

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC SƯ PHẠM THÀNH PHỐ HỒ CHÍ MINH
KHOA CÔNG NGHỆ THÔNG TIN



ĐỒ ÁN MÔN HỌC

**MÔ TẢ TRỰC QUAN VÀ DEMO CÁC THUẬT TOÁN
DIJKSTRA VÀ PRIM**

Học phần: <2211COMP170101 – Lý thuyết đồ thị và ứng dụng>

THÀNH PHỐ HỒ CHÍ MINH, THÁNG 5 NĂM 2023

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC SƯ PHẠM THÀNH PHỐ HỒ CHÍ MINH
KHOA CÔNG NGHỆ THÔNG TIN



ĐỒ ÁN MÔN HỌC

**MÔ TẢ TRỰC QUAN VÀ DEMO CÁC THUẬT TOÁN
DIJKSTRA VÀ PRIM**

Học phần: <2211COMP170101 – Lý thuyết đồ thị và ứng dụng>

Giáo viên hướng dẫn: TS.Nguyễn Viết Hưng, GV. Nguyễn Phương Nam

Sinh viên thực hiện: Nguyễn Hồng Long – 45.01.104.130

Phạm Nguyễn Đăng Khoa – 48.01.103.036

Bùi Nguyễn Thanh Bình – 47.01.103.030

Nguyễn Trần Yến Nhi – 47.01.103.076

THÀNH PHỐ HỒ CHÍ MINH, THÁNG 5 NĂM 2023

MỤC LỤC

NHIỆM VỤ THÀNH VIÊN NHÓM	5
DANH MỤC HÌNH ẢNH	6
MỞ ĐẦU	7
1. Lý do chọn đề tài	7
2. Mục tiêu và nhiệm vụ nghiên cứu	8
3. Đối tượng và phạm vi nghiên cứu	8
4. Phương pháp nghiên cứu	9
5. Kết cấu đề tài	9
CHƯƠNG 1: CƠ SỞ LÝ THUYẾT	10
1.1. Những kiến thức cơ sở về thuật toán	10
1.1.1. Khái niệm thuật toán	10
1.1.2. Các đặc trưng của thuật toán	10
1.1.3. Độ phức tạp của thuật toán	10
1.2. Tổng quan về mô phỏng thuật toán	11
1.2.1. Khái niệm mô phỏng thuật toán	11
1.2.2. Ứng dụng của mô phỏng thuật toán	11
1.2.3. Một số yêu cầu đối với mô phỏng thuật toán	12
1.3. Tổng quan về đồ thị	13
1.3.1. Định nghĩa đồ thị	13
1.3.2. Phân loại đồ thị	13
1.3.3. Cây khung và cây khung nhỏ nhất	14
1.3.4. Các phương pháp biểu diễn đồ thị	14
CHƯƠNG 2: THUẬT TOÁN DIJKSTRA VÀ THUẬT TOÁN PRIM	21
2.1 Thuật toán Prim	21
2.1.1. Giới thiệu về thuật toán Prim	21
2.1.2. Mô tả về thuật toán Prim	21
2.1.3. Các bước giải thuật Prim	21
2.2. Thuật toán Dijkstra	23

2.2.1. Giới thiệu về thuật toán Dijkstra	23
2.2.2. Mô tả thuật toán Dijkstra	24
2.2.3. Các bước giải thuật Dijkstra	25
CHƯƠNG 3: HỆ THỐNG MÔ PHỎNG THUẬT TOÁN	28
3.1. Các chức năng chính	28
3.2. Cài đặt thuật toán	29
3.3. Mô phỏng thuật toán	33
KẾT LUẬN	41
TÀI LIỆU THAM KHẢO	42

NHIỆM VỤ THÀNH VIÊN NHÓM

STT	Họ và tên	MSSV	Nhiệm vụ	Tự đánh giá
1	Nguyễn Hồng Long	45.01.104.130	Viết Word, PowerPoint Xây dựng source code Prim, Dijkstra	100%
2	Phạm Nguyễn Đăng Khoa	48.01.103.036	Xây dựng source code Prim, Dijkstra thuyết trình + Demo	100%
3	Bùi Nguyễn Thanh Bình	47.01.103.030	Viết Word, Powerpoint, xây dựng source code Prim, Dijkstra	100%
4	Nguyễn Trần Yến Nhi	47.01.103.076	Xây dựng source code Prim, Dijkstra, thuyết trình + Demo	100%

DANH MỤC HÌNH ẢNH

Hình 1.1. Ma trận kề vô hướng	15
Hình 1.2. Ma trận kề có hướng	16
Hình 1.3. Ma trận trọng số vô hướng	17
Hình 1.4. Ma trận trọng số có hướng	17
Hình 1.5. Ma trận liên thuộc vô hướng	18
Hình 1.6. Ma trận liên thuộc có hướng	19
Hình 1.7. Ma trận bậc	20
Hình 2.1. Nhà khoa học máy tính Edsger Dijkstra	24
Hình 3.1. Giao diện chính	28
Hình 3.2. Giao diện chính	31
Hình 3.3. Bước đầu mô phỏng thuật toán	33
Hình 3.4. Insert node	34
Hình 3.5. Liên kết các đỉnh	35
Hình 3.6. Kết quả sau khi thực hiện Insert link	36
Hình 3.6. Đồ thị mô phỏng	37
Hình 3.7. Thực hiện tìm đường đi ngắn nhất	38
Hình 3.9. Kết quả chạy thuật toán Prim	39
Hình 3.10. Kết quả chạy thuật toán Dijkstra	40

MỞ ĐẦU

Trong thời đại ngày nay thì việc mà chúng ta được học trên những nền tảng khác nhau là một điều khá là phổ biến, không còn là những cách học thông thường như trước đây và bằng những phương pháp học khác nhau trên nhiều nền tảng thì để minh họa trực quan cũng như là để cho người học hiểu rõ hơn về những vấn đề được đặt ra trong các bài học. Thấu hiểu các vấn đề đó thì việc chúng em tìm ra những phương pháp cũng như là cách thức để diễn đạt các vấn đề đó bằng cách tạo ra ứng dụng có thể sử dụng trên máy tính là một điều cần làm đối với chúng em đối với những người sinh viên khoa công nghệ thông tin.

Với những kiến thức đã học từ môn lý thuyết đồ thị, được truyền đạt những kiến thức cũng như là tính ứng dụng của môn lý thuyết đồ thị vào trong cuộc sống và song song với đó là những kiến thức được học từ chuyên ngành. Từ tất cả những điều đó, chúng em càng trở nên thích thú, hăng say và ham học hỏi tìm tòi ra những cái mới hay là hình dung và diễn đạt những thuật toán trong lý thuyết đồ thị bằng nhiều phương pháp khác nhau để có thể hiểu rõ hơn về những thuật toán.

1. Lý do chọn đề tài

Đề tài mô tả trực quan và demo thử các thuật toán Dijkstra và Prim là tạo ra ứng dụng có minh họa trực quan các thuật toán, có vai trò giúp cho người học có thể dễ dàng hình dung và tiếp thu những kiến thức đó bằng những phương pháp mới như là ứng dụng công nghệ thông tin vào trong giáo dục.

Trong thời đại công nghiệp hóa, hiện đại hóa ngày nay, nếu muốn nền giáo dục có những bước tiến mới, để việc dạy học có thể theo kịp cuộc sống, chúng ta cần phải chắc chắn là phải có phương pháp dạy học, trong đó có phương pháp ứng dụng công nghệ thông tin vào giáo dục và sử dụng các thiết bị để hỗ trợ các nền tảng từ

đó tạo ra được tư duy sáng tạo khác nhau, có được những kỹ năng thực hành, tạo ra những hứng thú trong học tập.

Ban đầu nghĩ nó sẽ không phù hợp vì đa phần mọi người sẽ sử dụng cách minh họa thuật toán bằng cách vẽ trên giấy và chạy thuật toán bằng công thức (hay còn được gọi là giải tay). Nhưng dần dần nhận ra rằng việc sử dụng phương pháp cũ thì rất mất thời gian và có thể dẫn đến những cách làm sai, những đáp án sai. Vì thế chúng em đã quyết định phải áp dụng những kiến thức đã học trong chuyên ngành công nghệ thông tin để có thể ứng dụng nó vào môn học.

Như vậy, việc chúng em minh họa trực quan và demo thử các thuật toán Dijkstra và Prim là một lựa chọn vô cùng hợp lý trong bối cảnh ngày nay, bắt kịp xu hướng hiện đại trong giáo dục từ đó mở ra những con đường mới trong việc dạy học.

2. Mục tiêu và nhiệm vụ nghiên cứu

2.1. Mục tiêu nghiên cứu

Giúp cho người học dựa trên những kiến thức lý thuyết đã học từ đó có thể hiểu sâu hơn khi được xem cách minh họa trực quan trên máy tính

2.2. Nhiệm vụ nghiên cứu

Hệ thống hóa lên những thuật toán, làm rõ cách thuật toán được sử dụng, tiếp cận với người dùng nhanh hơn.

Phân tích và đánh giá được chất lượng của phần mềm hỗ trợ, biết được những ưu điểm và hạn chế khi sử dụng phương pháp minh họa trực quan bằng phần mềm.

3. Đối tượng và phạm vi nghiên cứu

3.1. Đối tượng nghiên cứu

Thuật toán Dijkstra và thuật toán Prim trong môn Lý thuyết đồ thị.

3.2. Phạm vi nghiên cứu

Tìm hiểu về thuật toán Dijkstra và thuật toán Prim.

4. Phương pháp nghiên cứu

Việc nghiên cứu đề tài mô tả trực quan và demo các thuật toán Dijkstra và Prim sử dụng phương pháp sau:

- Phương pháp phân tích và tổng hợp.
- Phương pháp quy nạp và diễn giải.

