BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC NHA TRANG

KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO ĐỒ ÁN CƠ SỞ NGÀNH

MÔ PHỎNG BIỂU DIỄN ĐỒ THỊ BẰNG MA TRẬN KỀ VÀ DANH SÁCH KỀ

Giảng viên hướng dẫn : **TS. Phạm Thị Thu Thúy**

Sinh viên thực hiện : **Trần Mai Ngọc Duy**

Mã số sinh viên : **65130650**

Khánh Hòa – 2025

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC NHA TRANG
KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO ĐỒ ÁN CƠ SỞ NGÀNH

MÔ PHỎNG BIỂU DIỄN ĐỒ THỊ BẰNG MA TRẬN KỀ VÀ DANH SÁCH KỀ

Giảng viên hướng dẫn : TS. Phạm Thị Thu Thúy
Sinh viên thực hiện : Trần Mai Ngọc Duy
Mã số sinh viên : 65130650

Khánh Hòa – Tháng 12/2025

LỜI CẢM ƠN

Trước tiên, em xin gửi lời tri ân sâu sắc đến Ban Giám hiệu, cùng các quý phòng ban chức năng của Trường Đại học Nha Trang đã luôn quan tâm, tạo mọi điều kiện thuận lợi nhất về cơ sở vật chất, tài liệu học tập và môi trường nghiên cứu để em có thể hoàn thành tốt đề tài này.

Em xin bày tỏ lòng biết ơn chân thành đến toàn thể quý thầy cô Khoa Công nghệ thông tin những người đã truyền đạt cho em kiến thức chuyên môn quý báu suốt trong những năm học vừa qua và hỗ trợ em trong suốt quá trình học tập.

Đặc biệt, em xin gửi lời tri ân sâu sắc nhất đến cô TS. Phạm Thị Thu Thúy cô hướng dẫn trực tiếp của em. Với sự tận tâm, những góp ý quý giá của cô, em đã hoàn thiện đề tài một cách tốt nhất. Sự hướng dẫn nhiệt tình của cô không chỉ giúp em hoàn thành đề tài mà còn là hành trang quý giá cho em trên con đường học tập và nghiên cứu sau này.

Trong suốt quá trình thực hiện bài báo cáo, em đã cố gắng tìm hiểu, tổng hợp và hoàn thiện nội dung trong phạm vi kiến thức và khả năng của bản thân. Mặc dù đã nỗ lực hết sức, nhưng vì kinh nghiệm thực tiễn còn hạn chế và thời gian tiếp cận chưa nhiều nên bài báo cáo khó tránh khỏi những thiếu sót nhất định. Em rất mong nhận được những nhận xét, góp ý chân thành từ quý thầy cô để em có thêm cơ hội học hỏi, tích lũy kinh nghiệm và hoàn thiện bản thân hơn nữa, nhằm hoàn thiện tốt hơn các bài báo cáo và nghiên cứu trong tương lai.

Em xin chân thành cảm ơn!

Khánh Hòa, ngày……tháng……năm……

Sinh viên thực hiện

(Ký và ghi rõ họ tên)

Trần Mai Ngọc Duy

TRƯỜNG ĐẠI HỌC NHA TRANG

KHOA CÔNG NGHỆ THÔNG TIN

PHIẾU THEO DÕI TIẾN ĐỘ VÀ ĐÁNH GIÁ ĐỒ ÁN CƠ SỞ NGÀNH

(Dùng cho giảng viên hướng dẫn và nộp báo cáo đồ án cơ sở ngành của sinh viên)

Tên đề tài: Mô Phỏng Biểu Diễn Đồ Thị Bằng Ma Trận Kề Và Danh Sách Kề

Giảng viên hướng dẫn: TS. Phạm Thị Thu Thúy

Sinh viên được hướng dẫn: Trần Mai Ngọc Duy **MSSV:** 65130650

Khóa: 65 **Ngành:** Công nghệ thông tin

Lần báo cáo	*Ngày*	*Nội dung*	*Nhận xét của GVHD*
01	07/12/2025	Hoàn thành việc xây dựng cơ bản mô phỏng biểu diễn đồ thị bằng ma trận kề và danh sách kề, viết báo cáo lần 1.	Đã hoàn thành cơ bản các chức năng biểu diễn đồ thị. Tuy nhiên, giao diện hiển thị đồ thị trực quan còn nhỏ, chưa phù hợp với đồ thị có kích thước lớn. Cần chỉnh sửa lại vùng vẽ đồ thị với kích thước lớn hơn để dễ quan sát.
02	21/12/2025	Hoàn thành bản mô phỏng biểu diễn đồ thị bằng ma trận kề và danh sách kề đã sửa, viết báo cáo lần 2.	Đã khắc phục được vấn đề về giao diện hiển thị. Chương trình chạy ổn định với các bộ dữ liệu nhỏ. Tuy nhiên, phần báo cáo còn thiếu nội dung về đánh giá hiệu năng và so sánh giữa hai phương pháp biểu diễn. Cần bổ sung thêm chương đánh giá với các bộ test cụ thể, có số liệu định lượng để làm rõ ưu nhược điểm của từng cách biểu diễn.
03	24/12/2025	Cập nhật lại báo cáo và mã nguồn.	Bổ sung chi tiết phân tích kết quả (giải thích các bảng số liệu). Thêm so sánh rõ ràng hơn về thời gian và bộ nhớ giữa ma trận kề vs danh sách kề. Định lượng cụ thể về trường hợp nên sử dụng phương pháp nào Mã nguồn cần tách module rõ ràng hơn.

04	26/12/2025	Hoàn thành báo cáo và cập nhật lại mã nguồn.	

Nhận xét chung (sau khi sinh viên hoàn thành đồ án cơ sở ngành):

.....

.....

Điểm hình thức:……/10 Điểm nội dung:...../10 **Điểm tổng kết**:……/10

Kết luận sinh viên: Được bảo vệ: ☐ Không được bảo vệ: ☐

Khánh Hòa, ngày…….tháng…….năm……

Giảng viên hướng dẫn

(Ký và ghi rõ họ tên)

Phạm Thị Thu Thúy

TRƯỜNG ĐẠI HỌC NHA TRANG

KHOA CÔNG NGHỆ THÔNG TIN

PHIẾU CHẤM ĐIỂM ĐỒ ÁN CƠ SỞ NGÀNH

(Dành cho giảng viên chấm phản biện)

1. Họ tên giảng viên chấm: .....

2. Sinh viên thực hiện: Trần Mai Ngọc Duy **MSSV:** 65130650

3. Tên đề tài: Mô Phỏng Biểu Diễn Đồ Thị Bằng Ma Trận Kề Và Danh Sách Kề

4. Nhận xét

a) Kết quả thực hiện đề tài

.....

.....

.....

b) Báo cáo đồ án cơ sở ngành

- Hình thức: .....

.....

.....

- Nội dung: .....

.....

.....

.....

Điểm hình thức:……/10 Điểm nội dung:...../10 **Điểm tổng kết**:……/10

Khánh Hòa, ngày…….tháng…….năm……

Giảng viên chấm phản biện

(Ký và ghi rõ họ tên)

MỤC LỤC

DANH MỤC HÌNH ẢNH

DANH MỤC BẢNG

Chương 1. TỔNG QUAN VỀ ĐỀ TÀI

1.1 LÝ DO CHỌN ĐỀ TÀI

Trong bối cảnh công nghệ thông tin phát triển mạnh mẽ, đồ thị trở thành cấu trúc dữ liệu nền tảng trong nhiều lĩnh vực quan trọng như mạng xã hội, hệ thống đề xuất, mạng máy tính, trí tuệ nhân tạo và sinh học tin học. Vì vậy, việc hiểu rõ cách biểu diễn đồ thị trong máy tính là kiến thức cơ bản và cần thiết đối với sinh viên ngành Công nghệ thông tin.

