**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**ĐẠI HỌC SƯ PHẠM TP. HỒ CHÍ MINH**

**🙣 🙡**

**KHOA CÔNG NGHỆ THÔNG TIN**

**BÁO CÁO ĐỀ TÀI**

**MINH HỌA TRỰC QUAN THUẬT TOÁN PRIM VÀ DIJKSTRA BẰNG GIAO DIỆN ĐỒ HỌA**

**Học phần: 2311COMP170104 – Lý thuyết đồ thị và ứng dụng**

**Học kỳ III**

**Năm học 2022 – 2023**

**TP. Hồ Chí Minh 2023**

**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**ĐẠI HỌC SƯ PHẠM TP. HỒ CHÍ MINH**

**🙣 🙡**

**KHOA CÔNG NGHỆ THÔNG TIN**

**BÁO CÁO ĐỀ TÀI**

**MINH HỌA TRỰC QUAN THUẬT TOÁN PRIM VÀ DIJKSTRA BẰNG GIAO DIỆN ĐỒ HỌA**

**Học phần: 2311COMP170104 – Lý thuyết đồ thị và ứng dụng**

**Học kỳ III**

**Năm học 2022 – 2023**

**Giảng viên hướng dẫn:** Nguyễn Viết Hưng, Trần Quang Huy

**Thành viên nhóm**

**Họ và tên MSSV**

Đoàn Văn Nhân 48.01.104.099

Trần Đào Bá Thành 48.01.104.124

Ngô Đăng Khoa 48.01.104.072

La Thái Thụy 48.01.104.132

**TP. Hồ Chí Minh 2023**

**MỤC LỤC**

**LỜI CẢM ƠN 2**

**BẢNG PHÂN CÔNG 3**

**NỘI DUNG 4**

**1. Giới thiệu đề tài 4**

**2. Cơ sở lý thuyết 5**

* **2.1. Giới thiệu về thuật toán Prim 5**
* **2.2. Giới thiệu về thuật toán Dijkstra 6**

**3. Phương pháp thực hiện 8**

* **3.1. Công cụ sử dụng 8**
* **3.2. Các bước thực hiện 9**
* **3.3. Mô tả thuật toán của chương trình 9**

**4. Kết quả đạt được 10**

**5. Định hướng phát triển 10**

**TỔNG KẾT 10**

**TÀI LIỆU THAM KHẢO 10**

**LỜI CẢM ƠN**

Chúng em xin chân thành cảm ơn sự quan tâm và hỗ trợ của giảng viên bộ môn – thầy Nguyễn Viết Hưng, với sự chỉ dạy tận tình và tạo điều kiện thoải mái, vui vẻ cho chúng em trong suốt thời gian học tập những tiết học lý thuyết của bộ môn “Lý thuyết đồ thị và ứng dụng” vừa qua. Nhờ đó mà chúng em có thể vận dụng những kiến thức được giảng dạy vào việc nghiên cứu đề tài.

Nhóm chúng em cũng chân thành gửi lời cảm ơn đến thầy trợ giảng Trần Quang Huy đã luôn đồng hành và hỗ trợ chúng em trong những tiết học thực hành của bộ môn. Với sự chỉ dạy của thầy, chúng em cũng đúc kết được những kiến thức cụ thể hơn trong việc phân tích và xử lí các thuật toán, dựa vào đó để tiếp tục phát triển và hoàn thành đề tài.

Tuy đề tài được phát triển và hoàn thiện đáng kể nhưng vì khối lượng kiến thức của các thành viên trong nhóm vẫn còn có những mặt hạn chế nhất định, dẫn đến việc khó tránh khỏi những thiếu sót là điều tất yếu. Hi vọng với sự đánh giá, nhận xét của thầy và các bạn, cũng như là mọi người xung quanh sẽ là nền tảng kiến thức quý báu để chúng em có thể học hỏi thêm và vận dụng để từng bước xây dựng đề tài ngày càng hoàn thiện và hiệu quả.