5. Kết cấu đề tài

Chương 1: Cơ sở lý thuyết

Chương 2: Thuật toán Dijkstra và thuật toán Prim

Chương 3: Hệ thống mô phỏng thuật toán

CHƯƠNG 1: CƠ SỞ LÝ THUYẾT

1.1. Những kiến thức cơ sở về thuật toán

1.1.1. Khái niệm thuật toán

Thuật toán là một hệ thống chặt chẽ và rõ ràng các quy tắc nhằm xác định một dãy các thao tác trên cấu trúc dữ liệu sao cho: với một bộ dữ liệu vào, sau một số hữu hạn bước thực hiện các thao tác đã chỉ ra, ta đạt được mục tiêu đã định.

1.1.2. Các đặc trưng của thuật toán

- Tính đơn định: Ở mỗi bước của thuật toán, các thao tác phải hết sức rõ ràng, không gây nên sự nhập nhằng, lộn xộn, đa nghĩa. Thực hiện đúng các bước của thuật toán thì với một bộ dữ liệu vào chỉ cho duy nhất một kết quả ra.
- Tính dừng: Thuật toán không được rơi vào quá trình vô hạn, phải dừng lại và cho kết quả sau một số hữu hạn bước.
- Tính đúng: Sau khi thực hiện tất cả các bước của thuật toán theo đúng quá trình đã định, ta phải đạt được kết quả mong muốn với mọi bộ dữ liệu đầu vào. Kết quả đó được kiểm chứng bằng yêu cầu bài toán.
- Tính phổ dụng: Thuật toán phải dễ sửa đổi để thích ứng được với bất kì bài toán nào trong một lớp các bài toán và có thể làm việc trên các dữ liệu khác nhau.
- Tính khả thi: Kích thước dữ liệu phải đủ nhỏ, thuật toán phải được máy tính thực hiện trong thời gian cho phép, thuật toán phải dễ hiểu và dễ cài đặt.

1.1.3. Độ phức tạp của thuật toán

Cho f và g là hai hàm xác định dương với mọi n . Hàm $f(n)$ được gọi là $O(g(n))$ nếu tồn tại một hằng số $c > 0$ và một giá trị n_0 sao cho: $f(n) \leq c \cdot g(n)$ với mọi $n \geq n_0$

Nghĩa là nếu xét những giá trị $n \geq n_0$ thì hàm $f(n)$ sẽ bị chặn trên bởi một hằng số nhân với $g(n)$. Khi đó, nếu $f(n)$ là thời gian thực hiện của một giải thuật thì ta nói giải thuật đó có cấp là $g(n)$ (hay độ phức tạp tính toán là $O(g(n))$).

1.2. Tổng quan về mô phỏng thuật toán

1.2.1. Khái niệm mô phỏng thuật toán

Mô phỏng thuật toán (Algorithm Animation) là quá trình tách dữ liệu, thao tác, ngữ nghĩa và tạo mô phỏng đồ họa cho quá trình trên [Stasko 1990]. Mô phỏng thuật toán được thiết kế để giúp người dùng có thể hiểu thuật toán, đánh giá chương trình và sửa lỗi chương trình.

Một chương trình máy tính chứa các cấu trúc dữ liệu của thuật toán mà nó thực thi. Trong quá trình thực thi chương trình, các giá trị trong cơ sở dữ liệu được thay đổi.

1.2.2. Ứng dụng của mô phỏng thuật toán

Mô phỏng thuật toán sử dụng biểu diễn đồ họa để biểu diễn cấu trúc dữ liệu và chỉ ra sự thay đổi giá trị trong cơ sở dữ liệu trong mỗi trạng thái. Thông qua đó, người sử dụng có thể xem được từng bước thực thi chương trình và nhờ vậy có thể hiểu chi tiết được thuật toán

Mô phỏng thuật toán cũng được dùng để đánh giá một chương trình đã có bằng cách cung cấp các mô phỏng cho các thành phần của hệ thống, nhờ đó có thể kiểm tra được hiệu năng của hệ thống.

Bên cạnh việc giúp người sử dụng hiểu hơn về hệ thống, mô phỏng thuật toán còn được dùng để giúp thực hiện quá trình dò lỗi dễ dàng hơn. Để sử dụng mô phỏng thuật toán trong quá trình dò lỗi của một chương trình, người sử dụng chú thích vào các trạng thái của chương trình để tạo ra các lệnh mô phỏng, sau đó chúng sẽ được đưa vào hệ thống mô phỏng thuật toán để tạo mô phỏng. Người sử dụng có thể xem chương trình của họ đã thực hiện như thế nào, các giá trị dữ liệu ở mỗi bước và một bước sẽ ảnh hưởng tới các bước sau như thế nào. Nó sẽ giúp người sử dụng tìm ra tất cả các lỗi có thể xảy ra trong chương trình.

1.2.3. Một số yêu cầu đối với mô phỏng thuật toán

- Phản ánh đúng nội dung thuật toán: Thuật toán được đưa ra mô phỏng phải chính xác, các bước thực hiện thuật toán phải trực quan và phản ánh đúng theo nội dung thuật toán đã đưa ra để đảm bảo tính đúng đắn của thuật toán.
- Hệ thống mô phỏng phải được thực hiện theo từng bước: thuật toán thường là trừu tượng, nếu để chương trình chạy tự động thì người dùng sẽ khó hiểu. Vì vậy, cần phải có chế độ thực hiện mô phỏng thuật toán theo từng bước, để người học có thể quan sát, theo dõi sự thay đổi giá trị của từng biến. Nhờ đó, sẽ giúp cho người học hiểu thuật toán rõ hơn và nhanh hơn.
- Mô phỏng thuật toán phải có tính động: để mô tả trực quan hóa quá trình thực hiện của thuật toán ta nên đưa vào hình ảnh động (có thể có âm thanh) để thể hiện sự thay đổi của dữ liệu trong quá trình thực thi. Ví dụ như, trong thuật toán tìm cây khung nhỏ nhất – thuật toán Prim, ta đưa vào các đối tượng đồ họa là đỉnh, cạnh. Trong quá trình thực thi thuật toán thì các đỉnh/cạnh được chọn sẽ sáng nhấp nháy và đổi màu, sau khi được chọn thì các đỉnh/cạnh sẽ được tô màu khác với màu ban đầu. Tương ứng với quan sát sự thay đổi trên đồ thị, ta có thể đưa vào đoạn thuật toán thể hiện tương ứng và song song với quá trình duyệt trên đồ thị. Nhờ đó mà thuật toán được thực hiện một cách rõ nét, sinh động, trực quan, giúp người đọc dễ theo dõi, dễ hiểu thuật toán hơn.
- Thuật toán được thử nghiệm trong mọi trường hợp để đảm bảo thời gian thực hiện tốt nhất: một thuật toán được mô phỏng phải đảm bảo là thuật toán tốt, dễ hiểu và đúng đắn. Muốn vậy ta phải thử nghiệm trong các trường hợp dữ liệu ngẫu nhiên, tốt nhất, xấu nhất. Nếu thuật toán vẫn chạy tốt và trong một thời gian cho phép thì thuật toán mới hiệu quả. Ta không thể chấp nhận một thuật toán đúng mà thời gian chạy quá lớn.

- Tạo ra các mức độ khác nhau cho người dùng: Đối tượng học thuật toán thường là các sinh viên. Họ có trình độ tiếp thu khác nhau, nên ta phải đưa ra nhiều chế độ thao tác khác nhau để người học được phép lựa chọn.

1.3. Tổng quan về đồ thị

1.3.1. Định nghĩa đồ thị

Một đồ thị được hiểu là một bộ hai tập hợp hữu hạn: Tập hợp đỉnh và tập hợp cạnh nối các đỉnh này với nhau. Một đồ thị được mô tả hình thức như sau:

$$G = (V, E)$$

Trong đó V là tập hợp các đỉnh (Vertices) và E là tập hợp các cạnh (Edges).
Như vậy có thể coi đồ thị là tập các cặp (u, v) với u và v là hai đỉnh của V .

1.3.2. Phân loại đồ thị

Chúng ta có thể phân loại đồ thị theo đặc tính và số lượng của tập các cạnh E .
Cho đồ thị $G = (V, E)$.

- G được gọi là đơn đồ thị nếu giữa hai đỉnh u và v của V có nhiều nhất là một cạnh trong E nối từ u tới v .

- G được gọi là đa đồ thị nếu giữa hai đỉnh u và v của V có thể có nhiều hơn một cạnh trong E nối từ u tới v .

- G được gọi là đồ thị có hướng nếu các cạnh trong E là có định hướng, có thể có cạnh nối từ đỉnh u tới đỉnh v nhưng chưa chắc đã có cạnh nối từ đỉnh v tới đỉnh u .

Chu trình của đồ thị: chu trình là một dãy cạnh kế tiếp khép kín sao cho mỗi đỉnh của đồ thị được đi qua không quá một lần.

1.3.3. Cây khung và cây khung nhỏ nhất

- Cây là đồ thị vô hướng, liên thông, không có chu trình đơn. Đồ thị vô hướng không có chu trình đơn được gọi là một rừng (hợp của nhiều cây). Như vậy mỗi thành phần liên thông của một rừng là nhiều cây.
- Cây khung: Cho G là một đơn đồ thị. Một cây được gọi là cây khung của G nếu nó là một đồ thị con của G và chứa tất cả các đỉnh của G .
- Cây khung nhỏ nhất: Ta có thể định nghĩa cây khung nhỏ nhất cho một đồ thị G như sau: Nếu mỗi cạnh $e_{ij} = (v_i, v_j)$ có một trọng số c_{ij} thì cây khung nhỏ nhất là một tập hợp các cạnh ký hiệu là E_{span} , sao cho:

$$C = \sum(c_{ij} \mid \text{với mọi } e_{ij} \text{ thuộc } E_{\text{span}})$$

là nhỏ nhất.