Hiện nay, hai phương pháp biểu diễn đồ thị phổ biến nhất là ma trận kề và danh sách kề, mỗi phương pháp có ưu điểm và hạn chế riêng. Ma trận kề có ưu thế trong việc kiểm tra cạnh nhanh với thời gian O(1) nhưng tiêu tốn nhiều bộ nhớ $O(n^2)$, trong khi danh sách kề tiết kiệm bộ nhớ O(n+m), đặc biệt hiệu quả với đồ thị thưa. Việc so sánh và trực tiếp cài đặt hai cách biểu diễn này giúp sinh viên hiểu rõ sự đánh đổi giữa bộ nhớ và hiệu năng trong thực tế.

Xuất phát từ đó, em lựa chọn đề tài **"Mô phỏng biểu diễn đồ thị bằng Ma trận kề và Danh sách kề"** với mục tiêu xây dựng một công cụ trực quan, giúp minh họa rõ ràng cách biểu diễn đồ thị, hỗ trợ việc học tập môn Cấu trúc dữ liệu và giải thuật, đồng thời tạo nền tảng để mở rộng nghiên cứu và cài đặt các thuật toán đồ thị nâng cao trong tương lai.

1.2 MỤC TIÊU NGHIÊN CỨU

1.2.1 Kiến thức đạt được

Hệ thống hóa và củng cố các khái niệm cơ bản của lý thuyết đồ thị như: đồ thị có hướng và vô hướng, có trọng số và không trọng số, đỉnh, cạnh, đỉnh kề, cạnh kề, bậc của đỉnh, đồ thị thưa và đồ thị dày. Nắm vững hai phương pháp biểu diễn đồ thị phổ biến trong máy tính là ma trận kề và danh sách kề.

1.2.2 Chức năng phần mềm

Xây dựng chương trình cho phép nhập đồ thị từ bàn phím hoặc từ file, tự động sinh hai cấu trúc biểu diễn ma trận kề và danh sách kề. Hiển thị đồ thị dưới nhiều dạng bảng ma trận, danh sách kề và hình ảnh đồ thị trực quan. Cung cấp các chức năng mở rộng như thêm hoặc xóa đỉnh và cạnh, xuất nhập dữ liệu, nâng cao tính tương tác và khả năng mở rộng của phần mềm.

1.2.3 Đánh giá và so sánh

Phân tích và so sánh ưu, nhược điểm của ma trận kề và danh sách kề về độ phức tạp bộ nhớ, thời gian truy vấn và khả năng áp dụng trong thực tế. Đánh giá trực quan mức độ thưa hoặc dày của đồ thị và sự khác biệt hiệu năng giữa hai cách biểu diễn khi số đỉnh và số cạnh thay đổi.

Bảng 1.1. So sánh ma trận kề và danh sách kề

Tiêu chí đánh giá	Ma trận kề	Danh sách kề	Nên dùng khi
Độ phức tạp bộ nhớ	$O(n^2)$	O(n + m)	Danh sách kề nếu $m << n^2$
Kiểm tra cạnh (u,v)	O(1)	O(deg(u)) hoặc O(1) nếu dùng set	Ma trận kề nếu cần kiểm tra sự tồn tại cạnh thường xuyên
Duyệt tất cả đỉnh kề của u	O(n)	O(deg(u))	Danh sách kề
Thêm cạnh	O(1)	O(1) (thêm vào cuối danh sách) hoặc O(1) amortized nếu dùng vector	Ma trận kề nếu cần thêm nhiều
Xóa cạnh	O(1)	O(deg(u)) (tìm và xóa) hoặc O(1) nếu dùng set	Ma trần kề nếu cần xóa nhiều
Phù hợp đồ thị có hướng hoặc vô hướng	Cả hai	Cả hai	
Phù hợp đồ thị có trọng số	Lưu trực tiếp trọng số trong ma trận	Lưu cặp {đỉnh, trọng số}	

Khuyến nghị sử dụng	Đồ thị dày $m \approx n^2/2$ $n \leq \sim 5000 – 10000$	Đồ thị thưa $m << n^2$	

Trong đó: n là số đỉnh, m là số cạnh, deg(u) là bậc của đỉnh u.

1.2.4 Dữ liệu thử nghiệm

Sử dụng các bộ dữ liệu mẫu và dữ liệu thực tế Karate Club 34 đỉnh để kiểm chứng tính đúng đắn và trực quan của chương trình. Tạo ra một công cụ mô phỏng mang tính giáo dục cao, có thể sử dụng làm tài liệu hỗ trợ giảng dạy và học tập môn Cấu trúc dữ liệu và giải thuật cho sinh viên Khoa Công nghệ thông tin Trường Đại học Nha Trang.

Chương 2. CƠ SỞ LÝ THUYẾT

2.1 KHÁI NIỆM ĐỒ THỊ

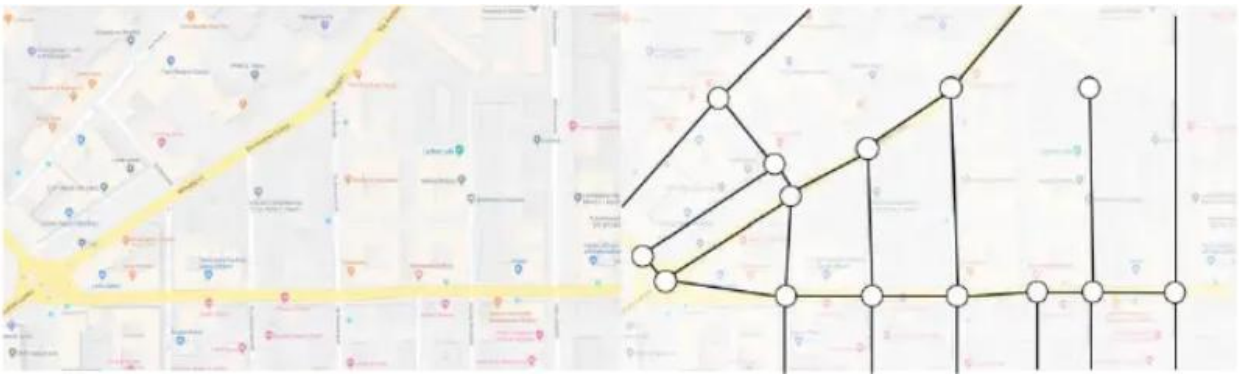
2.1.1 Định nghĩa đồ thị

Khi mô tả một cấu trúc rời rạc gồm các đối tượng và mối liên hệ giữa chúng, ta có thể sử dụng mô hình đồ thị. Mỗi đối tượng sẽ được biểu diễn bởi một đỉnh và mỗi mối liên hệ giữa hai đối tượng sẽ được biểu diễn bởi một cạnh. Chẳng hạn, để minh họa tại sao cần đến các loại đồ thị khác nhau, mô tả bản đồ đường đi bằng đồ thị trong đó đỉnh là các giao lộ và cạnh là các đoạn đường nối các giao lộ. Qua đó, ta thấy rằng đồ thị là một mô hình trừu tượng nhưng rất trực quan và mạnh mẽ, cho phép biểu diễn nhiều loại hệ thống thực tế dưới dạng cấu trúc gồm các điểm và các mối liên hệ giữa chúng.

Đồ thị là một tập các đối tượng được gọi là các đỉnh nối với nhau bởi các cạnh. Một đồ thị G được định nghĩa bởi cặp G = (V, E)[3]*. Trong đó*

V là tập các đỉnh của đồ thị.

E là tập các cạnh của đồ thị.



