ĐẠI HỌC KINH TẾ TP. HỒ CHÍ MINH

UEH UNIVERSITY

College of Technology and Design

SCHOOL OF BUSINESS INFORMATION TECHNOLOGY

BÁO CÁO

Nhóm sinh viên: Đặng Thị Thu Hiền, Đỗ Thanh Hoa, Bùi Tiến Hiếu, Nguyễn Trương Hoàng, Nguyễn Huy Hoàng

Môn học: Lập trình phân tích dữ liệu

Giảng viên hướng dẫn: Nguyễn An Tế

Đề tài: Nghiên cứu thuật toán Dijkstra cho bài toán tìm đường đi ngắn nhất

Hồ Chí Minh, ngày ... tháng ... năm...

THÀNH VIÊN NHÓM

STT	Họ và tên	MSSV
1	Đặng Thị Thu Hiền	31221025556
2	Bùi Tiến Hiếu	31221026291
3	Đỗ Thanh Hoa	31211025193
4	Nguyễn Huy Hoàng	31221023992
5	Nguyễn Trương Hoàng	31221026058

LỜI NÓI ĐẦU

Nhóm chúng con xin được gửi lời cảm ơn tới thầy Nguyễn An Tế vì những gì thầy đã truyền tải lại cho chúng con ở trên lớp. Những bài giảng của thầy có thể nói rằng rất hay, rất đáng nhớ, rất đáng quý và cực kỳ đáng trân trọng. Những tri thức được truyền lại đó quả thật là những "chất vàng mười" của tri thức mà thầy đã rèn nên thông qua vốn kiến thức uyên bác của thầy. Nhờ có "chất vàng mười" đó mà vốn hiểu biết của chúng con được mở rộng và tư duy được sắc bén hơn theo cách tư duy logic hơn, có hệ thống hơn không chỉ riêng đối với ngành học mà còn ở những lĩnh vực khác. Đó là nền tảng kiên cố làm tiền đề để nhóm có thể tự tìm hiểu cho những kiến thức liên quan đến lĩnh vực của đề tài về sau này. Những giá trị mà thầy đã mang đến sẽ còn mãi với thời gian, bởi đó là những gì thật sự quý và đáng trân trọng, thứ mang trong mình sức mạnh vượt qua được "sự băng hoại của thời gian".

Một lần nữa, nhóm 03 xin chân thành cảm ơn thầy Nguyễn An Tế vì tất cả những gì thầy đã mang đến cho chúng con. Nhóm chúng con chúc thầy luôn được khỏe mạnh và dồi dào sức khỏe. Mong rằng không chỉ thầy và những người xung quanh thầy sẽ luôn được bình an và hạnh phúc.

TP. HCM, Ngày 21 Tháng 11 Năm 2024,

Nhóm 03

MỤC LỤC

DANH MỤC BẢNG BIỂU

DANH MỤC HÌNH ẢNH

CHƯƠNG I: TỔNG QUAN ĐỀ TÀI

1.1. Tổng quan về bài toán tìm đường đi ngắn nhất

Bài toán tìm đường đi ngắn nhất (Shortest Path Problem) là một trong những vấn đề cơ bản và quan trọng trong lý thuyết đồ thị. Bài toán tìm đường đi ngắn nhất yêu cầu xác định đường đi có tổng trọng số nhỏ nhất giữa hai đỉnh (nodes) trong một đồ thị có trọng số. Đồ thị có thể là có hướng hoặc không có hướng, và trọng số của các cạnh (edges) có thể là dương hoặc âm.

Bài toán này có thể chia làm 2 loại. Bài toán đường đi ngắn nhất nguồn đơn là bài toán tìm một đường đi giữa hai đỉnh trong đồ thị G(V,E) có trọng số cạnh và hai đỉnh u, v thuộc V sao cho tổng các trọng số của các cạnh tạo nên đường đi đó là nhỏ nhất. Bài toán tìm đường đi ngắn nhất giữa tất cả các cặp đỉnh là một bài toán tương tự, trong đó ta phải tìm các đường đi ngắn nhất đi từ đỉnh u đến đỉnh v, với mọi cặp đỉnh u, v thuộc V [14].

Một số thuật toán phổ biến được sử dụng để xử lý bài toán này có thể kể đến như thuật toán Dijkstra, thuật toán Bellman-Ford, thuật toán Floyd-Warshall, thuật toán A*,... Bài toán tìm đường đi ngắn nhất có nhiều ứng dụng thực tiễn, từ việc tối ưu hóa mạng lưới giao thông, tìm kiếm đường đi trong bản đồ, đến các vấn đề trong viễn thông và mạng máy tính,...

1.2. Mục tiêu nghiên cứu

Đề tài sử dụng thuật toán Dijkstra để tìm đường đi ngắn nhất từ đó tìm ra tuyến đường tốt nhất với khoảng cách hoặc chi phí thấp nhất. Với kết quả đạt được, nhóm sẽ đánh giá trên một số tiêu chí và từ đó đưa ra nhận định về kết quả đạt được.

1.3. Phương pháp nghiên cứu

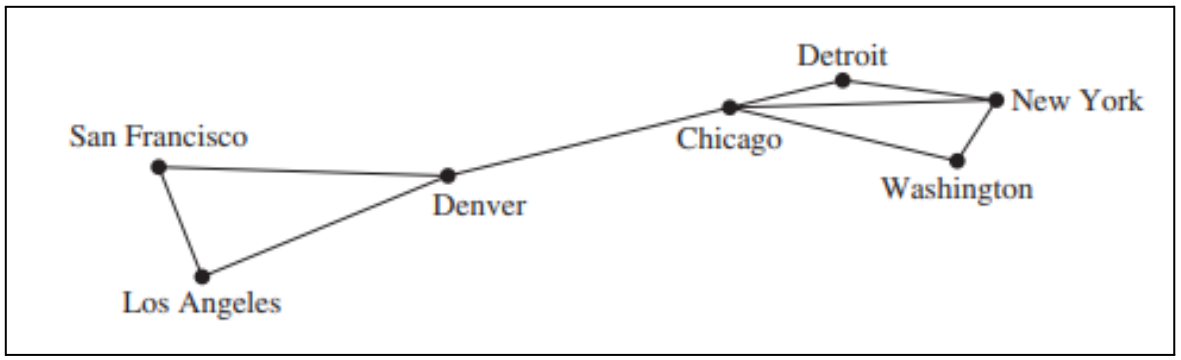
Thuật toán Dijkstra được triển khai để xác định đường đi ngắn nhất giữa các đỉnh trong đồ thị vô hướng từ tập tin Graph.csv. Đây là một tập tin chứa các thông tin về đỉnh và cạnh của một đồ thị, nhóm sẽ nghiên cứu thuật toán Dijkstra được biểu diễn thông qua đồ thị này.

CHƯƠNG II: CƠ SỞ LÝ THUYẾT

2.1. Đồ thị

2.1.1. Sơ lược về đồ thị (Khái niệm đồ thị)

Đồ thị, hay diễn giải theo một cách khác, đó là một cách để biểu diễn mối quan hệ giữa các đối tượng. Các đối tượng đó được thể hiện bằng các *đỉnh* trong đồ thị, và mối quan hệ giữa chúng được biểu diễn bằng cấu trúc được biết với cái tên *cạnh*. Ta đã bắt gặp ứng dụng của đồ thị trong nhiều lĩnh vực, đó là vận chuyển, đó là mạng máy tính, đó là kỹ thuật điện, hoặc đó cũng có thể là địa lý,... [1]. Một ví dụ về đồ thị được thể hiện qua hình dưới, trong đó đỉnh là các thành phố, đó là {San Francisco, Los Angeles, Denver, Chicago, Detroit, New York, Washington}. Các cạnh giữa các đỉnh này thể hiện đường đi giữa các thành phố.