1.3.4. Các phương pháp biểu diễn đồ thị

❖ Biểu diễn đồ thị bằng ma trận kề

Giả sử $G = (V, E)$ là một đơn đồ thị có số đỉnh (kí hiệu $|V|$ là n), không mất tính tổng quát, có thể coi các đỉnh được đánh số $1, 2, \dots, n$. Khi đó có thể biểu diễn đồ thị bằng một ma trận vuông $A = [a_{ij}]$ cấp n . Trong đó:

- $a_{ij} = 1$ nếu (i, j) thuộc E
- $a_{ij} = 0$ nếu (i, j) không thuộc E
- Quy ước $a_{ij} = 0$ với mọi i

Tính chất:

- Nếu nửa tam giác trên và nửa tam giác dưới đối xứng với nhau qua đường chéo chính \Rightarrow là đồ thị vô hướng. Ngược lại nếu có một phần tử không giống nhau \Rightarrow ma trận có hướng.
- Nếu G là đồ thị vô hướng thì bậc của đỉnh i bằng tổng phần tử khác 0 trên hàng i

- Nếu G là đồ thị có hướng thì nửa bậc ngoài của đỉnh i bằng tổng các phần tử khác 0 trên dòng i và nửa bậc trong đỉnh i bằng tổng các phần tử khác 0 trên cột i

Ưu điểm và nhược điểm

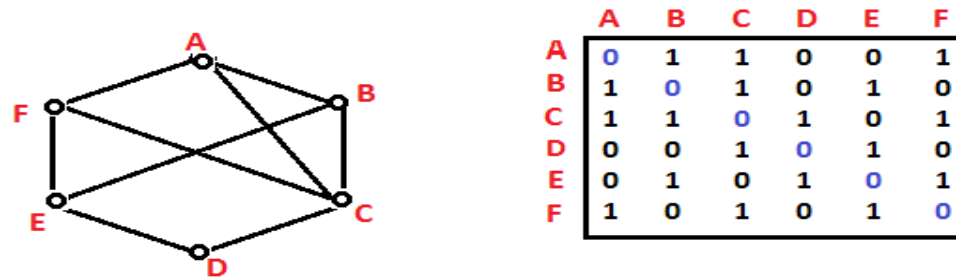
Ưu điểm:

- Đơn giản, dễ biểu diễn
- Nhìn vào ma trận ta biết được 2 đỉnh nào kề nhau.
- Biết được bậc của từng đỉnh nếu là đồ thị đơn

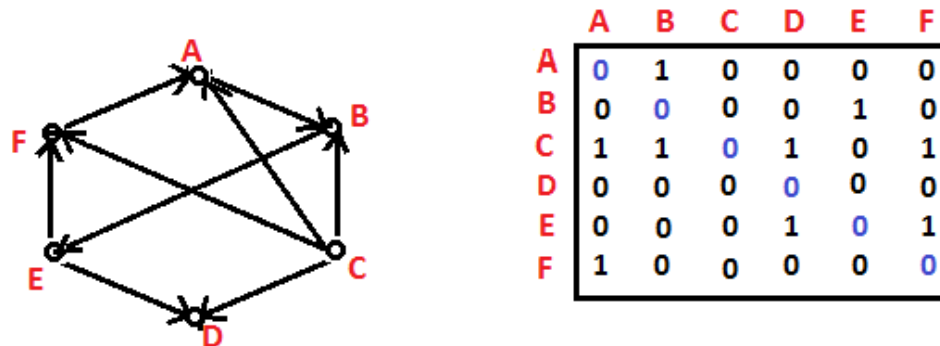
Nhược điểm:

- Không biểu diễn được những cạnh song song

Minh họa:



Hình 1.1. Ma trận kề vô hướng



Hình 1.2. Ma trận kề có hướng

❖ **Biểu diễn đồ thị bằng ma trận trọng số**

Cho $G = (V, E)$ là đơn đồ thị có trọng số. Ma trận trọng số của G là một ma trận vuông $A = (a_{ij})$ cấp n . Trong đó:

- $a_{ij} = w(i, j)$ nếu (i, j) thuộc E
- $a_{ij} = 0$ nếu (i, j) không thuộc E
- Quy ước $a_{ii} = 0$ với mọi i ;

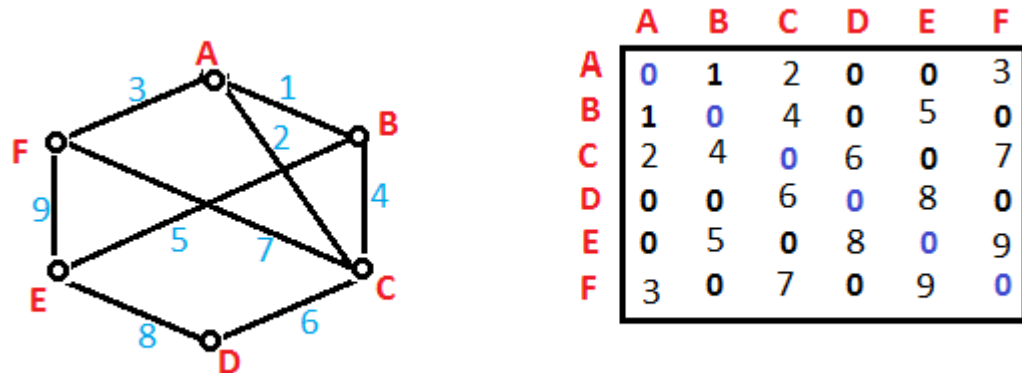
Ưu điểm

Đồ thị có khuyên vẫn biểu diễn được

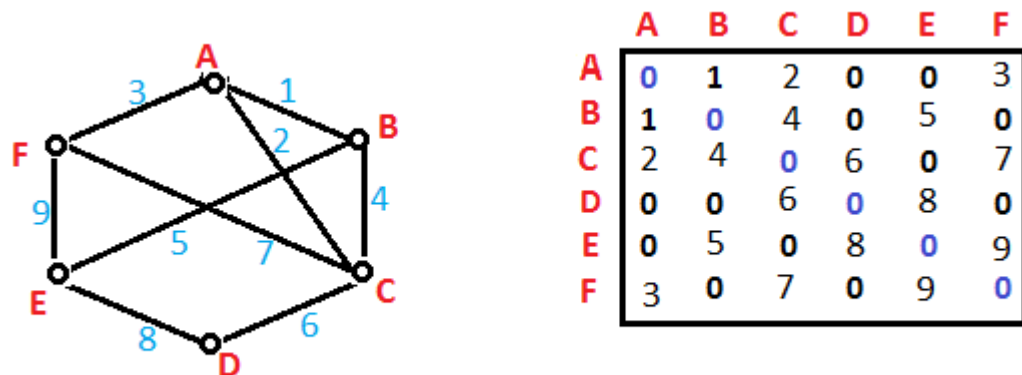
Nhược điểm

Đồ thị có cách song song không biểu diễn được.

Minh họa:



Hình 1.3. Ma trận trọng số vô hướng



Hình 1.4. Ma trận trọng số có hướng

❖ **Biểu diễn đồ thị bằng ma trận liên thuộc**

Giả sử $G = (V, E)$ là một đồ thị vô hướng với các đỉnh v_1, v_2, \dots, v_n và các cạnh là e_1, e_2, \dots, e_m . Khi đó ma trận liên thuộc $M = [m_{ij}]$ kích thước $n \times m$ trong đó:

- $m_{ij} = 0$ nếu cạnh e_j không nối với đỉnh v_i

- $m_{ij} = 1$ nếu cạnh e_j nối với đỉnh v_i
- $m_{ij} = -1$ nếu cạnh e_j đi vào đỉnh v_i

Tính chất

- Trong ma trận của đồ thị có hướng tổng bậc ra của các đỉnh = Tổng bậc vào của các đỉnh = Đỉnh.
- Trong ma trận của đồ thị vô hướng tổng bậc của tất cả các đỉnh = 2 lần số cạnh.

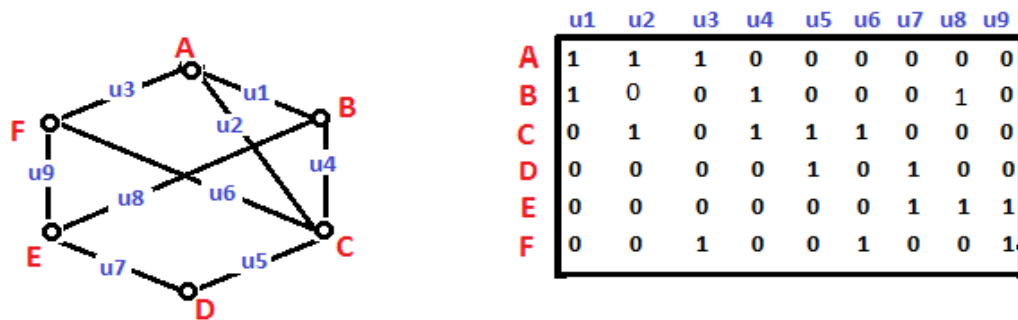
Ưu điểm

- Nhìn vào ma trận ta biết được số lượng cạnh, số lượng đỉnh, bậc trong và bậc ngoài của đỉnh đó.
- Biết được hướng của 1 cạnh xuất phát từ đỉnh nào đi tới đỉnh nào.