Hình 2.1. Biểu diễn bản đồ đường đi bằng đồ thị: đỉnh là các giao lộ, cạnh là các con đường

2.1.2 Các loại đồ thị

Đồ thị vô hướng: *Đồ thị vô hướng hoặc đồ thị G là một cặp không có thứ tự G = (V, E). Trong đó:*

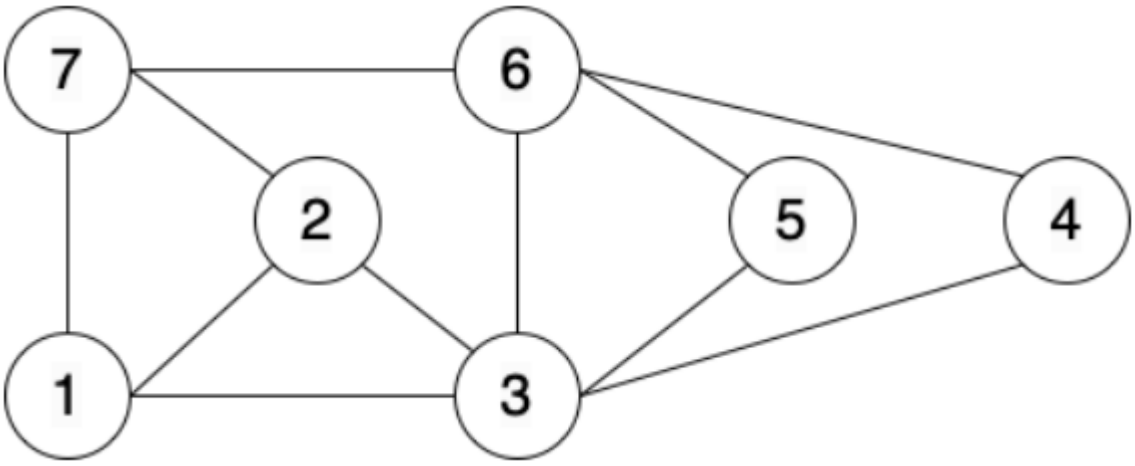
V là tập các đỉnh.

E là tập các cặp không thứ tự chứa các đỉnh phân biệt, được gọi là cạnh. Hai đỉnh thuộc một cạnh được gọi là đỉnh đầu và đỉnh cuối của cạnh đó.

Hay nói cách khác đồ thị chỉ chứa các cạnh vô hướng được gọi là đồ thị vô hướng.

Cạnh vô hướng: Không quan tâm đến hướng và coi hai đỉnh như nhau.

Ví dụ: Xét trong đồ thị vô hướng (Hình 1.2) ta có cạnh (1,2), (2,3).



Hình 2.2. Đồ thị vô hướng

Đồ thị có hướng: *Đồ thị có hướng hay đồ thị G là một cặp có thứ tự G = (V, A), trong đó:*

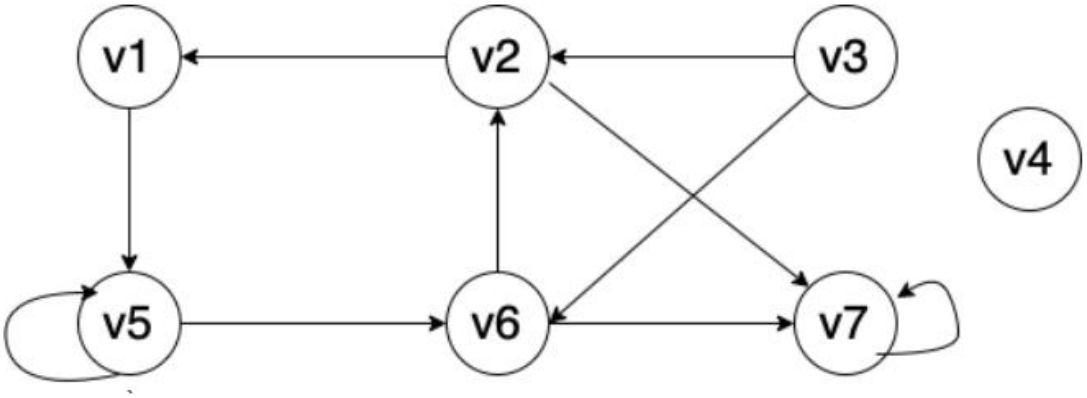
V là tập các đỉnh.

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

Hay nói cách khác đồ thị chỉ chứa các cạnh có hướng được gọi là đồ thị có hướng.

Cạnh có hướng: Là một cặp đỉnh có thứ tự. Trong mỗi cặp có thứ tự đó, đỉnh thứ nhất được gọi là đỉnh đầu, đỉnh thứ hai là đỉnh cuối.

Ví dụ: Xét trong đồ thị có hướng (Hình 1.3) ta có cạnh (v3,v2), cạnh (v2,v1)



Hình 2.3. Đồ thị có hướng

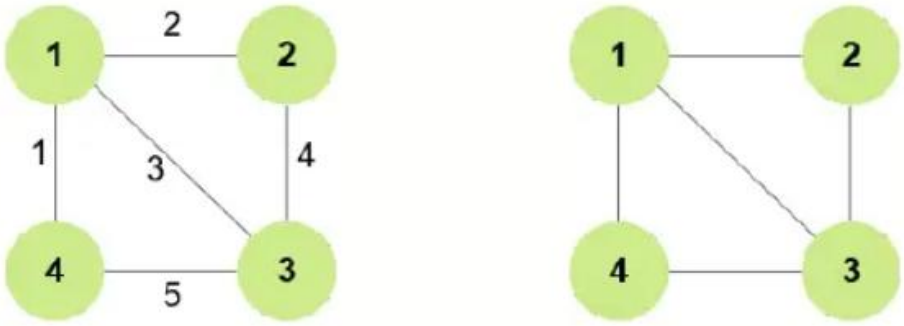
Ngoài ra, chúng ta có thể phân loại đồ thị dựa trên đặc điểm các cạnh có trọng số hoặc không có trọng số.

Đồ thị không có trọng số: *Là đồ thị mà tất cả các cạnh trong đồ thị đều không có trọng số.*

Cạnh không trọng số: Là các cạnh như trên Hình 1.1, 1.2 và 1.3, các cạnh đều có vai trò như nhau.

Đồ thị có trọng số*: Là đồ thị mà tất cả các cạnh trong đồ thị đều có trọng số.*

Cạnh có trọng số: Là cạnh được gán một giá trị thể hiện trọng số của cạnh.



Hình 2.4. Đồ thị có trọng số (trái) và đồ thị không có trọng số (phải)

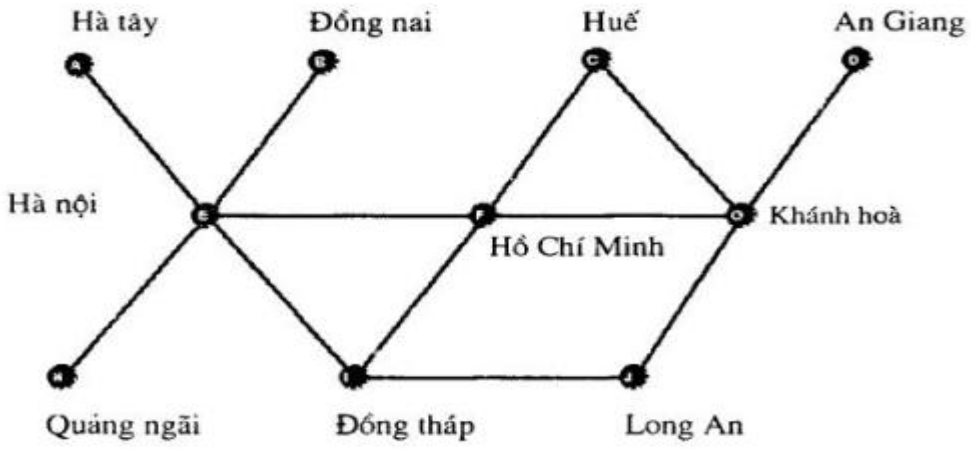
Hơn nữa, chúng ta có thể phân loại đồ thị dựa vào số cạnh nối giữa hai đỉnh xác định đơn đồ thị hay đa đồ thị.

a) Đơn đồ thị

Đơn đồ thị vô hướng G = (V, E) bao gồm V là tập các đỉnh khác rỗng, và E là tập các cặp không có thứ tự gồm hai phần tử khác nhau của V gọi là các cạnh, không có khuyên và không có cạnh song song.

Không có khuyên: tức là không tồn tại cung, nghĩa là không có cung nào đi từ một đỉnh đến chính nó.