**BẢNG PHÂN CÔNG**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| MSSV | Họ và tên | Trưởng nhóm | Chi tiết công việc | % Hoàn Thành Công Việc |
| 48.01.104.099 | Đoàn Văn Nhân | X | Code chính phần thuật toán Dijkstra, hỗ trợ thiết kế báo cáo | 95% |
| 48.01.104.124 | Trần Đào Bá Thành |  | Code chính phần thuật toán Prim, hỗ trợ thiết kế powerpoint | 95% |
| 48.01.104.072 | Ngô Đăng Khoa |  | Code hỗ trợ phần thuật toán Dijkstra, thiết kế powerpoint | 95% |
| 48.01.104.132 | La Thái Thụy |  | Code hỗ trợ phần thuật toán Prim, thiết kế báo cáo | 95% |

**NỘI DUNG**

**1. GIỚI THIỆU ĐỀ TÀI**

Toán học từ lâu đã trở thành một phần không thể thiếu trong cuộc sống của loài người, không chỉ đơn thuần là hệ thống số học và phép tính, toán học còn là một dạng ngôn ngữ, một cách tiếp cận đối với sự logic và mô hình hóa thế giới. Đây được xem như là công cụ để con người giải quyết những vấn đề phức tạp và là nguồn động lực thúc đẩy con người không ngừng phát triển tư duy của bản thân.

Việc ứng dụng toán học vào trong cuộc sống thực tế luôn rất đa dạng và cần thiết bởi sự phát triển bùng nổ của xã hội hiện đại, kèm theo xu hướng tạm gọi là “ tin học hóa toàn thể “, khi công nghệ thông tin đã và đang xâm nhập vào nhiều lĩnh vực đời sống như: y tế, khoa học, đời sống,…Nhưng để có thể ứng dụng một cách hiệu quả, hợp lí đòi hỏi con người phải hiểu rõ và nắm được các bản chất của thuật toán cần thiết, trong số đó bao gồm các thuật toán xuất hiện trong lí thuyết đồ thị.

Nhìn chung, các thuật toán xuất hiện trong lí thuyết đồ thị thường không dễ để tiếp cận và làm quen đối với người học, nhất là những sinh viên mới bắt đầu làm quen với chúng vì độ phức tạp, khó hiểu và khó hình dung của chúng. Không chỉ vậy, việc các giảng viên dạy học và hỗ trợ truyền đạt ý tưởng, giúp người học hiểu đường nội dung và cách thuật toán hoạt động đôi khi cũng rất khó. Nắm bắt được những vấn đề trên, nhóm chúng em đã quyết định tạo ra một giao diện đồ họa để hỗ trợ người học cũng như người giảng dạy có cái nhìn trực quan và dễ hiểu hơn về cách mà các thuật toán hoạt động, cụ thể ở đây là hai thuật toán Prim và Dijkstra, từ đó dễ dàng tiếp cận và học hỏi, vận dụng sao cho hợp lí.

**2. CƠ SỞ LÝ THUYẾT**

**2.1. Giới thiệu về thuật toán Prim**

Từ lâu, bài toán tối ưu luôn là vấn đề muôn thuở của các ngành nghề nói chung và ngành công nghệ thông tin nói riêng vì tính đa dạng và khả năng ứng dụng rộng rãi của nó. Có thể kể đến như việc quy hoạch nhân sự, xây dựng mạng lưới internet, xây dựng đường sắt đều,… sao cho hiệu quả và ít tốn kém, hay nói dễ hiểu là sao cho tối ưu nhất. Vì vậy trên thế giới cũng xuất hiện rất nhiều thuật toán để đáp ứng việc giải quyết các vấn đề trên, một trong số đó có thể kể đến thuật toán Prim.

Thuật toán Prim được nhà toán học người Séc Vojtěch Jarník tìm ra vào năm 1930 và sau đó bởi nhà nghiên cứu khoa học máy tính Robert C. Prim năm 1957, thông qua bài báo “ Shortest Connection Networks and Some Generalizations “ được xuất bản trên tạp chí Bell System Technical Journal. Sau đó vào năm 1959, nó được tìm ra một cách độc lập bởi Edsger Dijkstra, vì vậy nó còn được gọi là thuật toán DJP, thuật toán Jarník hay thuật toán Prim–Jarník.