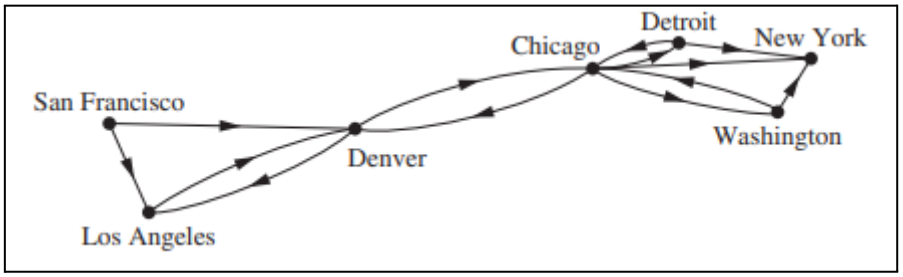
Hình 1: Ví dụ về đồ thị - Nguồn: [2]

Một cách tổng quát, gọi V = $\{v_1, v_2,..., v_n\}$ là một tập hữu hạn có n phần tử khác rỗng và gọi $E \subset \{\{a, b\}: a, b \in V\}$. Một đồ thị G với các đỉnh là phần tử của V và cạnh là phần tử của E được ký hiệu là $G = (V, E)$. Nếu không có sự nhầm lẫn về cạnh của đồ thị, cạnh {a,b} có thể được viết thành ab.

2.1.2. Các loại đồ thị

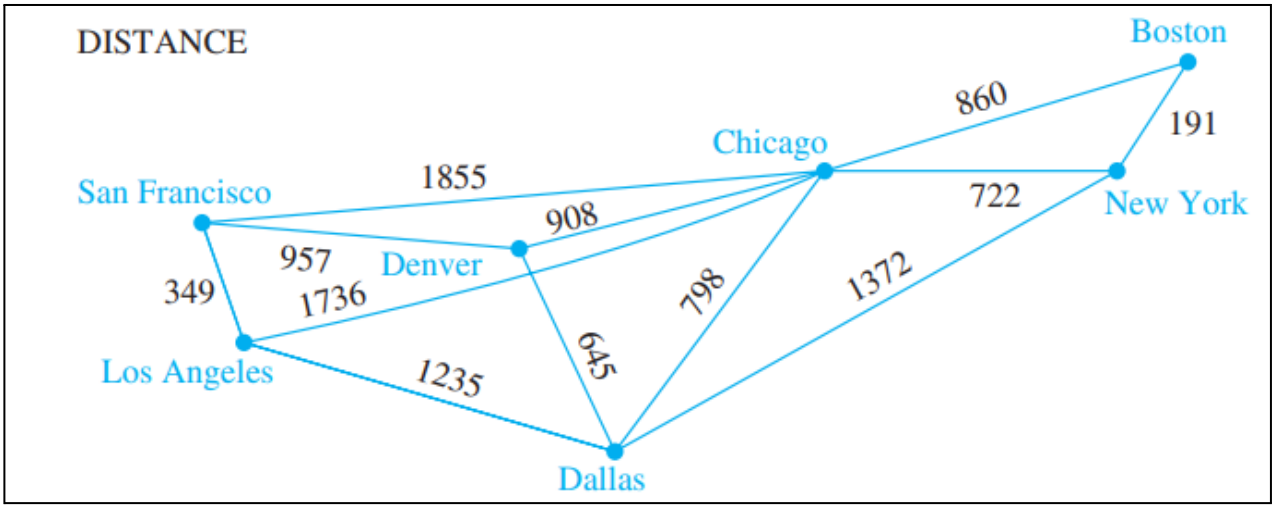
Một cạnh có thể có hướng hoặc không có hướng. Từ đó tạo nên các loại đồ thị như đồ thị vô hướng, đồ thị hữu hướng. Đồ thị vô hướng là đồ thị mà cạnh nối giữa 2 đỉnh được xem như là cặp đỉnh không sắp thứ tự, hay nói cách khác, một cạnh {a,b} được hiểu tương tự như {b,a}. Ta có thể thấy, hình 2.1 chính là ví dụ về đồ thị vô hướng. Đồ thị hữu hướng được biểu diễn như hình 2.2 là đồ thị mà trong đó cạnh nối giữa 2 đỉnh được xem như là cặp đỉnh có thứ tự, nghĩa là, cạnh {a,b} được quy định chặt chẽ đỉnh a sẽ ở trước đỉnh b,

và cạnh {a,b} sẽ được phát biểu "bắt đầu từ a và kết thúc ở b".



Hình 2: Ví dụ về đồ thị hữu hướng - Nguồn: [2]

Mối quan tâm mà nhóm đặt trọng tâm sẽ là đồ thị vô hướng. Cụ thể hơn, đó là đồ thị vô hướng có trọng số. Một đồ thị được gọi là có trọng số khi có một số được gán cho mỗi cạnh của đồ thị.



Hình 3: Đồ thị vô hướng có trọng số - Nguồn: [2]

Mỗi cạnh ở hình trên được gán một trọng số, trọng số này thể hiện khoảng cách giữa hai đỉnh của đồ thị. Trọng số này có thể được sử dụng để tìm đường đi ngắn nhất giữa 2 đỉnh. Đó cũng chính là vấn đề mà nhóm sẽ giải quyết trong bài báo cáo này.

2.2. Tìm kiếm trên đồ thị (Graph search)

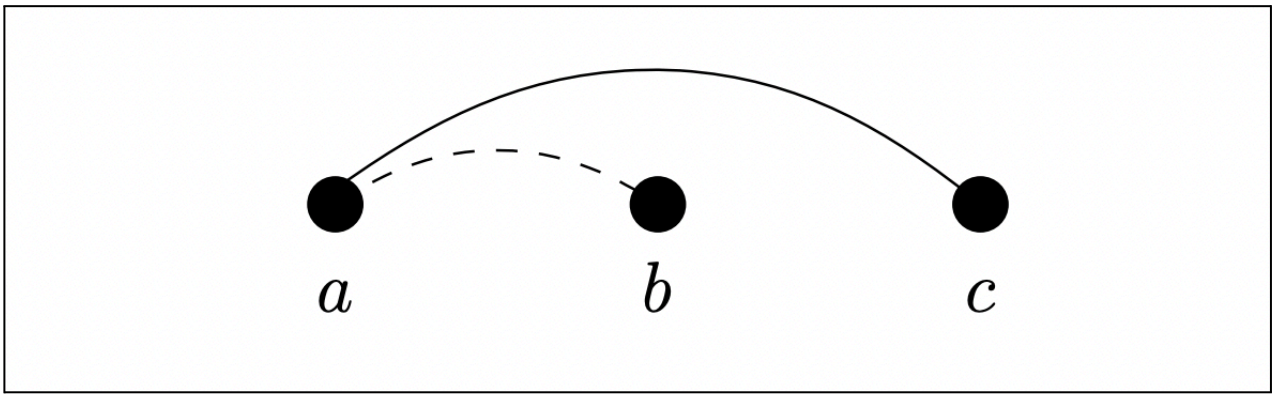
Trong số những thuật toán về đồ thị (Graph Algorithms) thì thuật toán tìm kiếm trên đồ thị (Graph Searching) là loại thuật toán đơn giản nhất. Tuy đây là dạng thuật toán đơn giản và được ứng dụng nhiều nhưng đến thời điểm hiện tại rất ít nghiên cứu chỉ ra nguyên lý rõ ràng về thứ tự duyệt đỉnh [4]. Theo [5], cách hoạt động chung của các thuật toán tìm kiếm trên đồ thị là bắt đầu từ một đỉnh gốc đã được chọn và tiến hành duyệt qua lần lượt các đỉnh và cạnh gần đó. Có rất nhiều thuật toán về tìm kiếm trên đồ thị, nhưng theo [6], các

thuật toán này có thể được phân loại thành hai nhóm chính: Tìm kiếm cơ bản (uninformed search) và tìm kiếm với tri thức bổ sung (informed search).

