhưng Google map phải hiển thị khoảng cách tối thiểu, vậy nên Thuật toán Dijkstra được sử dụng để tìm khoảng cách tối thiểu giữa hai vị trí dọc theo đường đi.

A person holding a tablet in a car

Description automatically generated with medium confidence

* Mạng xã hội: Cung cấp những đề xuất cho người dùng về danh sách những người có thể quen biết thông qua những thông tin chung của họ.



* Gửi tệp trong mạng Lan: Chọn ra đường đi ngắn nhất giữa máy gửi và máy nhận để tệp được gửi 1 cách nhanh nhất có thể.
* Robotic Path: theo sự phát triển của công nghệ thì máy bay không người láy, ô tô tự hành, robot tự hành được ra đời giúp giảm những tai nạn có nguyên nhân đến từ con người. Điều đó cần ứng dụng thuật toán Dijstras để tìm đường đi ngắn nhất giúp tối ưu đoạn đường di chuyển là ngắn nhất.

A picture containing text

Description automatically generated

# Nguyên lý hoạt động của Thuật toán Dijkstra.

Ban đầu tạo 1 tập hợp các đỉnh đường đi ngắn nhất đã đi qua (khởi tạo là mảng rỗng).

Đặt chi phí cho tất cả các đỉnh trong đồ thị (biểu thị độ dài của đường đi ngắn nhất từ đỉnh bắt đầu đến đỉnh đó). Chi phí của tất cả các đỉnh ban đầu sẽ được khởi tạo là “**infinity**” ngoại trừ đỉnh bắt đầu sẽ có chi phí là **0**, vì không có đường dẫn nào dẫn đến chính nó, được đánh dấu là đỉnh **s**.

Sau đó, lặp lại hai bước chính sau cho đến khi đồ thị được duyệt qua hết các đỉnh.

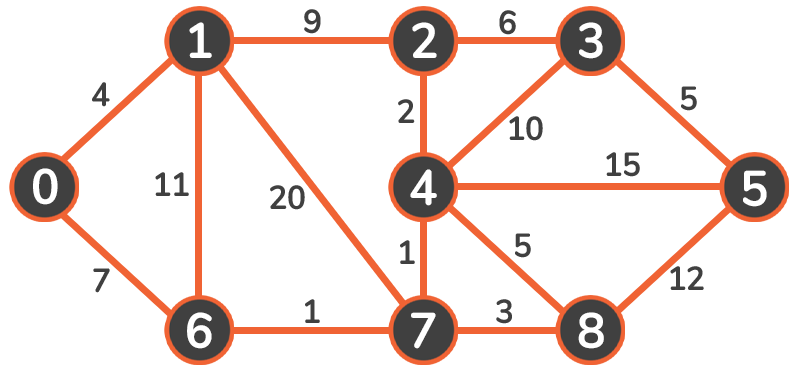
* Chọn một đỉnh có chi phí hiện tại ngắn nhất, thăm nó và thêm nó vào tập các đỉnh đã thăm.
* Cập nhật chi phí của tất cả các đỉnh lân cận của nó chưa được thăm. Đối với mọi cạnh giữa đỉnh n và m nếu cheapestPath(s, n) + cheapestPath(n,m) < cheapestPath(s,m) phải cập nhật đường đi nhỏ nhất giữa s và m bằng cheapestPath(s,n) + cheapestPath(n,m). Thuật toán được kết thúc sau khi tất cả các đỉnh đã được thêm vào đường dẫn.

# Yêu cầu của Thuật toán Dijkstra

Thuật toán Dijkstra chỉ có thể hoạt động với các đồ thị có trọng số dương. Điều này là do, trọng số của các cạnh phải được thêm vào để tìm đường đi ngắn nhất.