Thuật toán Prim thực hiện việc chọn lựa các cạnh của đồ thị một cách tận dụng chi phí nhỏ nhất để kết nối tất cả các đỉnh mà không tạo thành chu trình. Phương pháp này giúp tối ưu hóa chi phí trong việc xây dựng mạng lưới kết nối, đồng thời có ứng dụng rộng rãi trong các lĩnh vực lập trình, quản lí,… biến thuật toán Prim trở thành một công cụ vô cùng quan trọng không chỉ trong toán học mà còn trong cả đời sống con người.

**Mô tả thuật toán:**

Trong công nghệ thông tin, thuật toán Prim là một thuật toán tham lam để tìm cây bao trùm nhỏ nhất của một đồ thị vô hướng có trọng số liên thông, nghĩa là nó tìm một tập hợp các cạnh của đồ thị tạo thành một cây chứa tất cả các đỉnh, sao cho tổng trọng số các cạnh của cây là nhỏ nhất.

Trong mỗi bước của thuật toán ta luôn tìm cách bổ sung vào cây một cạnh (cùng với đỉnh tương ứng) có trọng số nhỏ nhất trong tất cả các cạnh có thể, sao cho nó liên thông và nối một đỉnh nằm trong cây với một đỉnh nằm ngoài cây để không làm phát sinh chu trình trong cây. Thuật toán dừng lại sau khi tất cả các đỉnh được bổ sung vào cây.

**Các bước thực hiện:**

Bước 1: Chọn một đỉnh bất kỳ, đỉnh này có thể chọn bất kỳ hoặc theo một yêu cầu cụ thể.

Bước 2: Nếu có đỉnh chưa thuộc cây khung T nhưng kề với một đỉnh trong T thì chọn cạnh có trọng số nhỏ nhất trong số đó cho vào T.

Bước 3: Lặp lại cho đến khi tất cả đỉnh được thêm vào T và cây khung nhỏ nhất được hình thành.

Bước 4: Kết thúc thuật toán.

**Đánh giá thuật toán:**

Do khối lượng kiến thức còn hạn hẹp nên nhóm chúng em chỉ đưa ra đánh giá độ phức tạp trên ma trận kề, nếu sử dụng ma trận kề thì độ phức tạp của thuật toán Prim là O(V^2) với V là số lượng đỉnh. Nếu kết hợp với cấu trúc Heap thì độ phức tạp của thuật toán sẽ là O((m + n)lgn).

**2.2. Giới thiệu về thuật toán Dijkstra**

Các bài toán về tìm đường đi ngắn nhất và các biến tướng của nó không chỉ xuất hiện trong mỗi các môn học về toán, lập trình mà còn luôn hiện hữu trong các khía cạnh khác của đời sống hằng ngày. Ví dụ như việc tìm đường đi ngắn nhất từ thành phố này sang thành phố khác, tìm đường đi sử dụng các phương tiện công cộng hiệu quả, hay hiện nay phổ biến là tìm đường đi tối ưu nhất để thu nhập thông tin trên internet. Để đáp ứng việc ấy, con người đã không ngừng tìm hiểu và cho ra đời các thuật toán để giải quyết các vấn đề trên, một trong số đó có thể kể đến là thuật toán Dijkstra.

Thuật toán Dijkstra được đặt tên theo nhà toán học người Hà Lan Edsger Dijkstra, ông giới thiệu thuật toán này trong bài báo “A Note on Two Problems in Connexion with Graphs” được công bố vào năm 1959 trong tạp chí Numerische Mathematik. Bài báo này giới thiệu không chỉ thuật toán Dijkstra mà còn một số ý tưởng và phương pháp liên quan đến lý thuyết đồ thị và giải quyết vấn đề tìm đường đi ngắn nhất trong một đồ thị có trọng số.

Thuật toán Dijkstra được xem là công cụ quan trọng và phổ biến trong lĩnh vực đồ thị và tìm đường đi. Nó được ứng dụng rộng rãi trong các hệ thống định tuyến mạng, giải quyết vấn đề đường đi ngắn nhất trong các mạng lưới điện, quản lý đường đi giao thông,…và nhiều ứng dụng khác liên quan đến tối ưu hóa đường đi trong các hệ thống nhằm tiết kiệm chi phí mà vẫn đảm bảo tính hiệu quả.