2.2.1. Tìm kiếm cơ bản (Uninformed Search)

Theo [7], các thuật toán thuộc nhóm tìm kiếm cơ bản (Uninformed Search) còn được gọi là tìm kiếm mù, hoạt động theo nguyên tắc chỉ dựa trên định nghĩa vấn đề mà không có thông tin bổ sung về trạng thái hay không gian tìm kiếm. Các thuật toán này hoạt động bằng cách thử từng bước, tạo ra các trạng thái kế tiếp thông qua việc duyệt qua cây để phân biệt trạng thái mục tiêu và phi mục tiêu, mở rộng các nút tuần tự cho đến khi tìm ra giải pháp và dừng thuật toán lại. Điều này đồng nghĩa với việc chúng khám phá không gian tìm kiếm một cách mù quáng, kiểm tra mọi khả năng mà không ưu tiên hướng đi nào, bao gồm cả chi phí, thời gian hay một mục tiêu cụ thể nào đó.

Theo như ví dụ của Corneil và đồng nghiệp trong một bài nghiên cứu đã thể hiện rõ vấn đề của các thuật toán tìm kiếm cơ bản như sau [4]: Trong một thuật toán tìm kiếm cơ bản (Uninformed Search), thứ tự chọn đỉnh có thể không dựa vào bất kỳ thông tin mang ý nghĩa nào. Mặc dù có cạnh nối giữa a và c, đỉnh b vẫn được chọn trước c chỉ dựa trên cách thuật toán thực hiện việc mở rộng đỉnh mà không ưu tiên cụ thể về chi phí, khoảng cách, hay mục tiêu. Điều này thể hiện cách tiếp cận mù quáng của thuật toán khi lần lượt khám phá các đỉnh mà không có quy tắc hướng dẫn rõ ràng.



Hình 4: Ví dụ mô tả vấn đề "tìm kiếm mù" của các thuật toán Tìm kiếm cơ bản - Nguồn: [4]

Tất cả các thuật toán trong nhóm này đều được xây dựng dựa trên chiến lược không gian tìm kiếm và dưới đây là một số thuật toán phổ biến và được sử dụng nhiều trong nhóm tìm kiếm cơ bản (Uninformed Search) [8]:

- Tìm kiếm theo Chiều Rộng (Breadth-First Search - BFS): BFS duyệt các đỉnh theo từng mức, kiểm tra hết các đỉnh ở một cấp trước khi chuyển xuống cấp tiếp theo. Điều này giúp tìm giải pháp nhanh nhất nếu tồn tại.

- Tìm kiếm theo Chiều Sâu (Depth-First Search - DFS): DFS ưu tiên khám phá chiều sâu của một nhánh trước, quay lại khi cần để thăm các nhánh khác. Phương pháp này tiết kiệm bộ nhớ nhưng có thể mắc kẹt nếu có vòng lặp vô hạn.

- Tìm kiếm Chiều Sâu Giới Hạn (Depth-Limited Search): Phiên bản DFS có giới hạn độ sâu, tránh vòng lặp vô hạn nhưng có thể bỏ lỡ giải pháp nếu nằm ngoài giới hạn.

- Tìm kiếm Chiều Sâu Lặp Lại (Iterative Deepening DFS): Kết hợp BFS và DFS, lặp DFS với độ sâu tăng dần để đảm bảo tìm được giải pháp ở mức thấp nhất nếu có.

2.2.2. Tìm kiếm với tri thức bổ sung (Informed Search)

Theo [6], Informed Search, hay còn gọi là Heuristic Search, là loại thuật toán tìm kiếm dùng thông tin bổ sung hoặc hàm heuristic để đánh giá các lựa chọn, giúp thuật toán ưu tiên những trạng thái gần với mục tiêu hơn. Đây là điểm khác biệt so với Uninformed Search, loại thuật toán thực hiện tìm kiếm mà không có chỉ dẫn cụ thể nào. Với Informed Search, các thuật toán như A* và Dijkstra có thể tìm giải pháp nhanh và hiệu quả hơn bằng cách tận dụng thông tin về vấn đề để rút ngắn thời gian tìm kiếm.

Một số ứng dụng của các loại thuật toán trong nhóm Informed Search [9]:

- Trí tuệ nhân tạo và robot: Informed Search giúp AI và robot tìm đường đi ngắn nhất, tối ưu hóa hoạt động di chuyển trong môi trường phức tạp. Ví dụ: Trong các nhà kho tích hợp công nghệ hiện đại, các robot được lập trình với thuật toán A* sẽ được sử dụng để lập kế hoạch lộ trình tối ưu cho việc lấy và vận chuyển hàng hóa, giúp giảm thiểu thời gian di chuyển và tránh va chạm với chướng ngại vật. Thuật toán này tính toán các lộ trình hiệu quả nhất, đảm bảo robot hoạt động một cách an toàn và hiệu quả trong môi trường làm việc phức tạp.

- Tối ưu hóa tuyến đường trong hệ thống GPS: Hệ thống định vị GPS chủ yếu dựa vào Informed Search để xác định tuyến đường nhanh nhất đến đích. Các thuật toán này tính đến khoảng cách, tình trạng giao thông và điều kiện đường xá để cung cấp

lộ trình hiệu quả nhất. Ví dụ: Trong các ứng dụng như Google Maps, thuật toán A* và các thuật toán tương tự được sử dụng để tính toán lộ trình lái xe ngắn nhất, dựa trên dữ liệu giao thông theo thời gian thực để cập nhật lộ trình khi điều kiện thay đổi.

- Giải quyết vấn đề trong trò chơi: Informed Search được sử dụng rộng rãi trong video game để điều khiển chuyển động của nhân vật và giải quyết các câu đố phức tạp. Những thuật toán này giúp các nhân vật không phải người chơi (NPC) đưa ra quyết định thông minh về việc di chuyển trong môi trường game. Ví dụ: Trong các trò chơi như Pacman, thuật toán A* được sử dụng để hướng dẫn chuyển động của các bóng ma, giúp chúng đuổi theo người chơi qua mê cung một cách hiệu quả nhất.

2.3. Giải thuật Dijkstra

2.3.1. Ý tưởng

Thuật toán Dijkstra được đặt tên theo nhà khoa học máy tính người Hà Lan - Edsger W. Dijkstra và được công bố chính thức vào năm 1959. Thuật toán Dijkstra giải quyết bài toán tìm đường đi ngắn nhất từ một đỉnh nguồn đến tất cả các đỉnh còn lại trong một đồ thị có hướng với trọng số không âm trên các cạnh [11].

Ý tưởng của Thuật toán dựa trên việc duyệt qua các đỉnh đồ thị theo thứ tự khoảng cách từ ngắn nhất đến xa nhất tính từ đỉnh nguồn. Trong suốt quá trình, thuật toán sẽ theo dõi và cập nhật liên tục khoảng cách ngắn nhất từ đỉnh nguồn đến từng đỉnh được duyệt trong đồ thị. Mỗi khi phát hiện một con đường ngắn hơn đến một đỉnh bất kỳ, thuật toán sẽ cập nhật giá trị khoảng cách tương ứng từ đỉnh nguồn đến đỉnh đó [12].