Nếu có trọng số âm trong đồ thị, thì thuật toán sẽ không hoạt động đúng. Khi một đỉnh đã được đánh dấu là "đã ghé thăm", đường dẫn hiện tại đến nút đó được đánh dấu là đường ngắn nhất của đỉnh bắt đầu đến đỉnh đó. Và trọng số âm có thể thay đổi điều này nếu tổng chi phí có thể được giảm sau khi bước này xảy ra.

# Ví dụ về Thuật toán Dijkstra

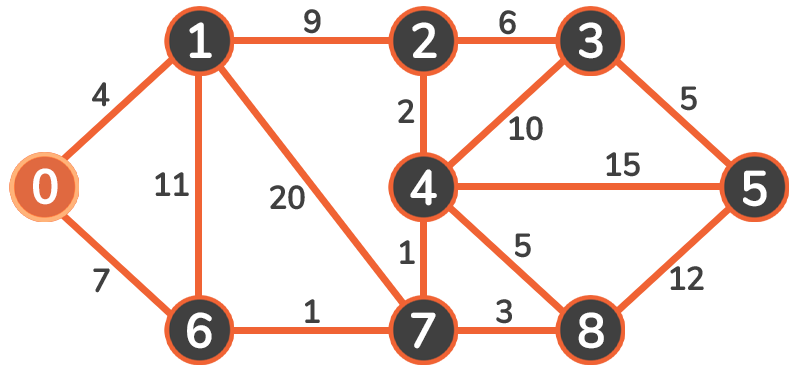


Chọn đỉnh 0 là đỉnh bắt đầu.

Chi phí đi từ đỉnh 0 đến các đỉnh còn lại:

|  |  |  |
| --- | --- | --- |
| Đỉnh | Chi phí để đi đến nó từ đỉnh 0 | Path |
| 0 | 0 | 0 |
| 1 | inf | - |
| 2 | inf | - |
| 3 | inf | - |
| 4 | inf | - |
| 5 | inf | - |
| 6 | inf | - |
| 7 | inf | - |
| 8 | inf | - |

Chọn đỉnh có chi phí tối thiểu là Đỉnh 0. Đánh dấu Đỉnh 0 là đã thăm và thêm nó vào tập hợp các đỉnh đã thăm. Đỉnh bắt đầu sẽ luôn có chi phí thấp nhất vì vậy nó sẽ luôn là đỉnh đầu tiên được thêm vào tập hợp các đỉnh đã thăm.



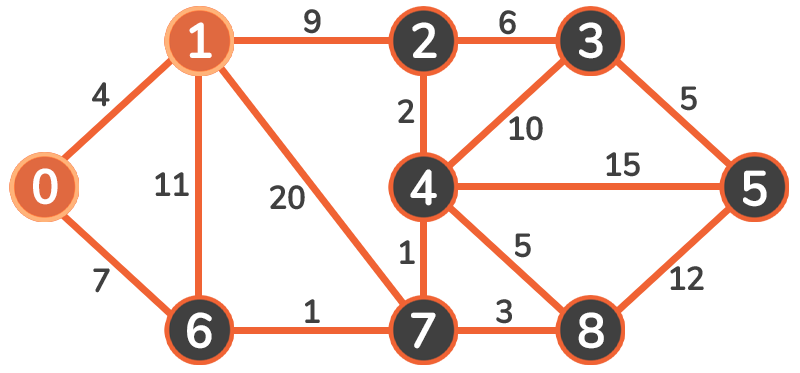
Cập nhật chi phí của các đỉnh liền kề (1 và 6).



Cập nhật chi phí cho các đỉnh như sau:

|  |  |  |
| --- | --- | --- |
| Đỉnh | Chi phí để đi đến nó từ đỉnh 0 | Path |
| 0 | 0 | 0 |
| 1 | 4 | 0, 1 |
| 2 | inf | - |
| 3 | inf | - |
| 4 | inf | - |
| 5 | inf | - |
| 6 | 7 | 0, 6 |
| 7 | inf | - |
| 8 | inf | - |

Tiếp tục với đỉnh có chi phí nhỏ nhất (Đỉnh 1)



Đánh dấu Đỉnh 1 là đã thăm và thêm nó vào tập hợp các đỉnh đã thăm.