**Mô tả thuật toán:**

Thuật toán Dijkstra là một thuật toán tìm đường đi ngắn nhất trong đồ thị có trọng số không âm. Nó hoạt động bằng cách duyệt qua các đỉnh và cập nhật khoảng cách từ đỉnh nguồn đến các đỉnh kề, sau đó chọn đỉnh có khoảng cách nhỏ nhất làm đỉnh hiện tại. Quá trình lặp này tiếp tục cho đến khi tất cả các đỉnh được xét và thuật toán kết thúc khi chọn được khoảng cách nhỏ nhất từ tất cả các đỉnh trong đồ thị.

**Các bước thực hiện:**

Bước 1: Từ đỉnh gốc, khởi tạo khoảng cách tới chính nó là 0 và khoảng cách nhỏ nhất ban đầu tới các đỉnh khác là (∞; -), ta được danh sách các khoảng cách tới các đỉnh.

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

Bước 3: Lần lượt xét các đỉnh kề B của đỉnh A, Nếu khoảng cách từ đỉnh gốc tới đỉnh B nhỏ hơn khoảng cách hiện tại đang được ghi nhận, cập nhật giá trị và đỉnh kề A vào khoảng cách hiện tại của B.

Bước 4: Sau khi xét tất cả đỉnh kề B của đỉnh A, danh sách khoảng cách tới các điểm đã được cập nhật, quay lại bước 2 với danh sách này. Thuật toán kết thúc khi chọn được khoảng cách nhỏ nhất từ tất cả các điểm.

**Đánh giá thuật toán:**

Thuật toán Dijkstra đảm bảo tìm ra đường đi ngắn nhất giữa các đỉnh trong một đồ thị với trọng số không âm, nếu có nhiều hơn một đường đi thì thuật toán vẫn có thể tìm ra tất cả đường đi ngắn nhất. Tuy nhiên, thuật toán Dijkstra không thích hợp cho việc xử lí có đồ thị có trọng số âm, điều mà thuật toán Ford – Bellman làm việc hiệu quả hơn.

**3. PHƯƠNG PHÁP THỰC HIỆN**

**3.1. Công cụ sử dụng**

**Ngôn ngữ Python:**

Python là một ngôn ngữ lập trình bậc cao cho các mục đích lập trình đa năng do Guido van Rossum tạo ra và lần đầu ra mắt vào năm 1991, được thiết kế với ưu điểm mạnh là dễ đọc, dễ học và dễ nhớ. Python là ngôn ngữ có hình thức và cấu trúc rõ ràng, thuận tiện cho người mới học lập trình bởi sự dễ đọc, dễ hiểu và khả năng ứng dụng rộng rãi trong đời sống, đặc biệt là những ngành nghề đòi hỏi kỹ năng lập trình tốt. Vì vậy, nhóm chúng em đã quyết định sử dụng Python làm ngôn ngữ lập trình chính để thiết kế và phát triển đề tài lần này.

**Visual Studio Code:**(Viết sau)

**3.2. Các bước thực hiện**

Đầu tiên, nhóm chúng em tiến hành thảo luận và đưa ra các ý tưởng cho việc lập trình giao diện đồ họa, lấy nền tảng minh họa thuật toán Prim và Dijkstra, sau đó nhóm tiến hành tập hợp và thống nhất các ý tưởng cần thiết cho việc thiết kế.

Tiếp đến, các thành viên trong nhóm tiếp tục tìm hiểu kiến thức liên quan về cơ sở lý thuyết của hai thuật toán trên, song song với đó là tìm hiểu về ngôn ngữ Python cũng như cách ứng dụng của nó cho việc lập trình, phát triển đề tài.

Sau khi các thành viên trong nhóm đã nắm vững các kiến thức cần thiết, cả nhóm tiến hành code giao diện minh họa, đồng thời hỗ trợ lẫn nhau khi cần thiết. Trong quá trình code, cả nhóm cũng tiến hành nhiều lần chạy thử và sửa code nếu vấn đề phát sinh.