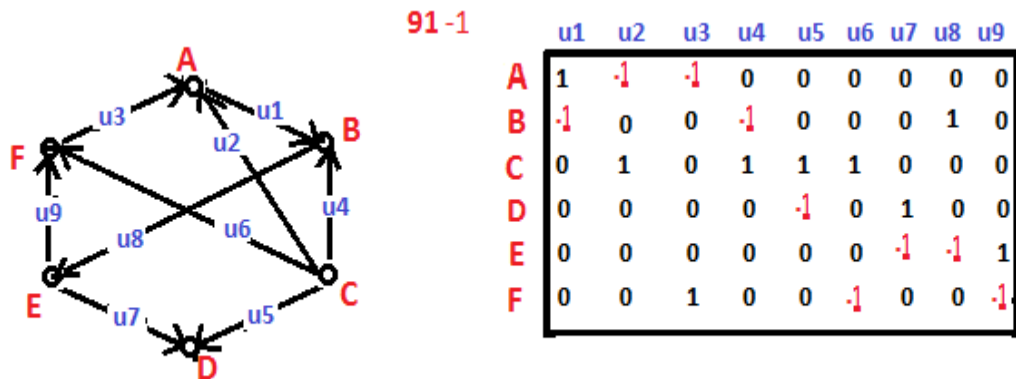
Nhược điểm

- Biểu diễn phức tạp nếu đồ thị có số cạnh nhiều => khó biểu diễn

Minh họa:



Hình 1.5. Ma trận liên thuộc vô hướng



Hình 1.6. Ma trận liên thuộc có hướng

❖ **Biểu diễn đồ thị bằng ma trận bậc**

Ma trận bậc (degree matrix) là một ma trận đường chéo chứa thông tin về bậc của mỗi đồ thị.

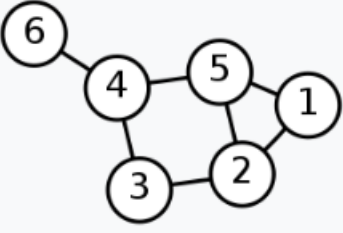
Cho một đồ thị G , ma trận bậc D của đồ thị G là một ma trận vuông $n \times n$ được định nghĩa như sau:

$$d_{i,j} := \begin{cases} \deg(v_i) & \text{khi } i = j \\ 0 & \text{khi } i \neq j \end{cases}$$

Tính chất

Ma trận bậc của đồ thị chính quy bậc k có một đường chéo chứa các hằng số k

Minh họa:

Đồ thị	Ma trận bậc
	$\begin{pmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$

Hình 1.7. Ma trận bậc

CHƯƠNG 2: THUẬT TOÁN DIJKSTRA VÀ THUẬT TOÁN PRIM

2.1 Thuật toán Prim

2.1.1. Giới thiệu về thuật toán Prim

Trong khoa học máy tính, 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.

2.1.2. Mô tả về thuật toán Prim

Thuật toán xuất phát từ một cây chỉ chứa đúng một đỉnh và mở rộng từng bước một, mỗi bước thêm một cạnh mới vào cây, cho tới khi bao trùm được tất cả các đỉnh của đồ thị.

- Dữ liệu vào: Một đồ thị có trọng số liên thông với tập hợp đỉnh V và tập hợp cạnh E (trọng số có thể âm). Đồng thời cũng dùng V và E để ký hiệu số đỉnh và số cạnh của đồ thị.

- Khởi tạo: $V_{\text{mới}} = \{x\}$, trong đó x là một đỉnh bất kì (đỉnh bắt đầu) trong V , $E_{\text{mới}} = \{\}$

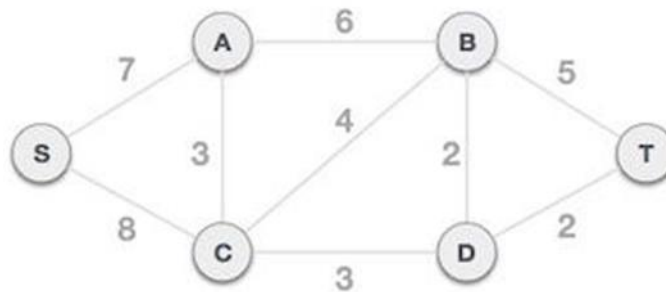
- Lặp lại cho tới khi $V_{\text{mới}} = V$:

- Chọn cạnh (u, v) có trọng số nhỏ nhất thỏa mãn u thuộc $V_{mới}$ và v không thuộc $V_{mới}$ (nếu có nhiều cạnh như vậy thì chọn một cạnh bất kì trong chúng)
 - Thêm v vào $V_{mới}$, và thêm cạnh (u, v) vào $E_{mới}$
- Dữ liệu ra: $V_{mới}$ và $E_{mới}$ là tập hợp đỉnh và tập hợp cạnh của một cây bao trùm nhỏ nhất

2.1.3. Các bước giải thuật Prim

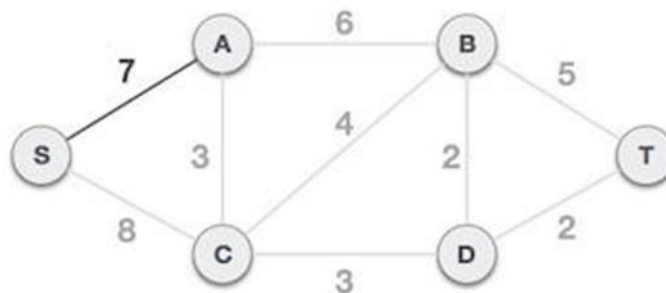
Bước 1: Chọn một nút bất kỳ để làm nút gốc

Giả sử chúng ta chọn nút S làm nút gốc với giải thuật Prim. Chúng ta có thể chọn tùy ý bất kỳ nút nào khác để làm nút gốc

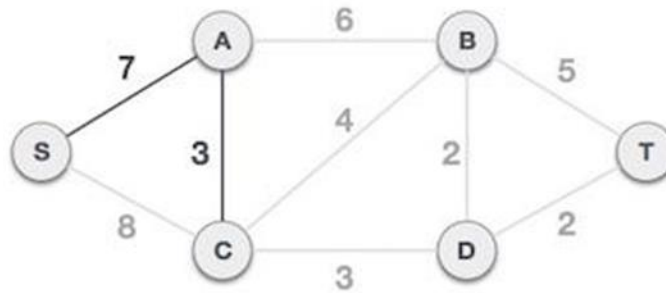


Bước 2: Kiểm tra các cạnh còn lại và chọn một cạnh có trọng số nhỏ nhất

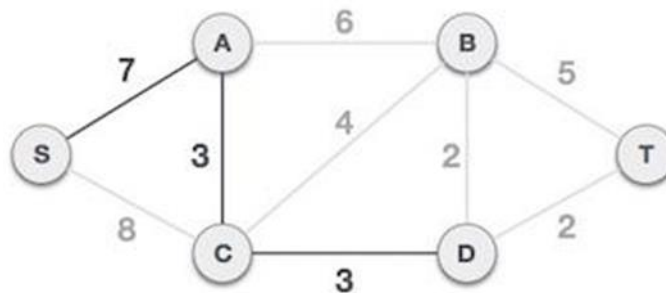
Sau khi chọn nút gốc S, chúng ta thấy rằng SA và SC là hai cạnh có trọng số tương ứng là 7 và 8. Chúng ta chọn cạnh có trọng số nhỏ hơn là SA.



Bây giờ, cây S-7-A được xem như là một nút và chúng ta kiểm tra tất cả các cạnh còn lại bắt đầu từ nút này. Chúng ta tiếp tục chọn cạnh có trọng số nhỏ nhất và thêm nó vào trong cây.



Sau bước này tạo nên cây S-7-A-3-C. Bây giờ chúng ta lại coi đó là một nút và kiểm tra tất cả các cạnh còn lại và sẽ chỉ chọn cạnh có trọng số nhỏ nhất. Trong ví dụ này là cạnh C-3-D có trọng số thấp nhất.



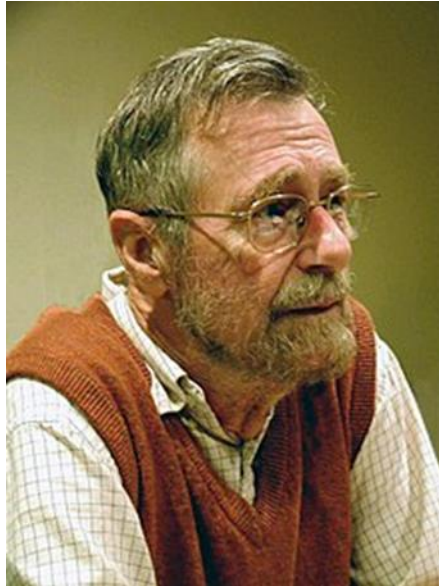
Sau khi thêm nút D vào cây khung, bây giờ chúng ta còn hai cạnh mà có cùng trọng số: D-2-T và D-2-B. Do đó chúng ta có thể thêm một trong hai cạnh. Tuy nhiên, bước tiếp theo sẽ lại thêm cạnh có trọng số 2 còn lại. Dưới đây là hình minh họa sau khi đã thêm hai cạnh.

2.2. Thuật toán Dijkstra

2.2.1. Giới thiệu về thuật toán Dijkstra

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



Hình 2.1. Nhà khoa học máy tính Edsger Dijkstra

Thuật toán Dijkstra là một tham lam phương pháp sử dụng một thực tế toán học rất đơn giản để chọn một nút ở mỗi bước.