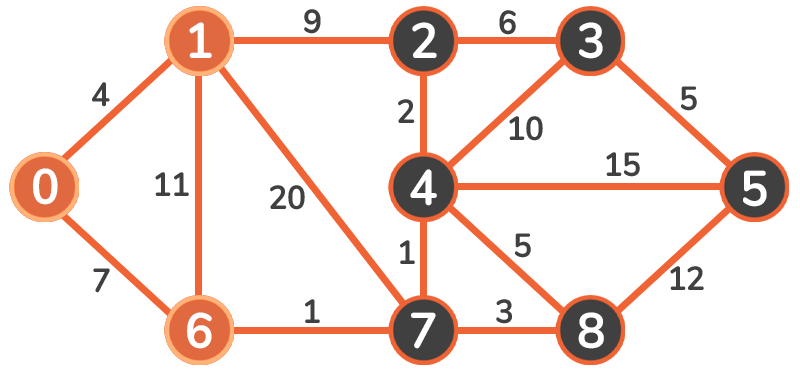
Cập nhật chi phí của các đỉnh liền kề (2, 6 và 7).

* => chi phí mới của đỉnh 2 sẽ là 13.
* => chi phí của đỉnh 6 sẽ vẫn là 7.
* => chi phí mới của đỉnh 7 sẽ là 24.

Cập nhật chi phí cho các đỉnh như sau:

|  |  |  |
| --- | --- | --- |
| Đỉnh | Chi phí để đi đến nó từ đỉnh 0 | Path |
| 0 | 0 | 0 |
| 1 | 4 | 0, 1 |
| 2 | 13 | 0, 1, 2 |
| 3 | inf |  |
| 4 | inf |  |
| 5 | inf |  |
| 6 | 7 | 0, 6 |
| 7 | 24 | 0, 1, 7 |
| 8 | inf |  |

Tiếp tục với đỉnh có chi phí nhỏ nhất (Đỉnh 6)

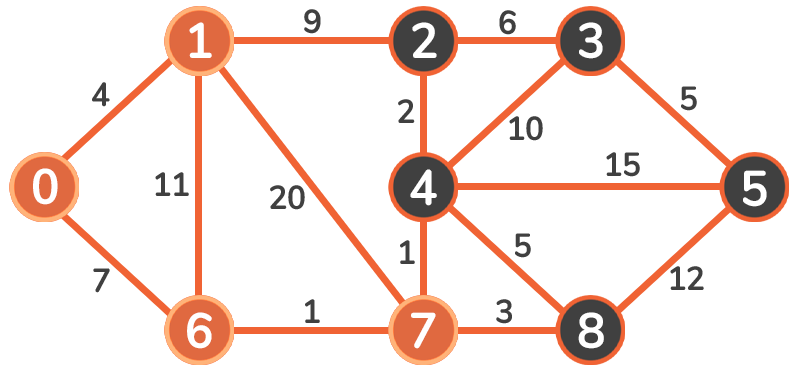


Đánh dấu Đỉnh 6 là đã thăm và thêm nó vào tập hợp các đỉnh đã thăm.

Cập nhật chi phí của các đỉnh liền kề (7).

|  |  |  |
| --- | --- | --- |
| Đỉnh | Chi phí để đi đến nó từ đỉnh 0 | Path |
| 0 | 0 | 0 |
| 1 | 4 | 0, 1 |
| 2 | 13 | 0, 1, 2 |
| 3 | inf |  |
| 4 | inf |  |
| 5 | inf |  |
| 6 | 7 | 0, 6 |
| 7 | 8 | 0, 6, 7 |
| 8 | inf |  |

Tiếp tục với đỉnh có chi phí nhỏ nhất (Đỉnh 7)