Khi hoàn tất việc code giao diện đồ họa, kiểm tra kỹ lưỡng để chắc rằng giao diện đáp ứng đầy đủ các chức năng cần thiết để minh họa hai thuật toán Prim và Dijkstra, cả nhóm tiến hành hỗ trợ lẫn nhau trong việc viết báo cáo và thiết kế powerpoint.

**3.3. Mô tả thuật toán**

**Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự động**

- Gồm các hàm check

* Check đồ thị có hướng hay có hướng bằng đường chéo chình
* Check xem có trọng số âm hay không để chạy thuật toán Dijsktra
* Check đồ thị có hướng hay không bằng cách so sánh các phần tử ij đối xứng qua chéo chính

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự động

- Thuật toán Dijkstra

* distances: Một từ điển lưu trữ khoảng cách ngắn nhất từ start\_vertex đến mọi đỉnh khác. Ban đầu, tất cả các khoảng cách được đặt là vô cùng lớn, ngoại trừ start\_vertex có khoảng cách bằng 0.
* predecessors: Một từ điển lưu trữ đỉnh liền trước trên đường đi ngắn nhất từ start\_vertex đến mọi đỉnh khác. Ban đầu, tất cả các giá trị là None.
* priority\_queue: Một hàng đợi ưu tiên lưu trữ các đỉnh và khoảng cách tương ứng từ start\_vertex. Ban đầu, chỉ chứa start\_vertex với khoảng cách là 0.

a. Chọn đỉnh có khoảng cách ngắn nhất:

* Sử dụng heapq.heappop để lấy đỉnh có khoảng cách ngắn nhất từ priority\_queue.
* Nếu khoảng cách đã được cập nhật trước đó, bỏ qua bước này.

b. Duyệt qua các đỉnh kề:

* Duyệt qua các đỉnh kề của đỉnh hiện tại.
* Cập nhật khoảng cách từ start\_vertex đến đỉnh kề nếu có đường đi ngắn hơn.

c. Đặt đỉnh kề vào priority\_queue:

* Đặt đỉnh kề và khoảng cách tương ứng vào priority\_queue nếu khoảng cách đã được cập nhật.

Tạo đường đi ngắn nhất:

* Sử dụng predecessors để theo dõi đỉnh liền trước trên đường đi ngắn nhất từ start\_vertex đến target\_vertex.
* Xây dựng đường đi bằng cách đi từ target\_vertex về start\_vertex theo các đỉnh liền trước đã lưu trong predecessors.
* Kết quả
* shortest\_path: Đường đi ngắn nhất từ start\_vertex đến target\_vertex.
* distances[target\_vertex]: Khoảng cách ngắn nhất từ start\_vertex đến target\_vertex.

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự động

- Thuật toán Prim

* Tạo một đồ thị mới (min\_spanning\_tree) để lưu cây bao phủ nhỏ nhất.
* Tạo một tập hợp rỗng visited để theo dõi các đỉnh đã được thăm.
* Thêm start\_vertex vào visited.

a. Tìm các cạnh có thể thêm vào cây:

* Duyệt qua các đỉnh đã thăm (visited).
* Duyệt qua các cạnh kết nối với các đỉnh đã thăm, nhưng đỉnh kết nối chưa được thăm.
* Thêm các cạnh này vào danh sách possible\_edges.

b. Chọn cạnh có trọng số nhỏ nhất:

* Sử dụng hàm min để chọn cạnh có trọng số nhỏ nhất từ danh sách possible\_edges.

c. Thêm đỉnh mới và cạnh vào cây:

* Thêm đỉnh kết nối chưa được thăm vào visited.
* Thêm cạnh được chọn vào cây bao phủ (min\_spanning\_tree).

- > min\_spanning\_tree: Cây bao phủ nhỏ nhất.