2.2.2. Mô tả thuật toán Dijkstra

Tất cả thuật toán tìm đường đi ngắn nhất đều dựa vào việc lồng nhau giữa các đường đi ngắn nhất nghĩa là một nút k thuộc một đường đi ngắn nhất từ i tới j thì đường đi ngắn nhất từ i tới j sẽ bằng đường đi ngắn nhất từ i tới k kết hợp với đường đi ngắn nhất từ k tới j . Vì thế chúng ta có thể tìm đường đi ngắn nhất bằng công thức đệ quy sau:

$$d_j = \min_k (d_i + d_{kj})$$

Thuật toán Dijkstra phù hợp cho việc tìm đường đi ngắn nhất từ một nút i tới tất cả các nút khác. Bắt đầu bằng cách thiết lập

$$d_i = 0 \text{ và } d_j = \infty \text{ nếu } i \neq j$$

Sau đó thiết lập

$d \leftarrow \infty$ là nút kề cận của i

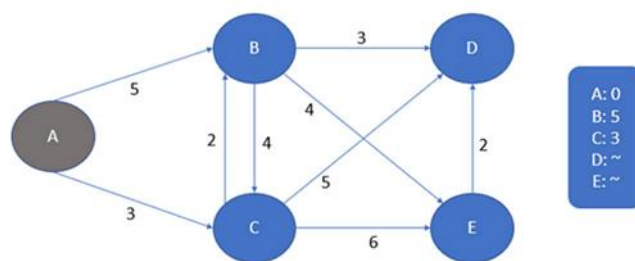
Sau đó tìm nút j có d là bé nhất. Tiếp đó lấy chính nút j vừa chọn để khai triển các khoảng cách các nút khác, nghĩa là bằng cách thiết lập

$d \leftarrow \min (d, d+|l|)$

Tại mỗi giai đoạn của quá trình, giá trị của d là giá trị ước lượng hiện có của đường đi ngắn nhất từ i tới k và thực ra là độ dài đường đi ngắn nhất đã được tìm cho tới thời điểm đó. Xem d như là nhãn trên nút k . Quá trình sử dụng một nút để triển khai các nhãn cho các nút khác gọi là quá trình quét nút.

Thực hiện tương tự, tiếp tục tìm các nút chưa được quét có nhãn bé nhất và quét nó. Chú ý rằng, vì giả thiết tất cả các l đều dương do đó một nút không thể dán cho các nút khác một nhãn bé hơn chính nhãn của nút đó. Vì vậy, khi một nút được quét thì việc quét lại nó nhất thiết không bao giờ xảy ra. Nếu nhãn trên 1 nút thay đổi nhãn đó phải được quét lại

2.2.3. Các bước giải thuật Dijkstra

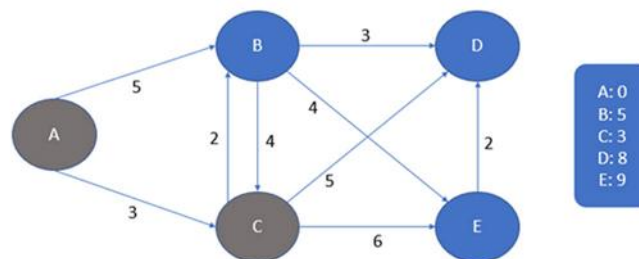


Bước 1: bắt đầu với nút A và ta có 2 con đường.

Đầu tiên là từ A đến B với độ dài 5 và từ A đến C với độ dài 3.

Vì vậy, chúng ta có thể viết trong danh sách kế bên với các đỉnh đã truy cập là 2 đỉnh mới (B, C) và trọng số để đến đó.

Sau đó, như đã nói trước đó – chúng ta sẽ chọn con đường từ A \leftarrow C.



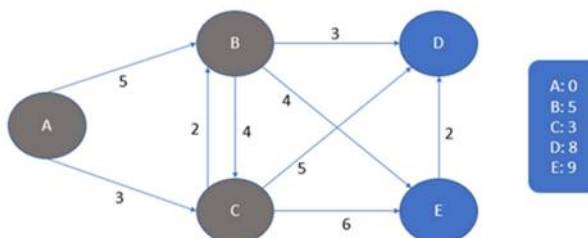
Bước 2: Khi truy cập vào đỉnh C, ta có thể thấy rằng có 3 đường đi khác nhau:

Con đường đầu tiên là C đến B

Con đường thứ hai là C đến D

Con đường thứ ba là C đến E

Vì vậy, ghi vào danh sách hai đỉnh mới và chọn con đường ngắn nhất là C đến B.



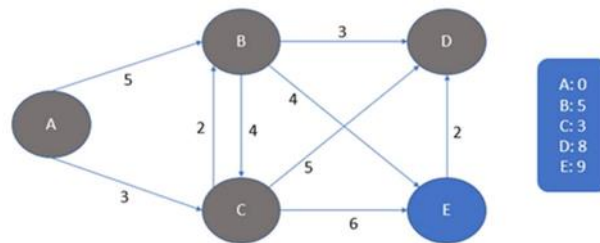
Bước 3: Bây giờ tại B, ta có 3 đường:

B đến D

B đến E

Và B quay lại C

Ta chọn con đường ngắn nhất là B đến D và chúng ta cập nhật vào danh sách trọng số mới của các đường đi từ A đến các đỉnh khác.



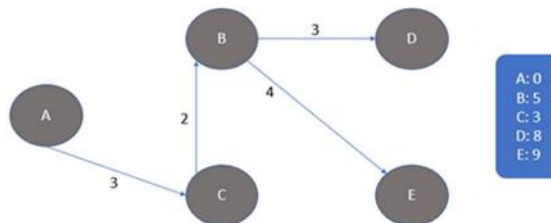
Bước 4: Như ta có thể thấy không có đường đi mới nào từ D đến E. Trong trường hợp đó, ta quay lại đỉnh trước đó để kiểm tra đường đi ngắn nhất.

Bây giờ có một đường với độ dài 4 đi đến E và một đường đi đến C.

Trong trường hợp này, chúng ta chọn bất kỳ đường nào chúng ta thích.

Ta có thể thấy rằng bất kỳ phương án nào chúng ta đi trên đường từ A đến E đều có trọng số như nhau vì các đường đi ngắn nhất được ghi trong danh sách.

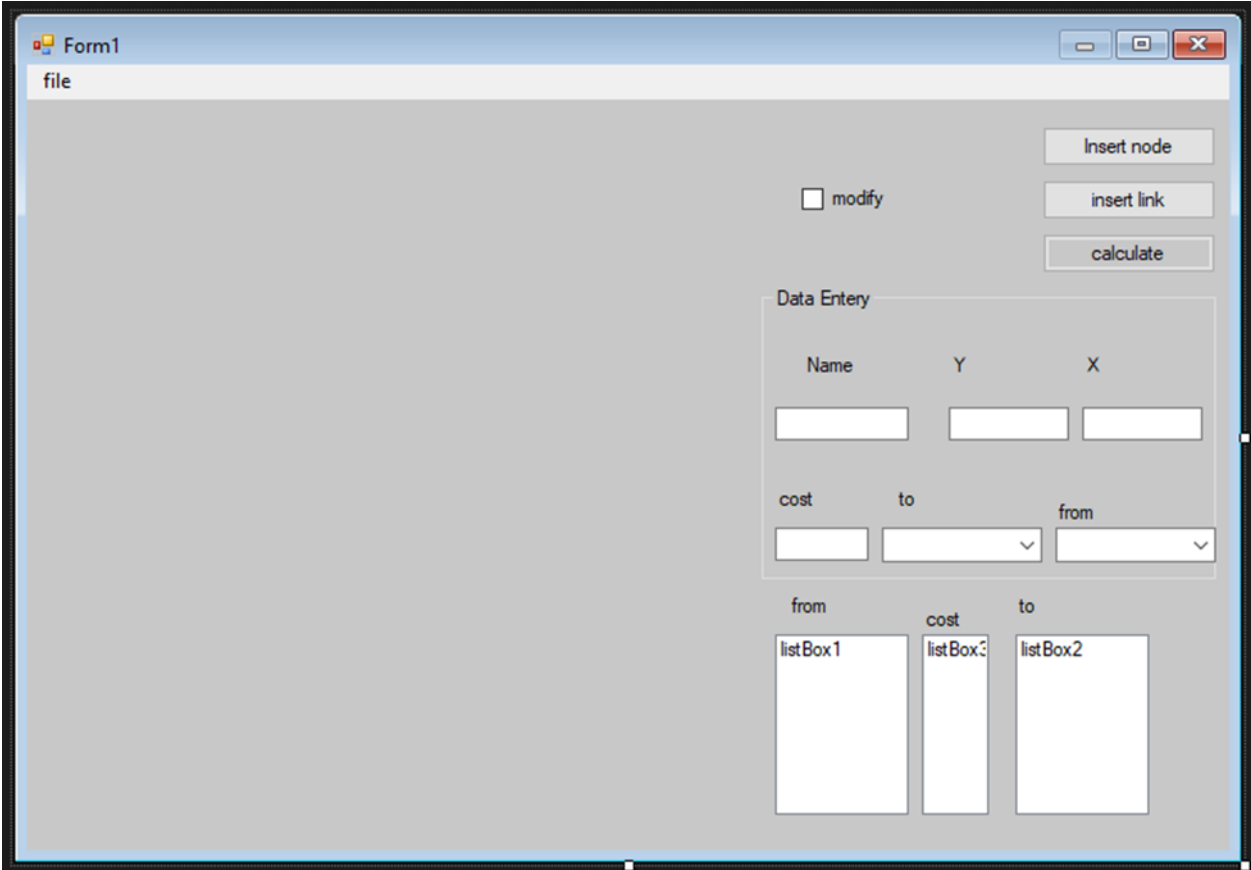
Cuối cùng, ta có tất cả các đường đi mà chúng ta đã sử dụng.