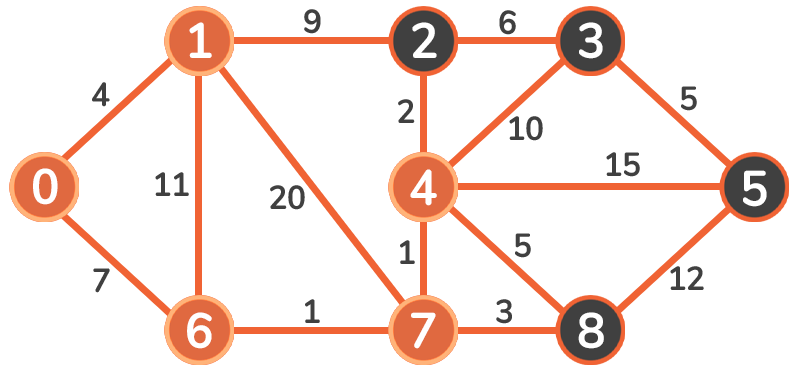


Đánh dấu Đỉnh 7 là đã thăm và thêm nó vào tập hợp các đỉnh đã thăm.

Cập nhật chi phí của các đỉnh liền kề (4, 8).

|  |  |  |
| --- | --- | --- |
| Đỉnh | Chi phí để đi đến nó từ đỉnh 0 | Path |
| 0 | 0 | 0 |
| 1 | 4 | 0, 1 |
| 2 | 13 | 0, 1, 2 |
| 3 | inf |  |
| 4 | 9 | 0, 6, 7, 4 |
| 5 | inf |  |
| 6 | 7 | 0, 6 |
| 7 | 8 | 0, 6, 7 |
| 8 | 11 | 0, 6, 7, 8 |

Tiếp tục với đỉnh có chi phí nhỏ nhất (Đỉnh 4)

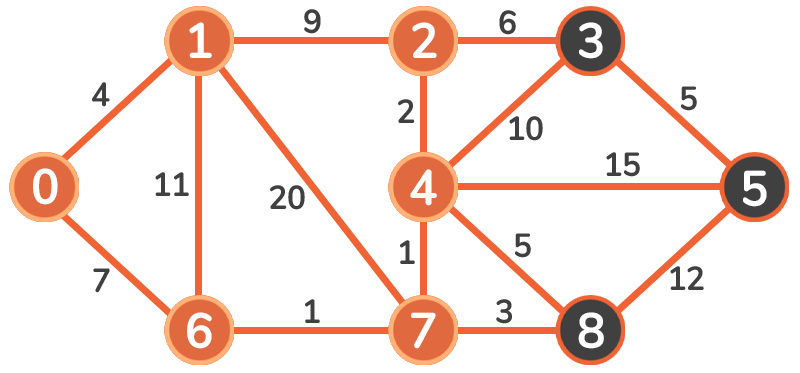


Đánh dấu Đỉnh 4 là đã thăm và thêm nó vào tập hợp các đỉnh đã thăm.

Cập nhật chi phí của các đỉnh liền kề (2, 3, 5, 8).

|  |  |  |
| --- | --- | --- |
| Đỉnh | Chi phí để đi đến nó từ đỉnh 0 | Path |
| 0 | 0 | 0 |
| 1 | 4 | 0, 1 |
| 2 | 11 | 0, 6, 7, 4, 2 |
| 3 | 19 | 0, 6, 7, 4, 3 |
| 4 | 9 | 0, 6, 7, 4 |
| 5 | 24 | 0, 6, 7, 4, 5 |
| 6 | 7 | 0, 6 |
| 7 | 8 | 0, 6, 7 |
| 8 | 11 | 0, 6, 7, 8 |

Tiếp tục với đỉnh có chi phí nhỏ nhất (Đỉnh 2)