Một đỉnh được gán nhãn là "đã thăm" khi thuật toán xác định được đường đi ngắn nhất từ đỉnh nguồn đến đỉnh đó. Các đỉnh được gán nhãn sẽ được thêm vào tập hợp các đỉnh đã được tối ưu hóa đường đi. Vòng lặp này sẽ tiếp diễn cho đến khi toàn bộ các đỉnh được đều được gán nhãn. Từ đó, chúng ta có một đường đi kết nối đỉnh nguồn với tất cả các đỉnh khác theo đường đi ngắn nhất có thể để đạt đến mỗi đỉnh [12]. Chi tiết thuật toán có thể mô tả như sau:

Ta có bài toán dạng tổng quát: Cho đồ thị trọng số G = (V, E) yêu cầu tìm đường đi ngắn nhất từ đỉnh nguồn ký hiệu là s (s $\in$ V) đến đỉnh đích ký hiệu là e (e $\in$ V).

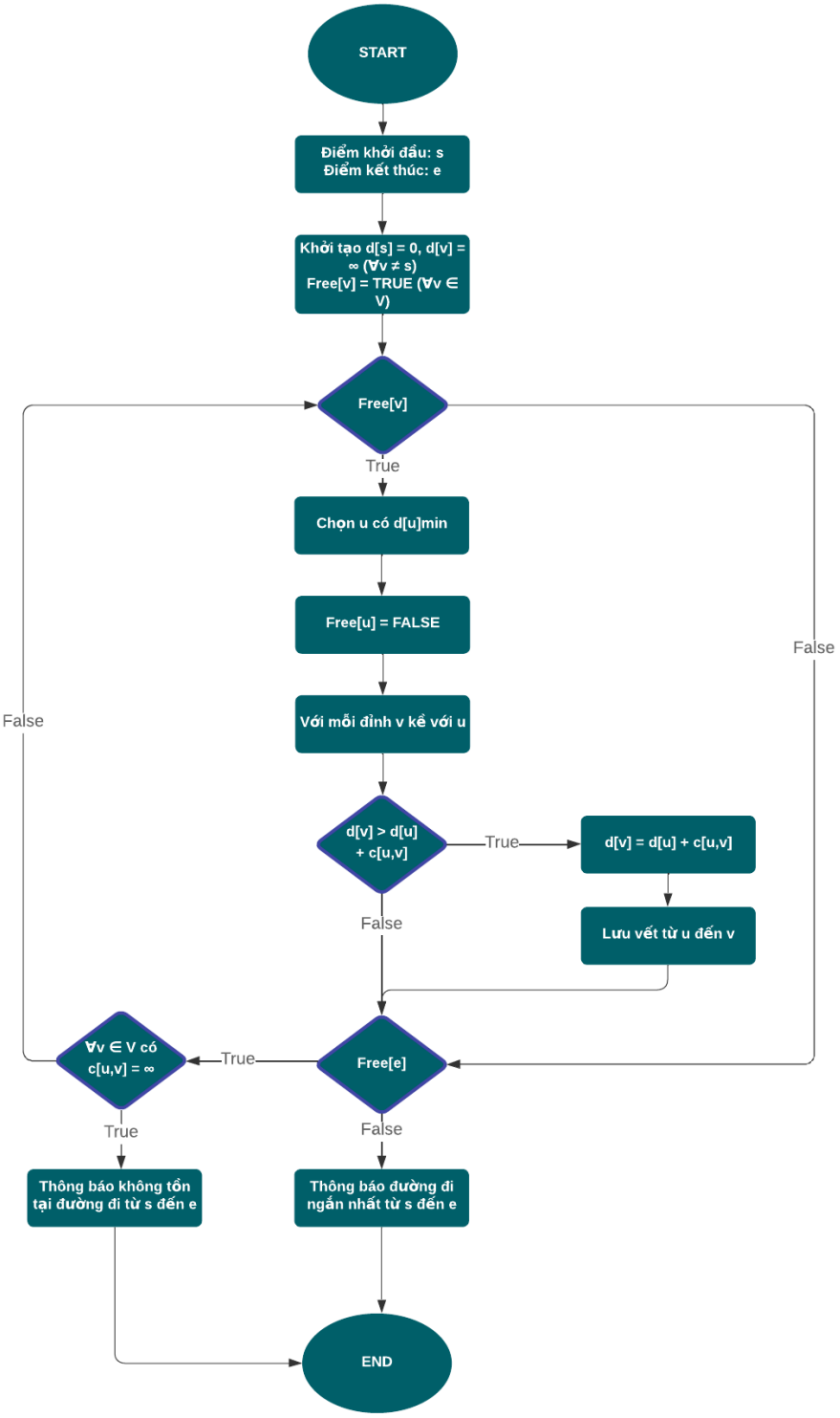
Bước 1: Ta khởi tạo một đồ thị với đỉnh v $\in$ V, gọi nhãn d[v] là độ dài đường đi ngắn nhất từ s tới v, với d[v] là một mảng (array) một chiều hay một list dùng để lưu trữ, trong đó mỗi phần tử của mảng hoặc list tương ứng với một đỉnh trong đồ thị. Chỉ số của phần tử là đỉnh v và giá trị của phần tử đó là d[v]. Nếu không tồn tại đường đi giữa các đỉnh ta gán nhãn khoảng cách bằng +∞.Ban đầu d[v] được khởi gán như sau d[s] = 0 và d[v] = +∞ với $\forall$v $\neq$ s. 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. Để 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 [3].

Bước 2: Thực hiện lặp với hai thao tác. Đầu tiên, ta cố định nhãn bằng các chọn trong các đỉnh có nhãn tự do hay Free[v] = True, lấy ra đỉnh u là đỉnh có d[u] nhỏ nhất sau đó cố định nhãn của đỉnh u. Tiếp theo, ta sửa nhãn bằng cách dùng đỉnh u để xét tất cả các đỉnh v và sửa lại các d[v] theo công thức: d[v] := min (d[v], d[u]+c[u,v]) với c[u,v] là trọng số trên cạnh nối đỉnh u đến đỉnh v. Ta sẽ tiếp tục thực hiện 2 thao tác này đến khi đỉnh đích được cố định nhãn (tìm được đường đi ngắn nhất từ đỉnh nguồn đến đỉnh đích) hoặc tại thao tác cố định nhãn ta nhận thấy tất cả các đỉnh tự do đều có nhãn là +∞ (không tồn tại đường đi) [3].

Ta có thể đặt giả thuyết rằng khi cố định nhãn sẽ tồn tại việc d[u] còn có thể tối ưu thêm được nữa. Khi đó, tất yếu phải có một đỉnh x mang nhãn tự do sao cho d[u] > d[x] + c[x,u]. Do điều kiện bài toán là trọng số không âm thế nên d[u] > d[x] điều này mâu thuẫn với việc chọn d[u] là nhỏ nhất. Tất nhiên trong lần lặp đầu tiên thì s là đỉnh được cố định nhãn do ta mặc định d[s] = 0 [3].

Bước 3: Kết hợp với việc lưu vết đường đi trên từng bước sửa nhãn, ta đưa ra thông báo đường đi ngắn nhất tìm được hoặc cho biết không tồn tại đường đi (d[e] = +∞) [3].

2.3.2. Flowchart



Hình 5: Flowchart đối với giải thuật Dijkstra

Bước 1: Ta bắt đầu thuật toán và cho biết đỉnh nguồn và đỉnh đích cần tìm.