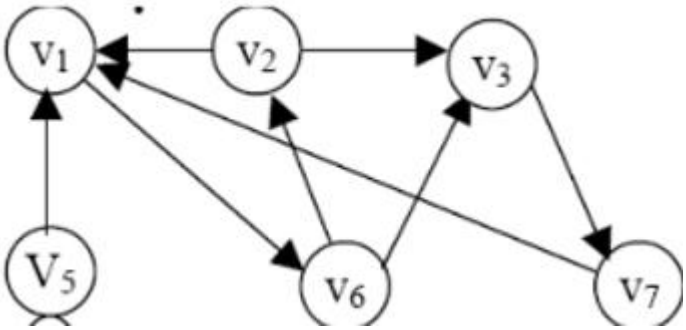
Không có cạnh song song: tức là giữa hai đỉnh, không có nhiều hơn một cung cùng chiều



Hình 2.5. Sơ đồ mạng máy tính

Sơ đồ trên biểu diễn một mạng máy tính gồm các nút tương ứng với các tỉnh thành. Các đường nối giữa các nút là các liên kết truyền thông trong mạng. Khi mô hình hóa dưới dạng đồ thị, ta thu được một đơn đồ thị G = (V, E), trong đó mỗi đỉnh là một vị trí địa lý có thiết bị mạng, và mỗi cạnh là một kết nối vật lý. Do không tồn tại cạnh song song và không có khuyên, mạng được mô hình hóa dưới dạng một *đơn đồ thị vô hướng.*

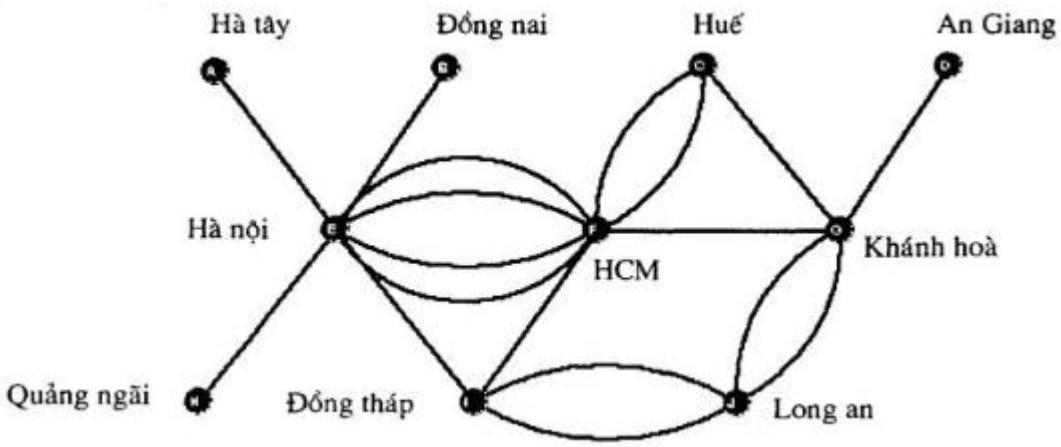
Đơn đồ thị có hướng G = (V, E) bao gồm V là tập các đỉnh khác rỗng, và E là tập các cặp có thứ tự gồm hai phần tử khác nhau của V gọi là các cung, không có khuyên và không có cạnh song song.



Hình 2.6. Đơn đồ thị có hướng

b) Đa đồ thị

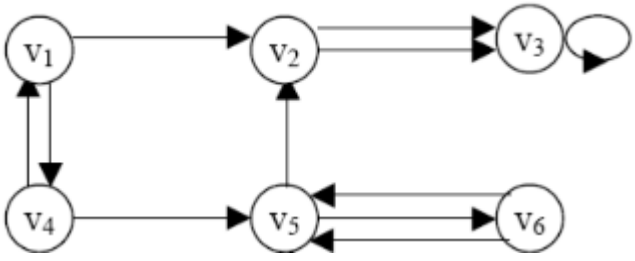
Đa đồ thị vô hướng G = (V, E) bao gồm V là tập các đỉnh khác rỗng và E là tập các cặp không có thứ tự gồm hai phần tử khác nhau của V gọi là các cạnh. Hai cạnh e1, e2 tương ứng với cùng một cặp đỉnh được gọi là cạnh lặp.



Hình 2.7. Sơ đồ mạng máy tính đa kênh thoại

Sơ đồ trên biểu diễn một mạng máy tính gồm các nút tương ứng với các tỉnh thành. Các đường nối giữa các nút là các liên kết truyền thông trong mạng. Khi mô hình hóa dưới dạng đồ thị, ta thu được một đa đồ thị vô hướng G = (V, E), trong đó mỗi đỉnh biểu diễn một vị trí địa lý có thiết bị mạng, còn mỗi cạnh tương ứng với một kênh truyền thông vật lý giữa hai nút. Do giữa một số cặp đỉnh tồn tại nhiều hơn một kênh liên lạc, đồ thị thu được có các cạnh song song, và vì không có kết nối tự quay lại chính nó nên không xuất hiện khuyên**.** Việc mô hình hóa mạng dưới dạng *đa đồ thị vô hướng* cho phép mô tả chính xác cấu trúc đa kênh, thể hiện rõ sự tồn tại của nhiều tuyến truyền song song nhằm đảm bảo băng thông và tính dự phòng của hệ thống.

Đa đồ thị có hướng G = (V, E) bao gồm V là tập các đỉnh khác rỗng và E là tập các cặp có thứ tự gồm hai phần tử khác nhau của V gọi là các cung. Hai cạnh v3, v5 tương ứng với cùng một cặp đỉnh được gọi là cung lặp.



Hình 2.8. Đa đồ thị có hướng

2.1.3 Các thuật ngữ

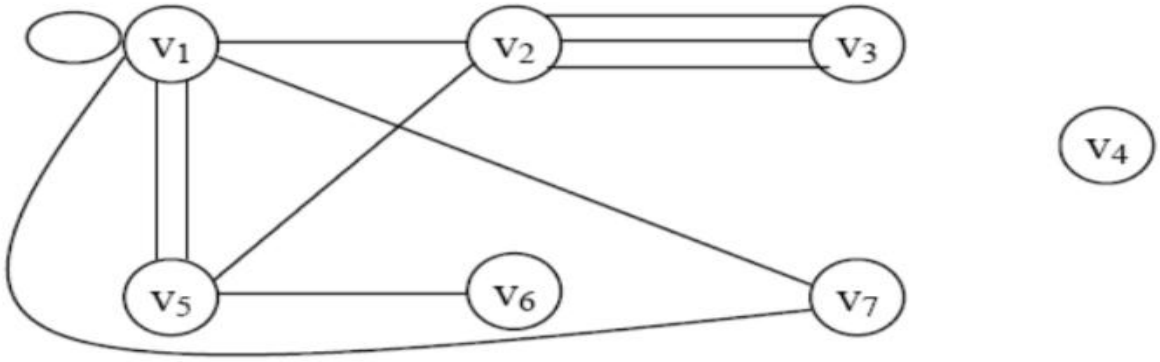
Trong mục này, chúng ta sẽ tìm hiểu và trình bày một số khái niệm nền tảng của lĩnh vực lý thuyết đồ thị. Những khái niệm này đóng vai trò quan trọng giúp ta mô tả, phân tích và hiểu được cấu trúc của các loại đồ thị khác nhau. Để bắt đầu, chúng ta sẽ

tập trung vào nhóm thuật ngữ dùng để mô tả các đỉnh và các cạnh hai thành phần cơ bản cấu thành nên một đồ thị.

Định nghĩa 2.1.3.1. *Hai đỉnh còn u và v của đồ thị vô hướng G được gọi là kề nhau nếu (u, v) là cạnh của đồ thị G. Nếu e = (u,v) là cạnh của đồ thị thì ta nói cạnh này là cạnh liên thuộc với hai đỉnh u và v, hoặc cũng nói là cạnh e là nối đỉnh u và v, đồng thời, các đỉnh u và v sẽ được gọi là các đỉnh đầu của cạnh (u,v)*[5].

Để đo lường số lượng cạnh liên thuộc xuất phát từ một đỉnh, ta giới thiệu định nghĩa sau.

Định nghĩa 2.1.3.2. *Ta gọi bậc của đỉnh v trong đồ thị vô hướng là số cạnh liên thuộc với nó và sẽ ký hiệu là deg(v)*[5] .