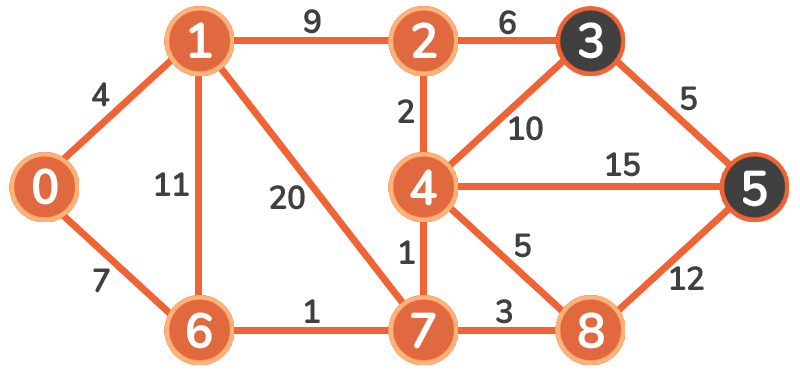


Đánh dấu Đỉnh 2 là đã thăm và thêm nó vào tập hợp các đỉnh đã thăm.

Cập nhật chi phí của các đỉnh liền kề (3).

|  |  |  |
| --- | --- | --- |
| Đỉnh | Chi phí để đi đến nó từ đỉnh 0 | Path |
| 0 | 0 | 0 |
| 1 | 4 | 0, 1 |
| 2 | 11 | 0, 6, 7, 4, 2 |
| 3 | 17 | 0, 6, 7, 4, 2, 3 |
| 4 | 9 | 0, 6, 7, 4 |
| 5 | 24 | 0, 6, 7, 4, 5 |
| 6 | 7 | 0, 6 |
| 7 | 8 | 0, 6, 7 |
| 8 | 11 | 0, 6, 7, 8 |

Tiếp tục với đỉnh có chi phí nhỏ nhất (Đỉnh 8)

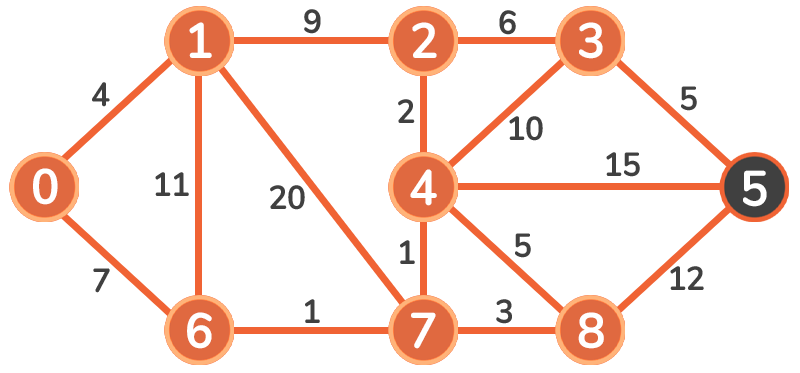


Đánh dấu Đỉnh 8 là đã thăm và thêm nó vào tập hợp các đỉnh đã thăm.

Cập nhật chi phí của các đỉnh liền kề (5).

|  |  |  |
| --- | --- | --- |
| Đỉnh | Chi phí để đi đến nó từ đỉnh 0 | Path |
| 0 | 0 | 0 |
| 1 | 4 | 0, 1 |
| 2 | 11 | 0, 6, 7, 4, 2 |
| 3 | 17 | 0, 6, 7, 4, 2, 3 |
| 4 | 9 | 0, 6, 7, 4 |
| 5 | 23 | 0, 6, 7, 8, 5 |
| 6 | 7 | 0, 6 |
| 7 | 8 | 0, 6, 7 |
| 8 | 11 | 0, 6, 7, 8 |

Tiếp tục với đỉnh có chi phí nhỏ nhất (Đỉnh 3)