Bước 2: Ta khởi tạo đồ thị và gán khoảng cách từ nguồn đến chính nó d[s] = 0, gán khoảng

cách từ nguồn đến các đỉnh v là $d[v] = \infty$ ($\forall v \neq s$), đánh dấu tất cả các đỉnh v trong đồ thị là chưa được thăm đồng nghĩa Free [v] = True ($\forall v \in V$).

Bước 3: Ta kiểm tra tất cả các đỉnh v đã được thăm hay chưa. Nếu Free[v] = FALSE với mọi v ta chuyển sang bước 8. Nếu Free [v] = TRUE tức là có đỉnh chưa thăm ta chuyển qua bước 4.

Bước 4: Trong các đỉnh v ta chọn lấy một đỉnh u sao cho d[u]min (với những đỉnh kế đỉnh khởi đầu ta có thể sử dụng trọng số giữa u và điểm khởi đầu thay cho d[u] khi chọn u).

Bước 5: Đánh dấu đỉnh u là đã thăm, Free[u] = FALSE.

Bước 6: Ta xét lần lượt từng đỉnh v kề với đỉnh u và kiểm tra xem d[v] > d[u] + c[u,v] hay không? Nếu có ta cập nhật khoảng cách d[v] = d[u] + c[u,v] và lưu vết đường đi từ u đến v. Nếu không ta không cập nhật mà chuyển sang đỉnh kề tiếp theo.

Bước 7: Kiểm tra xem điểm đích đã được thăm hay chưa? Nếu chưa được thăm ta xét tất cả các đỉnh còn lại còn lối đi hay không? Nếu có quay trở lại bước 3 và lặp tiếp. Nếu không chuyển đến bước 8. Nếu điểm đích đã được thăm ta thông báo "Đường đi ngắn nhất từ s đến e".

Bước 8: Thông báo "Không tồn tại đường đi từ s đến e" và kết thúc thuật toán.

2.4. Đánh giá

Để đưa ra đánh giá về giải thuật Dijkstra, nhóm sẽ tiến hành đánh giá hai giải thuật dựa trên các tiêu chí sau.

Một, độ phức tạp của thuật toán. Dijkstra sẽ được nhóm tiến hành đưa ra đánh giá và nhận định thông qua độ phức tạp về thời gian cũng như bộ nhớ. Công cụ mà nhóm sẽ sử dụng đó là Big O Notation.

Hai, tính hoàn chỉnh và tính tối ưu của Dijkstra. Tính hoàn chỉnh sẽ cho ta biết liệu nếu thực sự tồn tại một lời giải thì có tìm được lời giải đó hay không và tính tối ưu sẽ cho ta

biết liệu có đảm bảo rằng sẽ tìm được lời giải với chi phí thấp nhất hay không.

CHƯƠNG III: KẾT QUẢ THỰC NGHIỆM

3.1. Phát biểu bài toán

Giả sử ta có đơn đồ thị vô hướng G được lưu trong tập tin Graph.csv dưới dạng biểu diễn danh sách cạnh liên thuộc, trong đó mỗi dòng gồm 3 thông tin như sau:

(v_from, v_to, weight)

Với v_from là đỉnh bắt đầu của cạnh trong đồ thị, v_to là đỉnh kết thúc của cạnh trong đồ thị, weight là trọng số của cạnh nối giữa hai đỉnh v_from và v_to.

Bài toán đặt ra ở đây, đó là áp dụng thuật toán Dijkstra để tìm đường đi ngắn nhất giữa 2 đỉnh bất kỳ.

3.2. Mô hình hóa

Để tìm lời giải cho bài toán tìm đường đi ngắn nhất thông qua thuật toán Dijkstra nhóm tiến hành mô hình hóa bài toán như sau: Toàn bộ bài toán xoay quanh một lớp chính có tên là Graph.

Lớp	Mô tả
Graph	Đại diện cho đồ thị, là lớp chứa thông tin về các đỉnh, các cạnh và trọng số của chúng. Cung cấp các phương thức để thao tác với đồ thị như thêm cạnh, đếm số lượng cạnh và đỉnh, tìm đường đi ngắn nhất và hiển thị đồ thị.

Bảng 1: Lớp chính trong mô hình bài toán tìm đường đi ngắn nhất bằng giải thuật Dijkstra

Trong lớp Graph, ta có các hàm để thực hiện các chức năng như sau:

Hàm	Mô tả
__init__	Hàm khởi tạo của lớp Graph. Hàm cho phép khởi tạo các thuộc tính của đồ thị bao gồm tên (name), danh sách lưu trữ các

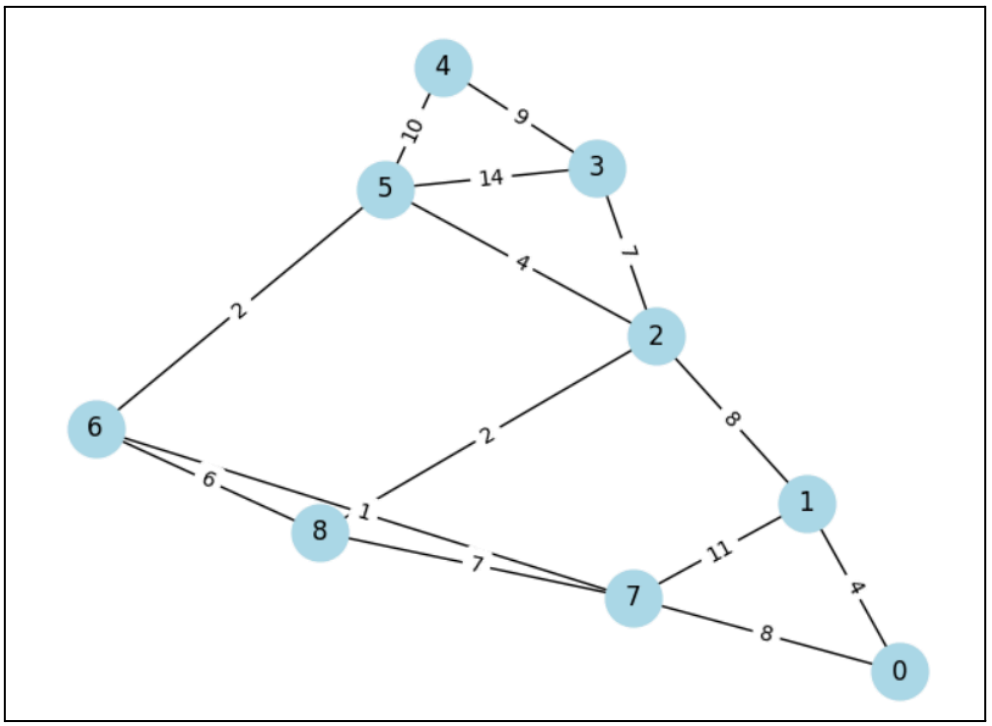
	cạnh (_list_luu), số lượng cạnh của đồ thị (_dem_canh), và cách bố trí đồ thị để hiển thị (_layout).
add_edge	Được sử dụng để thêm một cạnh vào đồ thị. Khi thêm một cạnh sẽ nhận vào 3 tham số là start, end và weight. Hàm này đảm bảo rằng cả hai nút start và end đều có trong _list_luu, sau đó thêm cạnh vào danh sách các cạnh của cả hai nút. Cộng số lượng cạnh lên 1 (_dem_canh +1)
tai_dothi	Tải dữ liệu đồ thị từ file CSV
_set_layout	Thiết lập cách bố trí các điểm trên đồ thị để thuận lợi cho việc hiển thị
dijkstra	Hàm này triển khai thuật toán Dijkstra để xác định đường đi ngắn nhất giữa hai nút trong đồ thị. Thuật toán hoạt động bằng cách duy trì một tập hợp các nút với khoảng cách ngắn nhất tạm thời từ nút nguồn. Ban đầu, khoảng cách đến tất cả các nút được đặt là vô cùng, ngoại trừ nút nguồn có khoảng cách là 0. Một hàng đợi ưu tiên được sử dụng để lưu trữ các nút và khoảng cách tương ứng của chúng, cho phép thuật toán hiệu quả chọn nút có khoảng cách ngắn nhất để khám phá tiếp theo. Trong quá trình thực thi, thuật toán cập nhật khoảng cách đến các nút lân cận và lưu trữ nút trước đó trên đường đi ngắn nhất. Kết quả cuối cùng là khoảng cách