Hình 2.9. Bậc của đỉnh trong đồ thị vô hướng G

Ta xét đồ thị trong Hình 1.9, ta có

$deg(v_1) = 7$, $deg(v_2) = 5$, $deg(v_3) = 3$,

$deg(v_4) = 0$, $deg(v_5) = 4$, $deg(v_6) = 1$, $deg(v_7) = 2$.

Đỉnh bậc 0 gọi là *đỉnh cô lập.* Đỉnh bậc 1 được gọi là *đỉnh treo.* Trong đồ thị Hình 1.9 đỉnh v_6 là đỉnh treo, đỉnh v_4 là đỉnh cô lập. Từ đó ta có tính chất bậc của đỉnh.

Định lý 2.1.3.1. *Giả sử G = (V, E) là đồ thị vô hướng với m cạnh*[5]*. Khi đó*

$$2m = \sum_{v \in V} \deg(v)$$

Chứng minh. Rõ ràng mỗi cạnh *e = (u, v)* được tính một lần trong *deg(u)* và một lần trong *deg(v).*Từ đó suy ra tổng tất cả các bậc của các đỉnh bằng hai lần số cạnh.

Hệ quả. *Trong đồ thị vô hướng, số đỉnh bậc lẻ (nghĩa là có bậc là số lẻ) là một số chẵn*[5].

Chứng minh. Thực vậy, gọi O và U tương ứng là tập đỉnh bậc lẻ và tập đỉnh bậc chẵn của đồ thị, ta có

$$2m = \sum_{v \in V} \deg(v) = \sum_{v \in O} \deg(v) + \sum_{v \in U} \deg(v)$$

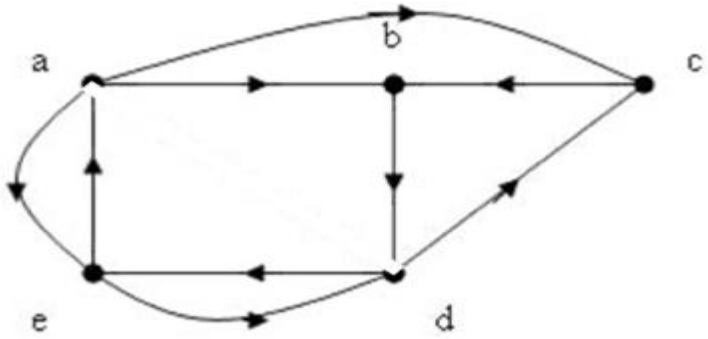
Do deg(v) là chẵn với v là đỉnh trong U nên tổng thứ nhất ở trên là số chẵn. Suy ra tổng thứ hai (chính là tổng bậc của các đỉnh bậc lẻ) cũng phải là số chẵn. Do tất cả các số hạng của nó là số lẻ, nên tổng này phải gồm một số chẵn các số hạng. Vì vậy, số đỉnh bậc lẻ phải là số chẵn.

Tiếp theo, ta sẽ xem xét các khái niệm mang tính tương tự nhưng áp dụng cho trường hợp đồ thị có hướng.

Định nghĩa 2.1.3.3. *Nếu e = (u,v) là cung của đồ thị có hướng G thì ta nói hai đỉnh u và v là kề nhau, và nói cung (u,v) nối đỉnh u với đỉnh v hoặc cũng nói cung này là đi ra khỏi đỉnh u và vào đỉnh v. Đỉnh u(v) sẽ được gọi là đỉnh đầu (cuối) của cung (u,v)*[5].

Từ khái niệm bậc của đỉnh, khi xét đến đồ thị có hướng, ta phân biệt thêm hai dạng bậc là bán bậc ra và bán bậc vào.

Định nghĩa 2.1.3.4. *Ta gọi bán bậc ra (bán bậc vào) của đỉnh v trong đồ thị có hướng là số cung của đồ thị đi ra khỏi nó (đi vào nó) và ký hiệu là* $deg^+(v)$ *(*$deg^-(v)$*)*[5].



Hình 2.10. Đồ thị có hướng G

Ta xét đồ thị trong Hình 1.10, ta có

$deg^-(a) = 1$, $deg^-(b) = 2$, $deg^-(c) = 2$, $deg^-(d) = 2$, $deg^-(e) = 2$,

$deg^+(a) = 3$, $deg^+(b) = 1$, $deg^+(c) = 1$, $deg^+(d) = 2$, $deg^+(e) = 2$.

Định lý 2.1.3.2. *Cho G = (V, E) là một đồ thị có hướng*[5]*. Khi đó:*

$$\sum_{v \in V} \deg + (v) = \sum_{v \in V} \deg - (v) = |E|$$

Rất nhiều tính chất của đồ thị có hướng không phụ thuộc vào hướng trên các cung của nó. Vì vậy, trong nhiều trường hợp, sự thuận tiện hơn nếu ta bỏ qua hướng trên các cung của đồ thị. Đồ thị vô hướng được tạo bằng cách bỏ qua hướng trên các cung được gọi là *đồ thị vô hướng tương ứng* với đồ thị có hướng đã cho[5].

2.2 PHƯƠNG PHÁP BIỂU DIỄN ĐỒ THỊ

Biểu diễn đồ thị là cơ sở để định nghĩa đồ thị như một cấu trúc dữ liệu trong các ngôn ngữ lập trình, từ đó có thể triển khai các thuật toán, chương trình xử lý tính toán dựa trên đồ thị trong các ứng dụng thực tế.

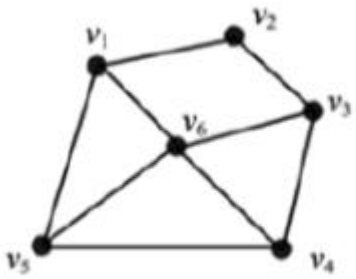
Để lưu trữ đồ thị và thực hiện các thuật toán toán học rời rạc trên máy tính, cần phải biến khái niệm đồ thị toán học thành một cấu trúc dữ liệu cụ thể mà các ngôn ngữ lập trình có thể xử lý. Việc chọn cách biểu diễn đồ thị phù hợp không chỉ ảnh hưởng trực tiếp đến hiệu quả bộ nhớ, thời gian chạy của các giải thuật mà còn quyết định tính khả thi khi triển khai vào các ứng dụng thực tế như phân tích mạng xã hội, định tuyến giao thông, sinh tin học, tối ưu hóa chuỗi cung ứng hay học máy trên dữ liệu đồ thị.

2.2.1 Ma trận kề

2.2.1.1 Ma trận kề của đồ thị không trọng số

Giả sử ta xét đơn đồ thị vô hướng *G = (V, E)* không có trọng số, với tập đỉnh $V = \{v_1, v_2, \ldots, v_n\}$, tập cạnh $E = \{e_1, e_2, \ldots, e_m\}$. Ta gọi ma trận kề $A = [a_{ij}]$, $i, j = 1, 2, \ldots n$ của đồ thị G. Một ma trận vuông cấp *n* được xác định như sau.

$$\begin{cases} a_{ij} = 1, \ \textit{nếu tồn cạnh } (v_i, v_j) \\ a_{ij} = 0, \ \textit{nếu không tồn tại cạnh } (v_i, v_j) \end{cases} i, j = 1,2,\ldots, n.$$

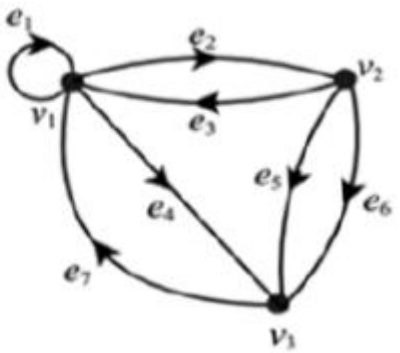


$$A_G = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$

Hình 2.11. Đồ thị vô hướng không có trọng số (trái) và ma trận kề của đồ thị vô hướng không có trọng số G (phải)

Ta thấy, trong Hình 1.11 ma trận A_G là ma trận kề biểu diễn đồ thị vô hướng G theo các đỉnh được sắp xếp theo thứ tự $v_1, v_2, v_3, v_4, v_5, v_6$.

Ma trận kề của đồ thị có hướng được định nghĩa một cách hoàn toàn tương tự.