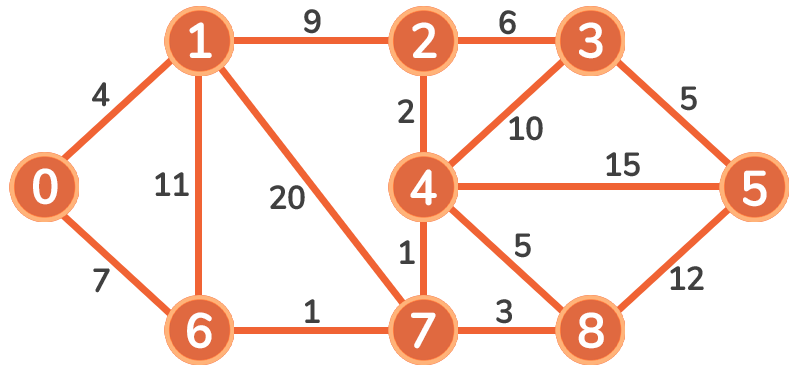
Đánh dấu Đỉnh 3 là đã thăm và thêm nó vào tập hợp các đỉnh đã thăm.

Cập nhật chi phí của các đỉnh liền kề (5).

|  |  |  |
| --- | --- | --- |
| Đỉnh | Chi phí để đi đến nó từ đỉnh 0 | Path |
| 0 | 0 | 0 |
| 1 | 4 | 0, 1 |
| 2 | 11 | 0, 6, 7, 4, 2 |
| 3 | 17 | 0, 6, 7, 4, 2, 3 |
| 4 | 9 | 0, 6, 7, 4 |
| 5 | 22 | 0, 6, 7, 4, 2, 3, 5 |
| 6 | 7 | 0, 6 |
| 7 | 8 | 0, 6, 7 |
| 8 | 11 | 0, 6, 7, 8 |

Tiếp tục với đỉnh có chi phí nhỏ nhất (Đỉnh 3)

Thuật toán được kết thúc sau khi tất cả các đỉnh đã được thêm vào đường dẫn.



|  |  |  |
| --- | --- | --- |
| Đỉnh | Chi phí để đi đến nó từ đỉnh 0 | Path |
| 0 | 0 | 0 |
| 1 | 4 | 0, 1 |
| 2 | 11 | 0, 6, 7, 4, 2 |
| 3 | 17 | 0, 6, 7, 4, 2, 3 |
| 4 | 9 | 0, 6, 7, 4 |
| 5 | 22 | 0, 6, 7, 4, 2, 3, 5 |
| 6 | 7 | 0, 6 |
| 7 | 8 | 0, 6, 7 |
| 8 | 11 | 0, 6, 7, 8 |

# Tài liệu tham khảo

freeCodeCamp. (n.d.). *freeCodeCamp.* Retrieved from freeCodeCamp: https://www.freecodecamp.org/news/dijkstras-shortest-path-algorithm-visual-introduction/

GeeksforGeeks. (n.d.). *GeeksforGeeks.* Retrieved from https://www.geeksforgeeks.org/applications-of-dijkstras-shortest-path-algorithm/

Lukic, M. (n.d.). *StackAbuse.* Retrieved from https://stackabuse.com/dijkstras-algorithm-in-python/

# Bài Tập

1) Thuật toán Dijkstra được dùng để:

A. Tìm đường đi ngắn nhất giữa các cặp đỉnh bất kì của đồ thị.

B. Tìm đường đi ngắn nhất từ một đỉnh đến các đỉnh còn lại của đồ thị

C. Tìm đường đi ngắn nhất giữa hai đỉnh của đồ thị.

D. Tìm đường đi ngắn nhất giữa một đỉnh nguồn và một đỉnh đích.

2) Thuật toán Dijkstra được áp dụng cho:

A. Đồ thị vô hướng hoặc có hướng có trọng số không âm.

B. Đồ thị liên thông có trọng số không âm

C. Đồ thị có hướng có trọng số không âm.

D. Đồ thị vô hướng hoặc có hướng không có chu trình âm