CHƯƠNG 3: HỆ THỐNG MÔ PHỎNG THUẬT TOÁN

3.1. Các chức năng chính

Khi mở chạy chương trình thì một màn hình sẽ xuất hiện như hình bên dưới:



Hình 3.1. Giao diện chính

Giải thích các chức năng trong giao diện

Name: tên điểm

X, Y: tọa độ điểm

Insert node: nút để chèn điểm vào sau khi đã điền đủ Name, X, Y

Cost: trọng số của đường đi (từ điểm bắt đầu đến điểm kết thúc)

To: điểm cuối

From: điểm đầu

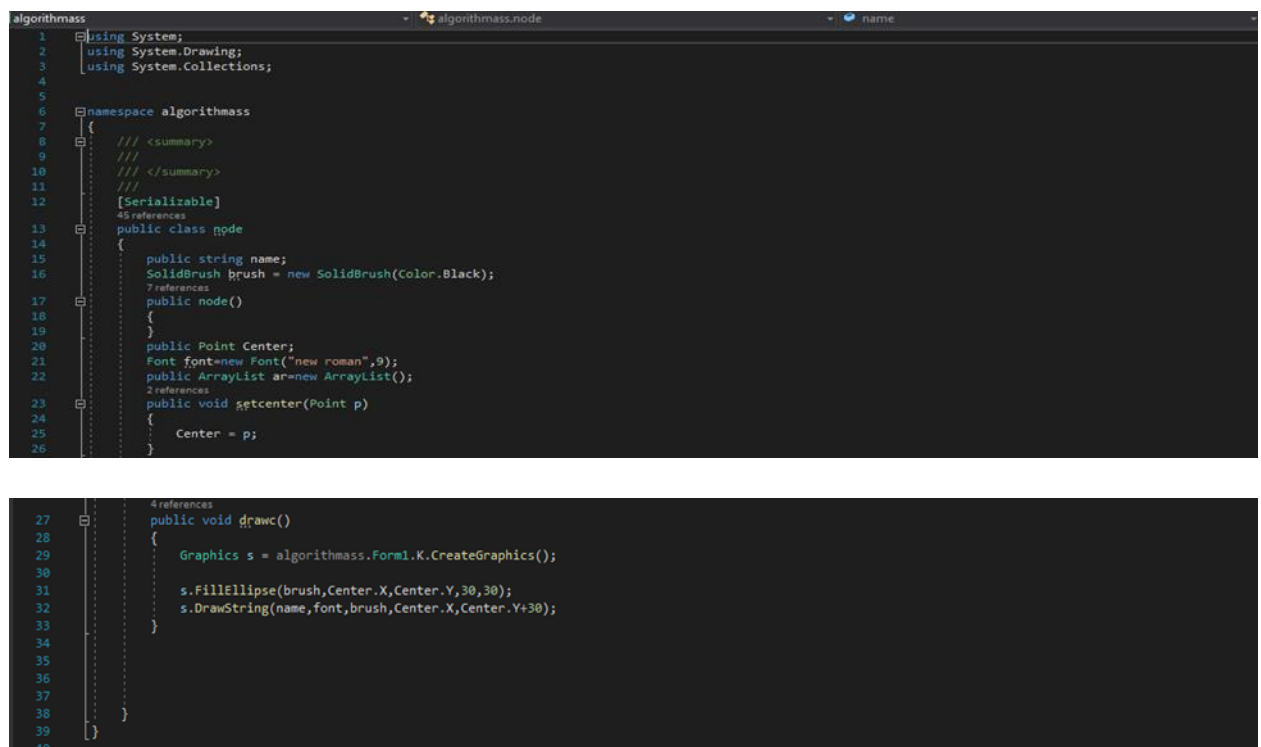
Insert link: là nút dùng để nối hai điểm lại với nhau khi đã điền đủ cost, to và from

Modify: là khi mình muốn di chuyển 1 điểm nào đó thì ta sẽ tích vào ô modify khi không muốn thì tắt

Calculate: là nút để tìm đường đi ngắn nhất

3.2. Cài đặt thuật toán

Node: Gồm tọa độ x,y, và chỉ số node của đỉnh. Khi thuật toán thực thi thì nhận được đưa vào cho mỗi node.



```
1  using System;
2  using System.Drawing;
3  using System.Collections;
4
5
6  namespace algorithmmass
7  {
8      /// <summary>
9      ///
10     /// </summary>
11     ///
12     [Serializable]
13     public class node
14     {
15         public string name;
16         SolidBrush brush = new SolidBrush(Color.Black);
17         public node()
18         {
19         }
20         public Point Center;
21         Font font=new Font("new roman",9);
22         public ArrayList ar=new ArrayList();
23         public void getcenter(Point p)
24         {
25             Center = p;
26         }
27
28         public void draw()
29         {
30             Graphics s = algorithmmass.Form1.K.CreateGraphics();
31             s.FillEllipse(brush,Center.X,Center.Y,30,30);
32             s.DrawString(name,font,brush,Center.X,Center.Y+30);
33         }
34     }
35
36
37
38
39
40
```

Link (lớp thể hiện một cạnh của đồ thị - dùng trong lưu trữ và vẽ đồ thị): Mỗi cạnh được xác định gồm node đầu, node cuối, trọng số của cạnh.

```

1  using System;
2  using System.Drawing;
3
4  namespace algorithmmass
5  {
6      /// <summary>
7      ///
8      /// </summary>
9      ///
10     [Serializable]
11     public class link
12     {
13         2 references
14         public link(long d,string f,string t)
15         {
16             distance=d;
17             from=f;
18             to=t;
19             // T000: Add constructor logic here
20             //
21         }
22         6 references
23         public link()
24         {
25             //
26             // T000: Add constructor logic here
27             //
28         }
29         Pen pen = new Pen(Color.Black,3);
30         SolidBrush brush = new SolidBrush(Color.Red);
31     }

```

```

32     public long distance;
33     public string from;
34     public string to;
35     //Font font = new Font("new roman",9);
36     1 reference
37     public void setcenter(Point p)
38     {
39         Point s = new Point();
40         s.X=p.X+10;
41         s.Y=p.Y+10;
42         Center1=s;
43     }
44     1 reference
45     public void setcenter1(Point p)
46     {
47         Point s = new Point();
48         s.X=p.X+10;
49         s.Y=p.Y+10;
50         Center2=s;
51     }
52     3 references
53     public void drawline()
54     {
55         Point p1=new Point();
56         Point p2=new Point();
57         p1.X=Center1.X;
58         p1.Y=Center1.Y;
59         p2.X=Center2.X;
60         p2.Y=Center2.Y;
61         //
62         /*Point K1=new Point();
63         Point K2= new Point();
64         K1.X=p1.X+10;
65         K2.Y=p2.Y+10;*/

```

```

64         Graphics g = algorithmmass.Form1.K.CreateGraphics();
65         g.DrawLine(pen,p1,p2);
66         //g.DrawString(distance.ToString(),font,brush,K1.X,K2.Y);
67     }
68     public Point Center1,Center2;
69 }
70
71

```

From: dùng để tạo giao diện hệ thống, chứa các hàm vẽ đồ thị (cụ thể là vẽ các node, các link, đồng thời thể hiện đồ thị lên Panel) và hàm thuật toán.

The screenshot shows a Windows-style application window titled "Form1". It has a menu bar with a "file" option. The main workspace is a large gray rectangle. On the right side of this workspace, there is a control panel. At the top of this panel are three buttons: "Insert node", "insert link", and "calculate". Below these buttons is a section titled "Data Entry". This section contains three input fields labeled "Name", "Y", and "X". Below the input fields are three dropdown menus labeled "cost", "to", and "from". At the bottom of the control panel, there are three list boxes labeled "listBox1", "listBox3", and "listBox2". Above each list box is a label: "from" for listBox1, "cost" for listBox3, and "to" for listBox2.

Hình 3.2. Giao diện chính

Cài đặt thuật toán Prim:

```
algorithmmass
- algorithmmass.Form1
- getminimum(long[,] arr, int[] index, out int where, ref int s)

507
508 static long getminimum(long [,]arr,int[] index,out int where,ref int s)
509 {
510     int max=arr.GetLength(1);
511     //temp
512     long temp=0;
513     for(int j=0;j<index.Length;j++)
514     {
515         for(int i=0;i<max;i++)
516         {
517             if(arr[index[j],i]>0)
518             {
519                 temp=arr[index[j],i];
520                 break;
521             }
522         }
523     }
524     where = 0;
525     for(int j=0;j<index.Length;j++)
526     {
527         for(int i=0;i<max;i++)
528         {
529             if(arr[index[j],i]<=temp&&arr[index[j],i]>0)
530             {
531                 temp = arr[index[j],i];
532                 where=i;
533                 s=index[j];
534             }
535             //return the minimum no in the row
536         }
537     }
538     return temp;
539 }
```

Cài đặt thuật toán Dijkstra:

```
algorithmmass
- algorithmmass.Form1
- mainMenu1

507
508 static long getminimum(long[,] arr, int[] index, out int where, ref int s, ref long[] cs, ref long temp, ref int[] point, int k)
509 {
510     int max = arr.GetLength(1);
511     int j = k;
512     for (int i = 0; i < max; i++)
513     {
514         if (arr[index[j], i] > 0 && (arr[index[j], i] + temp < cs[i]))
515         {
516             cs[i] = arr[index[j], i] + temp;
517             point[i] = index[j];
518         }
519     }
520 }
521
522 where = 0;
523 for (int i = 0; i < cs.Length; i++)
524 {
525     if (cs[i] != 1000)
526     {
527         temp = cs[i];
528         break;
529     }
530 }
531 int vt = -1;
532 for (int i = 0; i < cs.Length; i++)
533 {
534     if (cs[i] != 1000 && temp >= cs[i])
535     {
536         temp = cs[i];
537         vt = i;
538     }
539 }
540
541 if (vt == -1) temp = 0;
542 if (vt != -1)
543 {
544     cs[vt] = 1000;
545     where = vt;
546     s = point[vt];
547 }
548 return temp;
549 }
```

3.3. Mô phỏng thuật toán

Đầu tiên ta sẽ điền tên điểm a vào ô Nam và tọa độ vào X và Y

The screenshot shows a window titled 'Form1' with a menu bar containing 'file'. The main area is a large gray rectangle. On the right side, there is a control panel. At the top right of this panel are three buttons: 'Insert node', 'insert link', and 'calculate'. Below these buttons is a checkbox labeled 'modify'. Underneath the checkbox is a section titled 'Data Entry'. This section contains three input fields labeled 'Name', 'Y', and 'X'. The 'Name' field contains the letter 'a', the 'Y' field contains '50', and the 'X' field contains '50'. Below these fields are three dropdown menus labeled 'cost', 'to', and 'from'. At the bottom of the control panel, there are three large empty rectangular boxes labeled 'from', 'cost', and 'to'.