$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 2 \\ 1 & 0 & 0 \end{bmatrix}$$

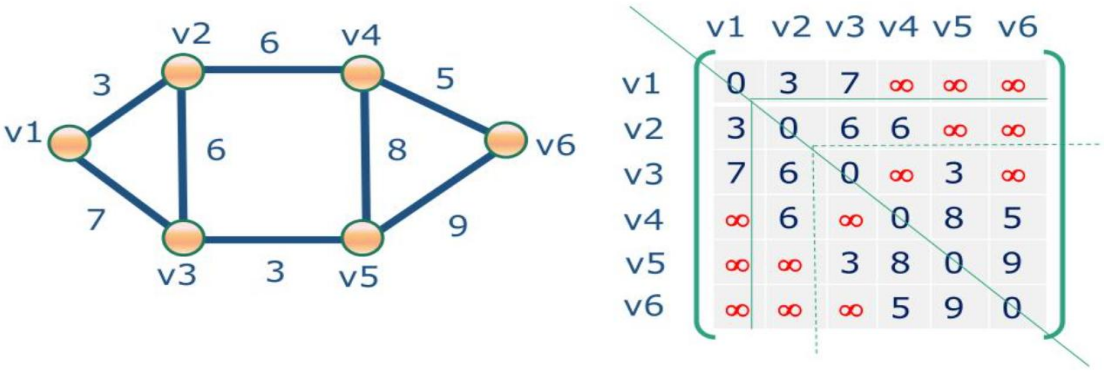
Hình 2.12. Đồ thị có hướng không có trọng số (trái) và ma trận kề của đồ thị có hướng không có trọng số G

Ta thấy, trong Hình 1.12 ma trận kề biểu diễn đồ thị có hướng theo các đỉnh được sắp xếp theo thứ tự v_1, v_2, v_3.

2.2.1.2 Ma trận kề của đồ thị có trọng số

Giả sử ta xét đơn đồ thị vô hướng $G = (V, E)$ có trọng số w_{ij}, với tập đỉnh $V = \{v_1, v_2,\ldots, v_n\}$, tập cạnh $E = \{e_1, e_2,\ldots, e_m\}$. Ta gọi ma trận kề $A = [a_{ij}]$, $i, j = 1, 2,\ldots$ n của đồ thị G. Một ma trận vuông cấp *n* được xác định như sau.

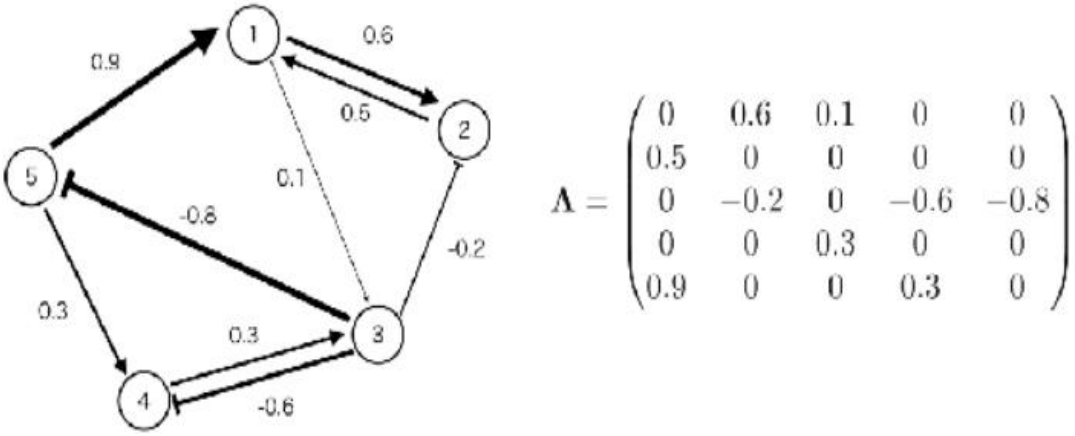
$$\begin{cases} a_{ij} = w_{ij}, \text{ nếu tồn tại cạnh } (v_i, v_j) \text{ với trọng số } w_{ij} \\ a_{ij} = 0 \text{ hoặc } \infty, \text{ nếu không tồn tại cạnh } (v_i, v_j) \end{cases} \quad i, j = 1,2,\ldots, n.$$



Hình 2.13. Đồ thị vô hướng có trọng số (trái) và ma trận kề của đồ thị vô hướng có trọng số G (phải)

Ta thấy, trong Hình 2.13, ma trận A_G là ma trận kề biểu diễn đồ thị vô hướng có trọng số G theo các đỉnh được sắp xếp theo thứ tự $v_1, v_2, v_3, v_4, v_5, v_6$. Mỗi phần tử a_{ij} của ma trận tương ứng với trọng số của cạnh nối giữa hai đỉnh v_i và v_j trong trường hợp không tồn tại cạnh, giá trị a_{ij} được gán bằng 0 hoặc ∞.

Ma trận kề của đồ thị có hướng có trọng số được định nghĩa một cách hoàn toàn tương tự.



Hình 2.14. Đồ thị có hướng có trọng số (trái) và ma trận kề của đồ thị có hướng có trọng số G (phải)

Ta thấy, trong Hình 2.14, ma trận A là ma trận kề biểu diễn đồ thị có hướng có trọng số G theo các đỉnh được sắp xếp theo thứ tự $v_1, v_2, v_3, v_4, v_5, v_6$. Mỗi phần tử a_{ij} của ma trận tương ứng với trọng số của cung có hướng từ đỉnh v_i đến đỉnh v_j. Trong trường hợp không tồn tại cung từ v_i đến v_j, giá trị a_{ij} được gán bằng 0. Trong trường hợp tồn tại các cung ngược chiều có trọng số khác nhau.

2.2.1.3 Các tính chất của ma trận kề:

1) Ma trận kề của đồ thị vô hướng là ma trận đối xứng, nghĩa là

$$a[i, j] = a[j, i], i, j = 1, 2, \ldots n.$$

2) Ma trận kề của đồ thị có hướng không phải là ma trận đối xứng.
3) Tổng các phần tử trên dòng i (cột j) của ma trận kề chính bằng bậc của đỉnh v_i.

a) Ưu điểm

Các phép toán cơ bản như thêm cạnh, xóa cạnh và kiểm tra sự tồn tại của cạnh giữa hai đỉnh i và j được thực hiện hiệu quả về mặt thời gian. Khi đồ thị có nhiều cạnh, ma trận kề thường là lựa chọn tốt vì nó giúp tiết kiệm bộ nhớ và thực hiện các thao tác nhanh chóng.

b) Nhược điểm

Ma trận kề cần lưu trữ thông tin về tất cả các cạnh giữa các đỉnh, dẫn đến việc tiêu thụ một lượng lớn bộ nhớ với độ phức tạp $O(n^2)$. Kích thước của ma trận là $n \times n$ (với n là số lượng đỉnh), dẫn đến một tải nặng về mặt bộ nhớ đặc biệt khi đồ thị lớn.

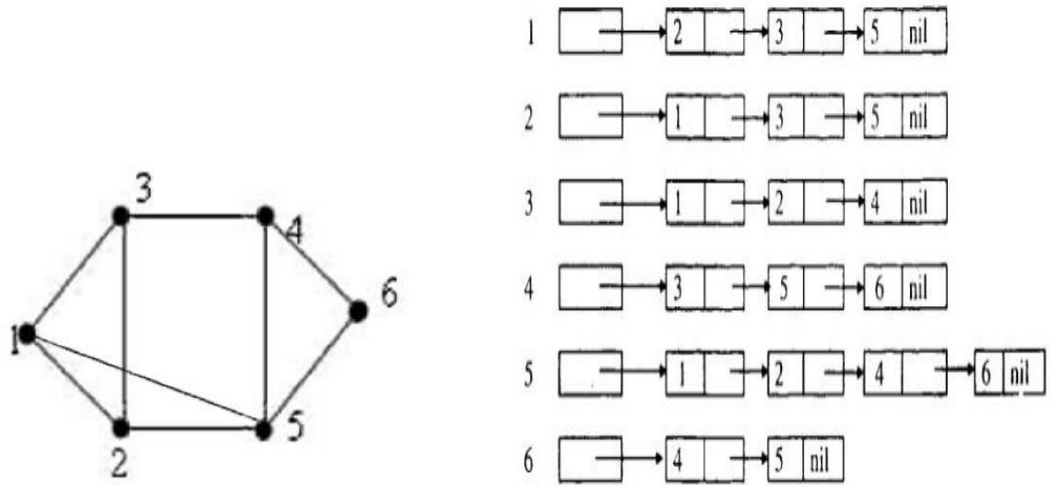
2.2.2 Danh sách kề

Trong quá trình biểu diễn và xử lý đồ thị, danh sách kề là một trong những cấu trúc dữ liệu được sử dụng phổ biến nhờ tính hiệu quả về bộ nhớ và khả năng truy xuất nhanh các đỉnh lân cận. Thay vì lưu trữ toàn bộ ma trận kích thước $n \times n$, danh sách kề biểu diễn mỗi đỉnh bằng một danh sách các đỉnh mà nó có cạnh nối tới.