	ngắn nhất từ nút nguồn đến nút đích và đường đi tương ứng, được biểu diễn dưới dạng một danh sách các nút.
display	Hàm này giúp thể hiện đường đi trên đồ thị khi hiển thị thông qua việc nhận một danh sách các nút tạo thành đường đi và tô màu đường đi đó.
get_shortest_path	Hàm này trả về khoảng cách ngắn nhất và danh sách các nút trên đường đi ngắn nhất thông qua việc gọi hàm dijkstra để tính toán đường đi ngắn nhất từ nút start đến nút end sau đó gọi hàm display để thể hiện đường đi trên đồ thị.
get_list_luu	Hàm giúp trả về danh sách lưu trữ các cạnh của đồ thị (_list_luu)
get_vertex_count	Hàm giúp trả về số lượng đỉnh trong đồ thị
get_dem_canh	Hàm giúp trả về số lượng cạnh trong đồ thị (_dem_canh)

Bảng 2: Các hàm trong mô hình bài toán tìm đường đi ngắn nhất bằng giải thuật Dijkstra

3.3. Kết quả đạt được

Từ bài toán áp dụng thuật toán Dijkstra để tìm đường đi ngắn nhất giữa 2 đỉnh trong đồ thị. Nhóm xét trường hợp tìm đường đi ngắn nhất từ đỉnh 0 đến đỉnh 4 trong đồ thị xây dựng từ Graph.csv.



Hình 6: Đồ thị xây dựng từ Graph.csv

Đầu tiên, ta sẽ input đồ thị với các đỉnh từ 0 đến 8 cùng với trọng số trên các cạnh nối các đỉnh. Tiếp theo đó sẽ khởi tạo khoảng cách giữa các đỉnh tới đỉnh ban đầu là vô cùng (d[v]=∞) ngoại trừ đỉnh nguồn (d[0]=0). Ta tiếp tục thiết lập một danh sách các đỉnh chưa được duyệt qua Q= {1,2,3,4,5,6,7,8}.

Sau khi thiết lập xong các bước ban đầu thì nhóm sẽ thực hiện các bước:

Đầu tiên, sẽ xuất phát từ đỉnh nguồn (đỉnh 0) ta sẽ cập nhật lại khoảng cách của đỉnh 0 với các đỉnh kề là 1 và 7 lần lượt là d[1]=min(∞,d[0]+4)=4 và d[7]=min(∞,d[0]+8)=8. Vì d[1] có khoảng cách nhỏ hơn nên ta chọn đỉnh 1 để xét tiếp.

Thứ hai, ta xét các đỉnh kề với đỉnh 1. Ta có đỉnh 2 và đỉnh 7 có khoảng cách với đỉnh 0 lần lượt là d[2]=min(∞,d[1]+8)=12 và d[7]=min(8,d[1]+11)=8 (không thay đổi). Ta chọn đỉnh 7 để xét tiếp vì khoảng cách từ đỉnh 0 đến đỉnh 7 là ngắn hơn.

Tại bước thứ ba, khi xét đỉnh 7, ta cập nhật khoảng cách từ đỉnh 0 đến các đỉnh kề của 7 là đỉnh 6 và đỉnh 8. Cụ thể, khoảng cách đến đỉnh 6 được cập nhật thành d[6]=min(∞,d[7]+1)=9 và đến đỉnh 8 là d[8]=min(∞,d[7]+7)=15. Do đỉnh 6 có khoảng cách ngắn hơn, ta tiếp tục xét đỉnh 6.

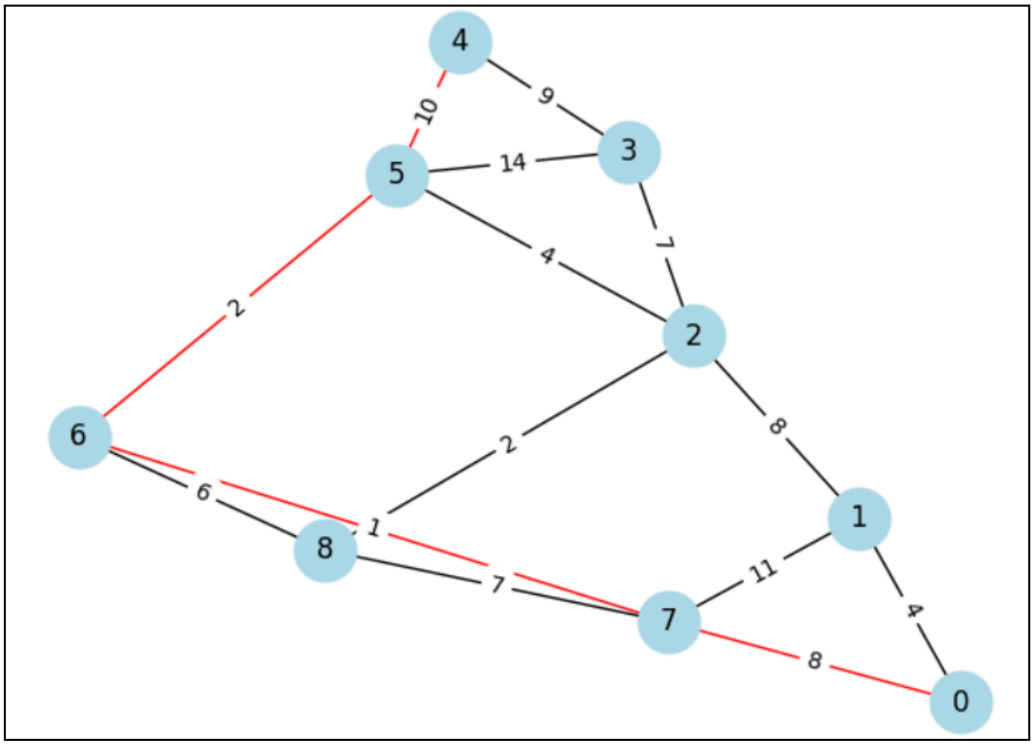
Tại bước thứ tư, từ đỉnh 6, ta cập nhật khoảng cách đến các đỉnh kề là đỉnh 5 và đỉnh 8.

Khoảng cách đến đỉnh 5 được tính là d[5]=min(∞,d[6]+2)=11, trong khi khoảng cách đến đỉnh 8 không thay đổi (d[8]=15). Tiếp tục, ta chọn đỉnh 5 để xét vì khoảng cách d[5]=11 là nhỏ nhất.

Trong bước thứ năm, từ đỉnh 5, các đỉnh kề là đỉnh 4 và đỉnh 3 được cập nhật. Ta tính được d[4]=min(∞,d[5]+10)=21 và d[3]=min(∞,d[5]+14)=25. Do khoảng cách nhỏ nhất hiện tại là đến đỉnh 2 (d[2]=12), ta chọn đỉnh 2 để xét tiếp.