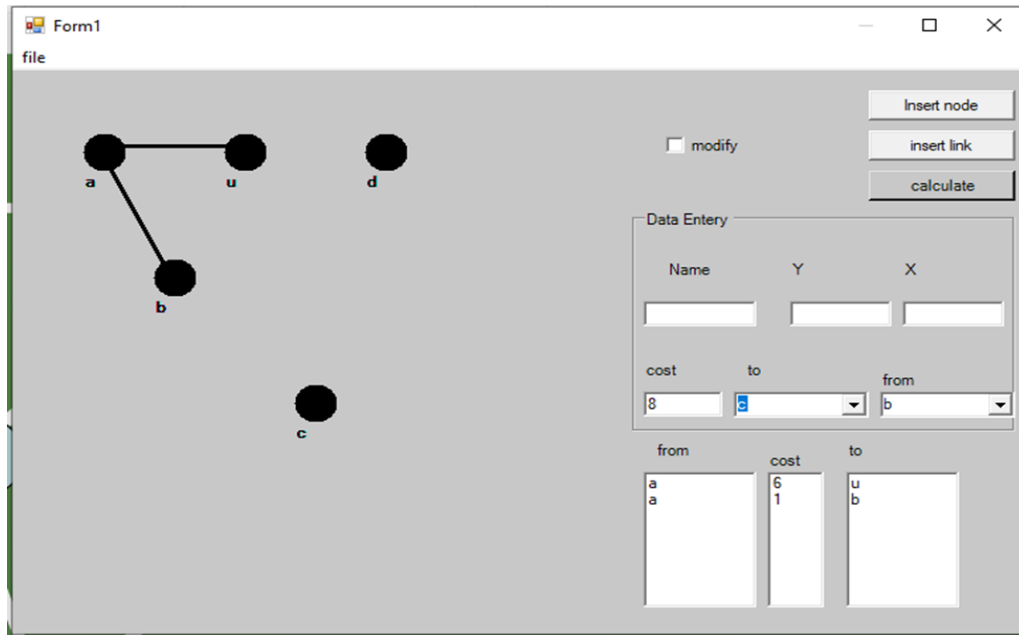
Hình 3.3. Bước đầu mô phỏng thuật toán

Sau khi đã điền đủ thông tin ta nhấn nút Insert node để insert điểm vào

This screenshot shows the same 'Form1' window after the 'Insert node' button has been clicked. In the main gray area, a black circle representing a node is now visible, with the letter 'a' positioned directly below it. The control panel on the right remains unchanged from the previous state, with the 'Data Entry' fields still containing 'a', '50', and '50'.

Hình 3.4. Insert node

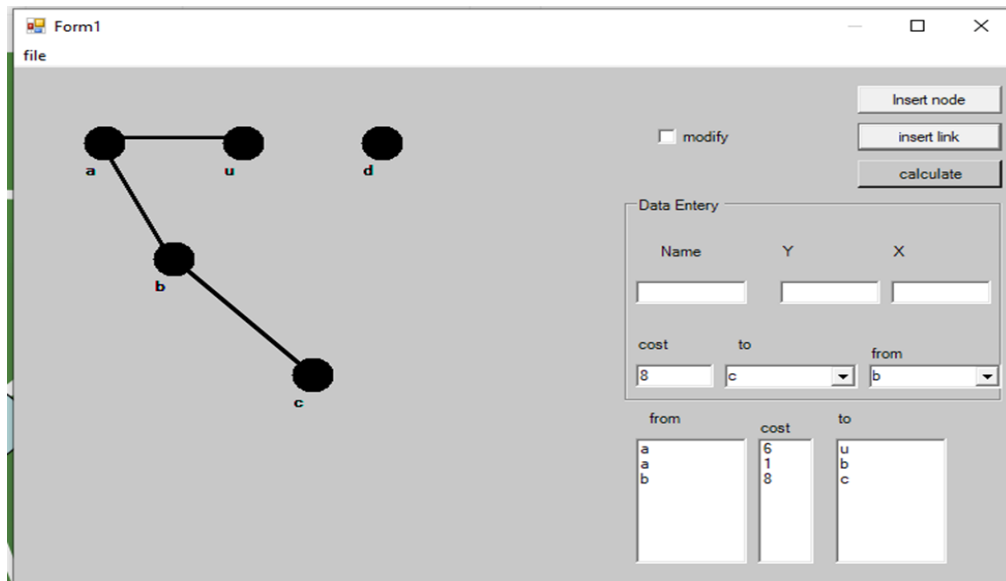
Sau khi insert hết tất cả các điểm thì bước tiếp theo ta sẽ nối các điểm lại với nhau



Hình 3.5. Liên kết các đỉnh

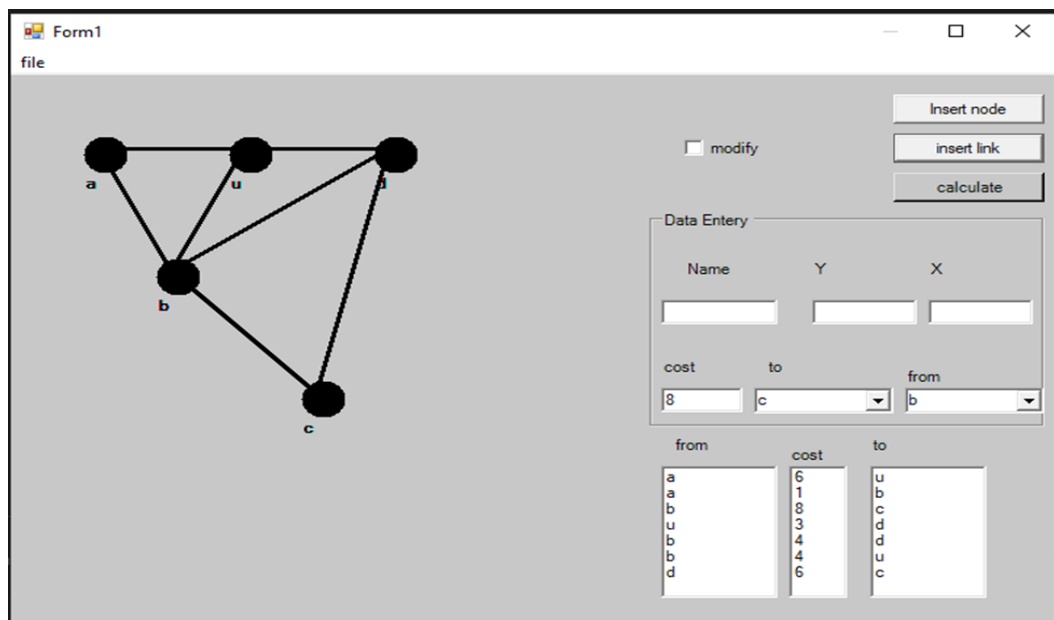
Ở hình trên ta sẽ nối từ điểm b sang điểm c với trọng số là 8

Sau khi điền đầy đủ thông tin ta sẽ nhấn nút Insert link để nối



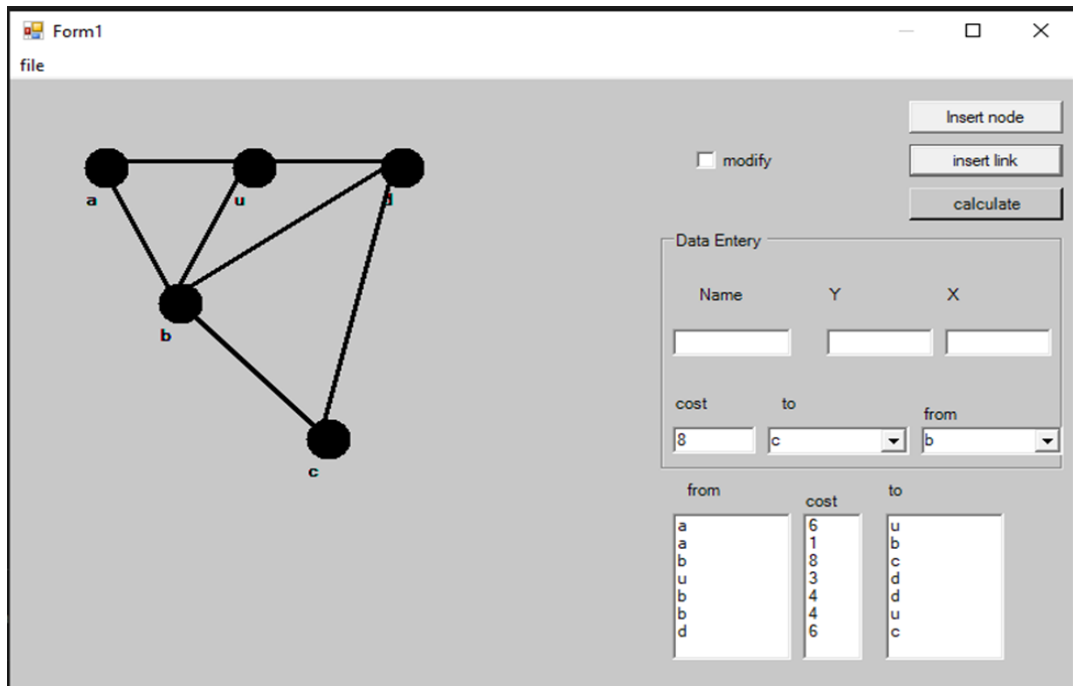
Hình 3.6. Kết quả sau khi thực hiện Insert link