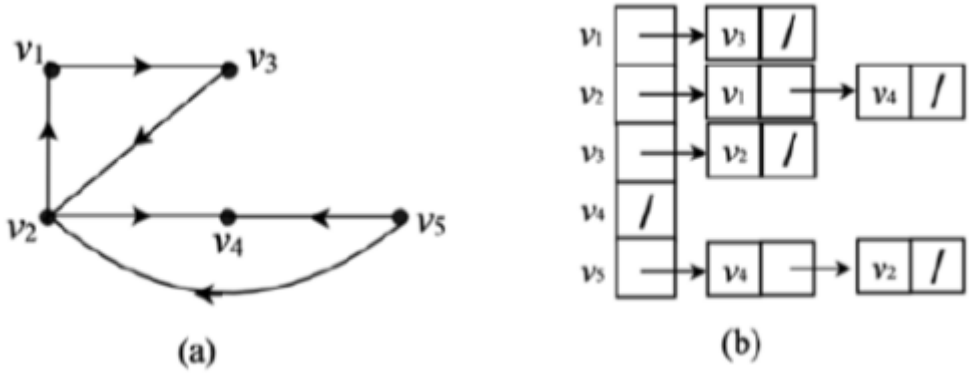
Tập cạnh của một đồ thị có thể được xác định bởi một tập các danh sách các đỉnh kề của mỗi đỉnh của đồ thị. Từ đó chúng ta có thể dùng danh sách (như là danh sách liên kết trong các ngôn ngữ lập trình) để biểu diễn đồ thị. Tập các đỉnh kề của một đỉnh của đồ thị được định nghĩa như sau.

Danh sách kề là biểu diễn một đồ thị dưới dạng một mảng các danh sách liên kết. Trong đó, chỉ số mảng đại diện cho đỉnh của đồ thị và các phần tử trong danh sách liên kết của đỉnh đó là các đỉnh có kết nối với đỉnh đó. Với mỗi đỉnh v, ta lưu trữ danh sách các đỉnh kề với nó, kí hiệu là Ke(v), nghĩa là

$$Ke(v) = \{ u \in V : (v, u) \in E\}$$[5]



Hình 2.15. Danh sách kề (phải) của đồ thị vô hướng (trái)



Hình 2.16. Danh sách kề (phải) của đồ thị có hướng (trái)

Những thao tác cơ bản thường gặp khi xử lý đồ thị

- incidentEdges(v): duyệt các đỉnh kề của đỉnh v.
- arcAdjacent(v, w): trả lại giá trị khi và chỉ khi hai đỉnh v, w là kề nhau.
- insertVertex(z): Bổ sung đỉnh z.
- insertEdge(v, w, e): Bổ sung cạnh e = (u, w).
- removeVertex(v): Loại bỏ đỉnh v.
- removeEdge(e): Loại bỏ cạnh e.

Thao tác	*Danh sách cạnh*	*Danh sách kề*	*Ma trận kề*
Bộ nhớ	$n + m$	$n + m$	n^2
incidentEdges(v)	m	deg(v)	n
areAdjacent(v, w)	m	min(deg(v), deg(w))	1
insertVertex(z)	1	1	n^2
insertEdge(v, w, e)	1	1	1
removeVertex(v)	m	deg(v)	n^2
removeEdge(e)	1	1	1

Hình 2.17. Bảng đánh giá bộ nhớ cần sử dụng và thời gian thực hiện, với n đỉnh và m cạnh

Việc đưa ra đánh giá thời gian đối với các loại đồ thị khác được thực hiện hoàn toàn tương tự.

a) Ưu điểm

So với ma trận kề, biểu diễn đồ thị dưới dạng danh sách kề giúp tiết kiệm bộ nhớ hơn, chỉ tốn $O(n + m)$ thay vì ma trận kề tốn bộ nhớ lưu trữ $O(n^2)$. Ta sẽ thấy rõ điều này khi biểu diễn đồ thị có số lượng lớn đỉnh nhưng có ít số cạnh kết nối. Việc duyệt các đỉnh kề với một đỉnh nào đó cũng cực kỳ nhanh chóng do mỗi đỉnh chỉ kết nối tới các đỉnh kề với nó.

b) Nhược điểm

Do sử dụng danh sách liên kết, việc kiểm tra 2 đỉnh bất kỳ có kết nối hay không cần phải duyệt tuần tự từ node head, sẽ chậm hơn so với việc kiểm tra nếu cài đặt bằng ma trận kề.

2.3 NGÔN NGỮ LẬP TRÌNH PYTHON

2.3.1 Giới thiệu về ngôn ngữ Python

Python là một ngôn ngữ lập trình bậc cao, được Guido van Rossum phát triển và ra mắt lần đầu vào năm 1991. Ngôn ngữ này được thiết kế với tiêu chí đơn giản, dễ đọc và dễ học, giúp người mới bắt đầu có thể tiếp cận nhanh chóng nhưng đồng thời vẫn đủ mạnh mẽ để đáp ứng các nhu cầu lập trình chuyên sâu.

Python hỗ trợ nhiều mô hình lập trình như lập trình hướng đối tượng (OOP), lập trình thủ tục và lập trình hàm. Nhờ cú pháp rõ ràng và cấu trúc linh hoạt, Python trở

thành lựa chọn phổ biến trong giáo dục, nghiên cứu khoa học và phát triển phần mềm hiện đại.

Hình 2.18. Ngôn ngữ lập trình Python

Bên cạnh những đặc điểm nổi bật về cú pháp đơn giản và khả năng dễ tiếp cận, Python còn được đánh giá cao nhờ tính linh hoạt và khả năng ứng dụng rộng rãi trong nhiều lĩnh vực công nghệ hiện đại. Python là ngôn ngữ đa nền tảng, hoạt động tốt trên hầu hết các hệ điều hành phổ biến như Windows, Linux và macOS mà không cần thay đổi nhiều về mã nguồn. Điều này giúp các nhà phát triển dễ dàng triển khai ứng dụng trong nhiều môi trường khác nhau.

Ngoài ra, Python sở hữu một cộng đồng người dùng và nhà phát triển vô cùng lớn mạnh. Cộng đồng này liên tục đóng góp, xây dựng và cải tiến các thư viện, framework cũng như tài liệu hướng dẫn, giúp Python ngày càng hoàn thiện và phát triển. Sự hỗ trợ dồi dào từ cộng đồng cũng góp phần giúp người học dễ dàng tiếp cận, tìm kiếm tài liệu và giải quyết các vấn đề trong quá trình lập trình.

Python còn được tích hợp trong nhiều công cụ và nền tảng khoa học như Jupyter Notebook, Anaconda, giúp tối ưu hóa cho việc phân tích dữ liệu, thực nghiệm và mô phỏng. Đây cũng là lý do Python trở thành công cụ quen thuộc của các nhà nghiên cứu, sinh viên và những người làm việc trong lĩnh vực khoa học máy tính, trí tuệ nhân tạo và xử lý dữ liệu.