**4. KẾT QUẢ ĐẠT ĐƯỢC**

( Chèn hình kèm chú thích, tối đa 2 hình kèm chú thích mỗi trang, không quá 4 trang )

**5. ĐỊNH HƯỚNG PHÁT TRIỂN**

Trong quá trình nghiên cứu và phát triển ứng dụng minh họa trực quan, chúng em đã thành công trong việc thiết kế và triển khai giao diện đồ họa để minh họa rõ ràng cách hoạt động của thuật toán Prim và Dijkstra. Kết quả thu được cho thấy ứng dụng có tiềm năng lớn trong việc hỗ trợ quá trình giảng dạy và học tập thuật toán đồ thị, tuy nhiên thì ứng dụng vẫn chưa hoạt động mượt mà như mong muốn, cũng như là chưa hỗ trợ các thuật toán khác.

Đối với hướng phát triển tiếp theo, chúng em đề xuất một số ý kiến nhằm nâng cao tính linh hoạt và tính ứng dụng của ứng dụng. Đầu tiên, có thể mở rộng hỗ trợ cho các thuật toán khác như Kruskal, Bellman-Ford để tăng sự đa dạng trong việc giảng dạy và nghiên cứu. Thêm tính năng tương tác, như thêm/sửa/xóa đỉnh và cạnh trực tiếp trên giao diện, giúp người sử dụng thực hiện thay đổi và quan sát sự thay đổi của thuật toán trong thời gian thực.

Ngoài ra, chúng em cũng tiến hành cải thiện và tối ưu code để chương trình có thể hoạt động ổn định và mượt mà hơn, bằng việc nâng cấp và điều chỉnh code chương trình và thêm các chức năng mới sẽ giúp giao diện hoạt động tốt hơn, tránh các vấn đề phát sinh xuất hiện trong qua trình sử dụng.

Việc thống kê và đánh giá hiệu suất của thuật toán cũng là một hướng phát triển quan trọng. Bằng cách này, người dùng có thể so sánh và hiểu rõ hơn về ưu và nhược điểm của mỗi thuật toán trong các tình huống cụ thể. Không chỉ vậy, chúng em có dự định thiết kế chương trình với khả năng hỗ trợ đa ngôn ngữ chứ không chỉ là mỗi Python và tối ưu hóa giao diện sẽ làm cho ứng dụng trở nên linh hoạt, dễ tiếp cận hơn cho người dùng.

**6. KẾT LUẬN**

Trong bài báo cáo này, chúng em đã tập trung vào việc phát triển và minh họa trực quan hai thuật toán quan trọng trong lĩnh vực đồ thị, đó là thuật toán Prim và Dijkstra, thông qua một giao diện đồ họa dùng cho mục đích giáo dục. Quá trình nghiên cứu và triển khai đã đem lại những kết quả quan trọng và những hiểu biết mới về cách thức hoạt động của các thuật toán này.

Chúng em đã thành công trong việc xây dựng một giao diện đồ họa linh hoạt, giúp người dùng dễ dàng theo dõi và hiểu rõ cách mà thuật toán Prim xây dựng cây khung nhỏ nhất và thuật toán Dijkstra để tìm đường đi ngắn nhất trong đồ thị. Việc trực quan hóa quá trình làm việc của thuật toán không chỉ giúp sinh viên nắm bắt kiến thức một cách dễ dàng, mà còn tạo ra một công cụ hữu ích cho giảng viên và những người quan tâm đến lĩnh vực này để dùng trong công tác giảng dạy.

Không chỉ dừng lại ở việc mô phỏng và minh họa, chúng em còn đề xuất những hướng phát triển tiếp theo để tối ưu hóa ứng dụng, bao gồm sự mở rộng hỗ trợ cho nhiều thuật toán khác, tính năng tương tác nâng cao, thống kê hiệu suất và tối ưu hóa giao diện. Những hướng phát triển này sẽ giúp ứng dụng trở nên mạnh mẽ hơn và đáp ứng tốt hơn đối với nhu cầu của cộng đồng người dùng

Chúng em hy vọng rằng dự án sẽ là một nguồn tài nguyên hữu ích và khuyến khích thêm nhiều nghiên cứu và phát triển trong lĩnh vực này, chúng em xin chân thành cảm ơn sự quan tâm và hỗ trợ từ mọi người trong quá trình thực hiện đề tài này.

**TÀI LIỆU THAM KHẢO**