Tại bước thứ sáu, từ đỉnh 2, ta cập nhật khoảng cách đến đỉnh 3: d[3]=min(25,d[2]+7)=19. Ta xét tiếp khoảng cách từ đỉnh 3 đến đỉnh 4: d[4]=min(21,d[3]+9)=21. Do không còn đỉnh nào trong tập Q có khoảng cách nhỏ hơn khoảng cách hiện tại đến đỉnh 4 (d[4]=21), ta lần lượt xét các đỉnh còn lại.

Cuối cùng, sau khi duyệt hết tất cả các đỉnh trong tập Q, kết quả mà giải thuật đưa ra cho ta từ đỉnh 0 đến đỉnh 4 là thông qua các đỉnh 0 → 7 → 6 → 5 → 4, với tổng trọng số là 21.



Hình 7: Kết quả của thuật toán Dijkstra khi tìm đường đi ngắn nhất giữa đỉnh 0 và 4

CHƯƠNG IV: ĐÁNH GIÁ

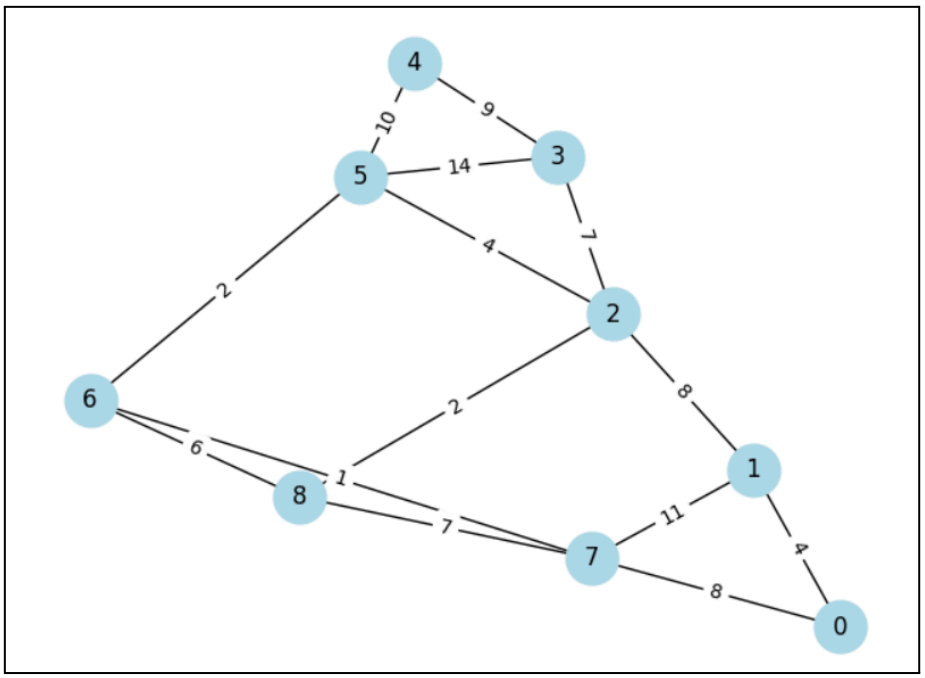
4.1. Độ phức tạp

Gọi V là số đỉnh của đồ thị, E là số cạnh của độ thị. Khi đó độ phức tạp về thời gian của giải thuật Dijkstra sẽ là O(V^2) nếu dùng cấu trúc mảng để lưu trữ, nhưng, nhóm không sử dụng cấu trúc mảng để lưu trữ mà thay vào đó nhóm sử dụng hàng đợi ưu tiên (Priority queue), khi thêm hoặc xóa một phần tử trong hàng đợi ưu tiên độ phức tạp sẽ là O(logV), sở dĩ có điều này là do hàng đợi ưu tiên cập nhật phần tử giống với cấu trúc cây nhị phân, trường hợp tệ nhất đó là phải duyệt tới mức sâu nhất của cây, hay nói cách khác đó là logV. Như vậy, ta có (V+E) phần tử, vậy độ phức tạp về thời gian sẽ là O((V+E)logV). Dijkstra sẽ lưu trữ tất cả các đỉnh được tạo ra, vì vậy, độ phức tạp về không gian sẽ là O(V).

4.2. Tính hoàn chỉnh và tính tối ưu

Xét về tính hoàn chỉnh của giải thuật, Dijkstra phụ thuộc vào loại đồ thị mà giải thuật được áp dụng để tìm đường đi. Nếu như đồ thị có trọng số âm, Dijkstra sẽ không hoàn chỉnh do sự xuất hiện của trọng số âm. Dijkstra sẽ không xét đến điểm đã đi qua, sau khi thăm một đỉnh, và nếu đỉnh đó có trọng số âm thì tổng khoảng cách sẽ bị giảm xuống tương ứng với trọng số âm của đỉnh đó làm cho quá trình tìm kiếm đường đi trở nên khó khăn và kém chính xác hơn bởi Dijkstra là giải thuật tham lam, nó luôn chọn đỉnh mà cho tổng khoảng cách nhỏ nhất từ đỉnh nguồn tới đỉnh hiện tại.

Tính hoàn chỉnh của Dijkstra còn phải xem xét đến đồ thị đó có phải là đồ thị liên thông hay không. Một đồ thị được coi là liên thông khi luôn tồn tại đường đi giữa hai đỉnh, và với đồ thị mà nhóm đã chọn như ở hình dưới, dễ thấy rằng tồn tại một chu trình sao cho đi qua tất cả đỉnh, đó là {0, 7, 8, 6, 5, 4, 3, 2, 1, 0}. Vì vậy, đây là đồ thị liên thông. Mà đồ thị liên thông luôn có đường đi giữa hai đỉnh, vậy đối với phạm vi bài toán áp dụng giải thuật Dijkstra để tìm đường đi giữa hai đỉnh, nhóm có thể nói rằng, Dijkstra hoàn chỉnh.



Hình 8: Đồ thị xây dựng từ Graph.csv

Dijkstra là thuật toán tham lam, nó sẽ luôn chọn đỉnh sao cho đạt được tối ưu cục bộ ở từng bước. Dijkstra sẽ không xét đến bất cứ thông tin nào khác và chỉ tìm kiếm dựa trên trọng số giữa các cạnh, điều này sẽ không đảm bảo tìm được lời giải tối ưu và do đó, Dijkstra không đạt được tính tối ưu.

CHƯƠNG V: KẾT LUẬN

5.1. Về Dijkstra

Sau khi tìm hiểu thuật toán Dijkstra cho bài toán tìm đường đi ngắn nhất giữa 2 đỉnh trong đồ thị, nhóm có một số nhận xét về thuật toán như sau,

Thông qua các bước logic và quy trình có thể đưa ra nhận xét khi tìm đường đi ngắn nhất từ đỉnh nguồn đến các đỉnh khác thì thuật toán đảm bảo được độ chính xác cao khi ở trong các điều kiện phù hợp (trọng số của các cạnh không âm, đồ thị liên thông). Với việc sử dụng hàng đợi ưu tiên, độ phức tạp về thời gian được giảm bớt từ $O(V^2)$ trở thành O((V+E)logV). Điều này cho phép thuật toán xử lý được các đồ thị lớn và thưa. Tuy nhiên, vẫn còn một số yếu điểm về dữ liệu đầu vào. Thuật toán chỉ xử lý được những đồ thị có trọng số dương, đối với những đồ thị có trọng số âm thuật toán sẽ không thể xử lý được hoặc đưa ra kết quả sai. Điều này làm giảm tính linh hoạt khi áp dụng vào các bài toán như đánh giá rủi ro tài chính hoặc tối ưu hóa trong các hệ thống chứa chi phí âm.