Và dưới đây là đồ thị mà ta muốn vẽ



Hình 3.7. Đồ thị mô phỏng

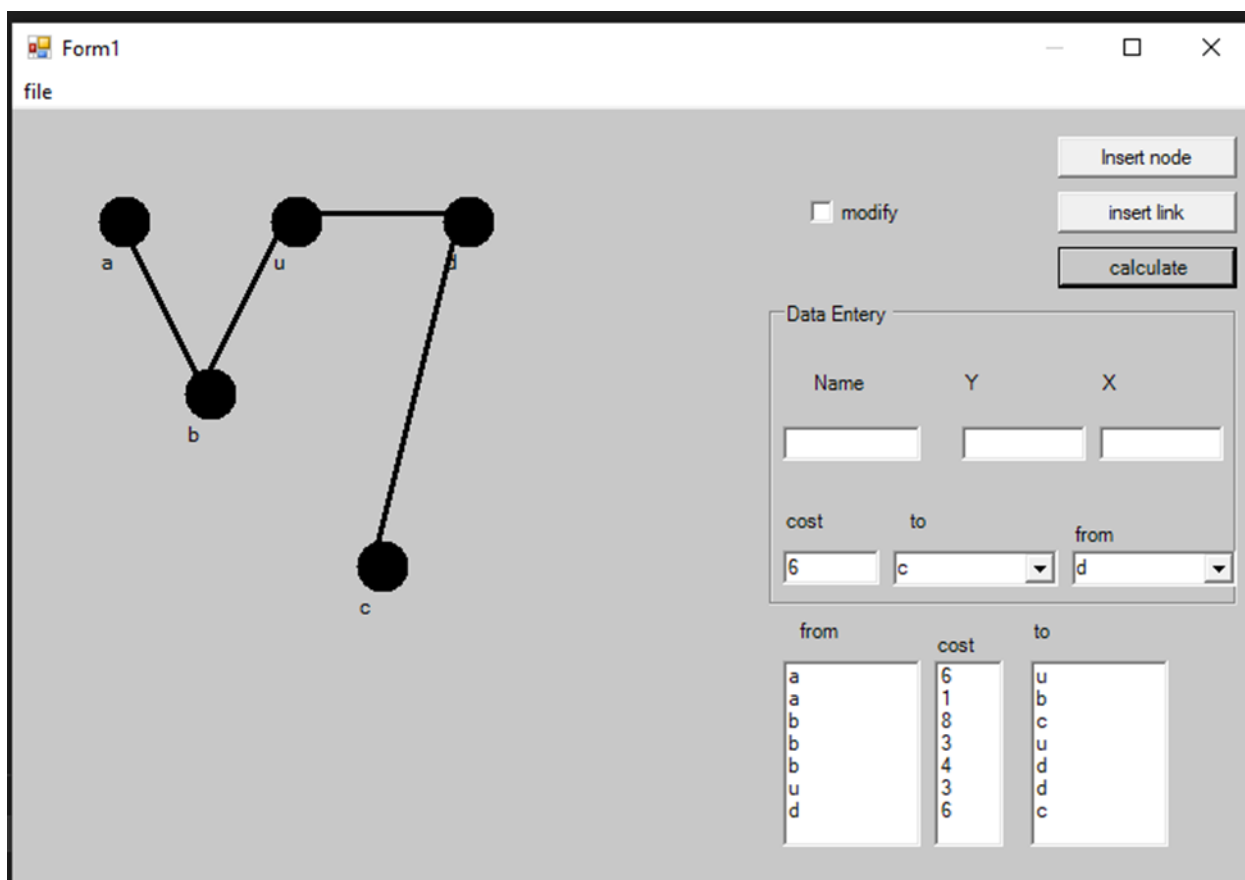
Sau khi đã hoàn thành đồ thị tiếp theo ta ấn nút calculate để tìm đường đi ngắn nhất



Hình 3.8. Thực hiện tìm đường đi ngắn nhất

Giải thích chạy theo thuật toán Prim: ta bắt đầu từ đỉnh C tiếp theo ta chọn đỉnh D và lấy cạnh CD vì cạnh CD nhỏ hơn cạnh CB, tiếp theo từ đỉnh D ta chọn đỉnh U và lấy cạnh DU vì cạnh DU nhỏ hơn cạnh CB và DB, tiếp theo ta đi từ đỉnh U chọn đỉnh B và cạnh UB vì UB nhỏ hơn cạnh CB, DB và UA, tiếp theo ta chọn đỉnh A và cạnh BA vì cạnh BA nhỏ hơn cạnh CB, DB, UA.

Dựa vào kết quả chạy thuật toán Prim ở trên ta có kết quả sau:

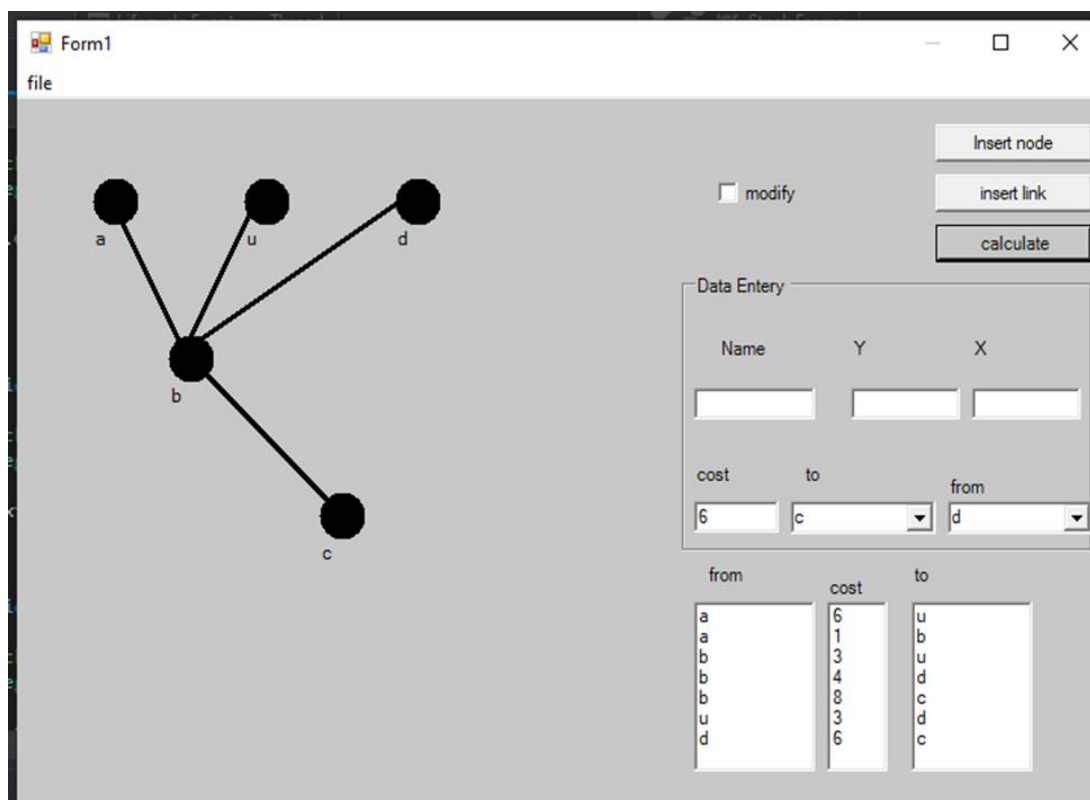


Hình 3.9. Kết quả chạy thuật toán Prim

Giải thích chạy theo thuật toán Dijkstra

a	b	c	d	u
0	$(\infty, -)$	$(\infty, -)$	$(\infty, -)$	$(\infty, -)$
*	$(1, a)^*$	$(\infty, -)$	$(\infty, -)$	$(6, a)$
*	*	$(9, b)$	$(5, b)$	$(5, b)^*$
*	*	$(9, b)$	$(5, b)^*$	*
*	*	$(9, b)^*$	*	*

Dựa vào kết quả chạy thuật toán Dijkstra ở trên ta có kết quả sau:



Hình 3.10. Kết quả chạy thuật toán Dijkstra

KẾT LUẬN

Để góp phần giúp cho người học dễ tiếp thu, hiểu sâu hơn các thuật toán, tạo ra niềm hứng thú trong học tập, có cái nhìn trực quan hơn về thuật toán Dijkstra và Prim, kết hợp với việc dạy học bằng những phương pháp mới trong giáo dục. Điều này là một điều cần thiết trong thời điểm bây giờ, vừa bắt kịp xu hướng thời đại, vừa nâng cao hiệu quả hiệu suất của người học. Điều này đáng ra cần phải được quan tâm và chú trọng hơn và việc tạo ra một ứng dụng và chính những phản hồi của người học về chất lượng của ứng dụng sẽ là nền tảng để có những sản phẩm tốt hơn, là như một bước đệm để tạo ra nhiều ứng dụng có thể phục vụ cho nhiều khía cạnh khác nhau không chỉ là trong giáo dục.

TÀI LIỆU THAM KHẢO

1. Terrence Aluda. Understanding Dijkstra's Shortest Path Algorithm in Network Routing using Python, 18/04/2023, từ <<https://www.section.io/engineering-education/dijkstra-python/?fbclid=IwAR2Tb6cN1f7MqrPKC38XnNNAhX-syAYtPN5Lg18aiJNW3pQzOEshP-Y7Cdc>>
2. Prim's Minimum Spanning Tree(MST) Greedy Algo -5, 21/04/2023, từ <<https://www.geeksforgeeks.org/prims-minimum-spanning-tree-mst-greedy-algo-5/>>
3. Biểu diễn đồ thị, 18/04/2023, từ <https://vi.wikipedia.org/wiki/Biểu_diễn_đồ_thị>