Thuật toán tốt trong việc tìm khoảng cách từ một đỉnh nguồn tuy nhiên khi tìm đường đi ngắn nhất giữa các cặp đỉnh thì việc lặp lại liên tục thuật toán Dijkstra có thể không tối ưu. Ta nên sử dụng các thuật toán khác ví dụ như Floyd-Warshall. Đầu vào của thuật toán yêu cầu đồ thị cần được định nghĩa rõ ràng (mô tả dưới dạng danh sách kề hay ma trận kề, trọng số bắt buộc phải dương). Nếu dữ liệu đầu vào không chuẩn xác có thể gây ra lỗi hoặc kết quả không chính xác. Với đồ thị không liên thông thuật toán chỉ có thể tìm đường đi ngắn nhất với những đỉnh trong cùng thành phần liên thông, nếu ngoài thành phần liên thông có thể gây ra lỗi.

5.2. Hướng phát triển

Đối với bài toán Dijkstra, hướng phát triển của bài toán là về cấu trúc dữ liệu. Trong nghiên cứu này thì nhóm tác giả đang sử dụng cấu trúc dữ liệu để lưu các đỉnh là PriorityQueue. Nhóm tác giả đề xuất hướng nghiên cứu trong tương lai chính là thử nghiệm cấu trúc Fibonacci Heap và self balancing tree để xem xét coi cấu trúc này sẽ giúp tối ưu được thuật toán Dijkstra không.

TÀI LIỆU THAM KHẢO

[1] Michael T. Goodrich, Roberto Tamassia, Michael H. Goldwasser (2013). Chapter 14: Graph Algorithms. *Data Structures & Algorithms in Python*, Wiley, page 620 - 686.

[2] Kenneth H. Rosen (2011). Chapter 10: Graphs. *Discrete Mathematics and Its Applications*, seventh edition, McGraw Hill, page 641 - 735.

[3] Lê Minh Hoàng (2002). Chương 4: Bài toán đường đi ngắn nhất. Giải thuật & lập trình, trang 238 - 241.

[4] Corneil, D. G., & Krueger, R. M. (2008). A unified view of graph searching. *SIAM Journal on Discrete Mathematics*, *22*(4), 1259-1276.

[5] Tarjan, R. (1972). Depth-first search and linear graph algorithms. *SIAM Journal on Computing, 1*(2), 146–160

[6] Algasi, K. A. (2017). *Uninformed and Informed Search Techniques in Artificial Intelligence* (Master's thesis, Eastern Mediterranean University (EMU)-Doğu Akdeniz Üniversitesi (DAÜ)).

[7] Kapi, A. Y., Sunar, M. S., & Zamri, M. N. (2020). A review on informed search algorithms for video games pathfinding. *International Journal*, *9*(3).

[8] Dục Đoàn Trình. (2022, March 25). *Thuật toán tìm kiếm Uninformed Search Algorithms*. w3seo. Retrieved October 29, 2024, from https://websitehcm.com/thuat-toan-tim-kiem-uninformed-search-algorithms/

[9] Applied AI. (2024, September 23). *Informed Search Algorithms in Artificial Intelligence*. Applied AI Course. Retrieved October 29, 2024, from https://www.appliedaicourse.com/blog/informed-search-algorithms-in-artificial-intelligence/

[10] Trần Quốc Chiến, Nguyễn Thanh Tuấn, ALGORITHMS OF THE PROBLEM OF FINDING THE SHORTEST PATHS FROM A SET OF NODES TO ANOTHER SET OF NODES **https://lrel.ued.udn.vn/vi/tu-sach-giang-vien/tin-hoc/nguyen-thanh-tuan-31.html?download=1&id=1**

[11] Robert Sedgewick, Kevin Wayne (2010). Algorithms fourth edition. Page 652

[12] Estefania Cassingena Navone. (2020). Dijkstra's Shortest Path Algorithm - A Detailed and Visual Introduction.

[13] Russell, S. J., Norvig, P., & Davis, E. (2010). *Artificial intelligence: A modern*

approach. Prentice Hall.

[14] GIẢI THUẬT TÌM ĐƯỜNG ĐI NGẮN NHẤT GIỮA HAI TẬP ĐỈNH. (n.d.). TT Học liệu & Công nghệ thông tin. Retrieved November 21, 2024, from https://lrel.ued.udn.vn/vi/tu-sach-giang-vien/tin-hoc/nguyen-thanh-tuan-31.html?download=1&id=1

PHỤ LỤC

1. Cách chạy code

Trước khi chạy thuật toán, người dùng nên tải các thư viện cần thiết, để xem các thư viện cần dùng cho thuật toán, hãy xem ở cell import các thư viện. Nếu người dùng chưa tải thư viện nào, hãy mở terminal lên và pip install <tên thư viện>.

Khi chạy code, vì là file .ipynb nên người dùng hãy bấm Run All để chạy hết các cell. Trong quá trình chạy, chương trình sẽ yêu cầu người dùng nhập đỉnh bắt đầu và đỉnh kết thúc. Khi nhập xong, chương trình sẽ thể hiện kết quả ra cho người dùng.

2. Link dự phòng

Trong trường hợp file .ipynb trong file .RAR không chạy được, người dùng hãy nhấp vào đây để đi tới thư mục chứa code. Tiến hành tải về và chạy thuật toán.

ĐÁNH GIÁ CÔNG VIỆC

1. Đặng Thị Thu Hiền

Công việc	Mức độ hoàn thành
Tìm hiểu về Dijkstra và bài toán tìm đường đi ngắn nhất	100%
Viết báo cáo: Chương I	100%
Slides dùng để trình chiếu: tìm template	100%
Slides dùng để trình chiếu: Chương I, Chương II (2.1, 2.2, 2.4), Chương IV, Chương V	100%
Test Code Dijkstra	100%

2. Bùi Tiến Hiếu

Công việc	Mức độ hoàn thành
Tìm hiểu về Dijkstra và bài toán tìm đường đi ngắn nhất	100%
Tìm tập dữ liệu Graph.csv	100%
Viết báo cáo: Chương II (2.3), Chương III (3.2 + 3.3), Chương V(5.1)	100%
Xây dựng class cho thuật toán Dijkstra và code giải thuật Dijkstra	100%
Visualisation cho giải thuật Dijkstra: biểu diễn không gian bài toán, lời giải	100%
Test Code Dijkstra	100%

3. Đỗ Thanh Hoa

Công việc	Mức độ hoàn thành
Tìm hiểu về Dijkstra và bài toán tìm đường đi ngắn nhất	100%
Viết báo cáo: Chương II (2.2)	100%
Visualisation cho giải thuật Dijkstra: tìm hiểu thư viện, cách thức thư viện đó biểu diễn.	100%
Slides dùng để trình chiếu: Chương II (2.3), Chương III	100%
Format lại báo cáo	100%
Test Code Dijkstra	100%

4. Nguyễn Huy Hoàng

Công việc	Mức độ hoàn thành
Tìm hiểu về Dijkstra và bài toán tìm đường đi ngắn nhất	100%
Viết báo cáo: Chương II (2.1, 2.4), Chương IV.	100%
Test Code Dijkstra	100%

5. Nguyễn Trương Hoàng

Công việc	Mức độ hoàn thành
Tìm hiểu về Dijkstra và bài toán tìm đường đi ngắn nhất	100%
Viết báo cáo: Chương III (3.1), Chương V (5.2)	100%

Test Code Dijsktra	100%