2.3.2 Thư viện sử dụng chính trong Python

a) Tkinter

Tkinter là thư viện giao diện đồ họa tiêu chuẩn đi kèm với Python, cho phép người dùng xây dựng các ứng dụng có cửa sổ, nút bấm, hộp văn bản và nhiều thành phần trực quan khác. Tkinter cung cấp cách tiếp cận đơn giản và dễ sử dụng, phù hợp cho cả người mới bắt đầu lẫn những dự án yêu cầu giao diện nhẹ, không phức tạp. Nhờ khả năng tích hợp trực tiếp trong Python mà không cần cài thêm thư viện mở rộng, Tkinter trở thành lựa chọn phổ biến để tạo ra các ứng dụng minh họa, công cụ hỗ trợ học thuật hoặc chương trình có giao diện đơn giản.

Cài đặt: Không cần cài đặt riêng, tích hợp sẵn trong Python.

b) NetworkX

NetworkX là một thư viện mạnh mẽ dùng để tạo, phân tích và thao tác với các cấu trúc đồ thị trong Python. Thư viện hỗ trợ nhiều loại đồ thị khác nhau như đồ thị có hướng, vô hướng, đa đồ thị, cùng hàng loạt thuật toán từ cơ bản đến nâng cao như tìm đường đi ngắn nhất, phân tích trung tâm, phát hiện cộng đồng. NetworkX đặc biệt hữu ích trong các lĩnh vực như khoa học dữ liệu, phân tích mạng xã hội, mô hình hóa hệ thống và nghiên cứu thuật toán nhờ khả năng xử lý linh hoạt và cú pháp thân thiện.

Cài đặt: pip install networkx

Trang chủ: https://networkx.org

c) Matplotlib

Matplotlib là thư viện vẽ đồ thị 2D phổ biến nhất trong Python, cung cấp giao diện tương tự MATLAB để tạo biểu đồ trực quan như biểu đồ đường, cột, phân tán, histogram và nhiều dạng hình học khác. Thư viện cho phép tùy chỉnh gần như toàn bộ thành phần của biểu đồ như màu sắc, kích thước, tiêu đề, nhãn và chú thích. Trong các bài toán trực quan hóa đồ thị, Matplotlib thường được kết hợp với NetworkX để hiển thị các nút và cạnh rõ ràng, dễ quan sát và phục vụ phân tích dữ liệu.

Cài đặt: pip install matplotlib

Trang chủ: https://matplotlib.org

d) Graphviz

Graphviz là một công cụ mạnh mẽ hỗ trợ trực quan hóa đồ thị bằng các thuật toán bố cục tự động như dot, neato, circo, fdp, giúp hiển thị các cấu trúc đồ thị rõ ràng, gọn gàng và mang tính chuyên nghiệp cao. Khi kết hợp với Python thông qua các thư viện như PyGraphviz hoặc python-graphviz, Graphviz cho phép tạo ra hình ảnh đồ thị chất lượng cao, dễ xuất ra file và sử dụng trong báo cáo hoặc nghiên cứu. Đây là một công cụ quan trọng trong phân tích cấu trúc mạng và các bài toán liên quan đến thuật toán đồ thị.

Cài đặt: tải từ https://graphviz.org/download

Python binding: pip install pygraphviz hoặc pip install graphviz

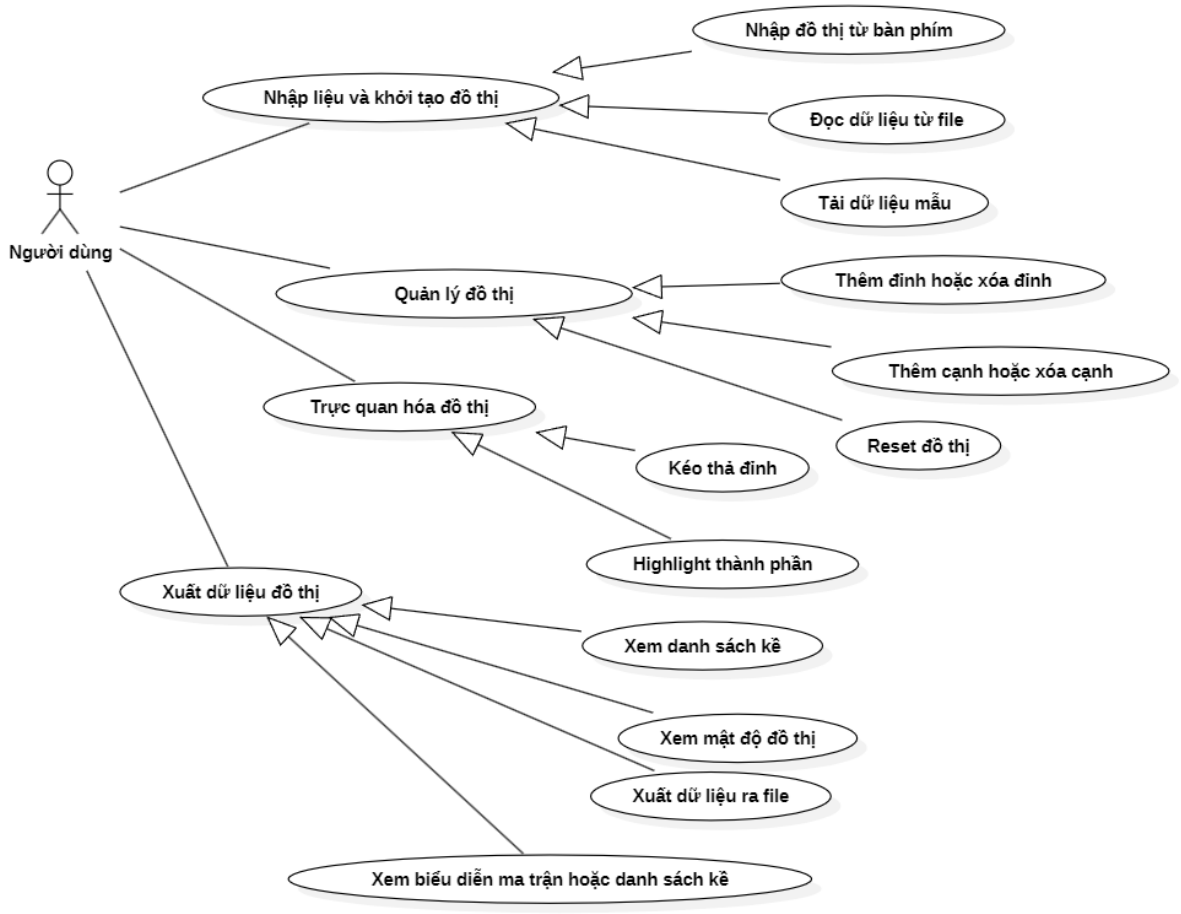
Chương 3. PHÂN TÍCH THIẾT KẾ HỆ THỐNG

3.1 SƠ ĐỒ USE-CASE

3.1.1 Các tác nhân

Trong hệ thống này, chỉ có một tác nhân duy nhất là người dùng. Người sử dụng có thể tự khởi tạo đồ thị bằng cách nhập dữ liệu thủ công từ bàn phím với tính năng cập nhật thời gian thực hoặc tải lên các tệp tin chứa cấu trúc đồ thị có sẵn. Tiếp đến, người dùng có thể linh hoạt tùy chỉnh các đặc tính của đồ thị như lựa chọn chế độ có hướng hoặc vô hướng, có trọng số hoặc không trọng số, và thực hiện các thao tác quản trị như thêm hoặc xóa các đỉnh và cạnh. Đặc biệt, người dùng còn có thể tương tác trực tiếp trên không gian biểu đồ để thay đổi bố cục bằng cách kéo thả hoặc highlight các thành phần quan trọng, đồng thời theo dõi các biểu diễn dữ liệu đồng bộ như ma trận kề, danh sách kề và mật độ đồ thị để có cái nhìn trực quan và phân tích sâu sắc hơn về cấu trúc dữ liệu đang làm việc.

3.1.2 Các Use-Case



Hình 3.1. Sơ đồ Use-Case

Sơ đồ mô tả các tương tác của người dùng là tác nhân duy nhất với hệ thống thông qua 4 nhóm chức năng chính:

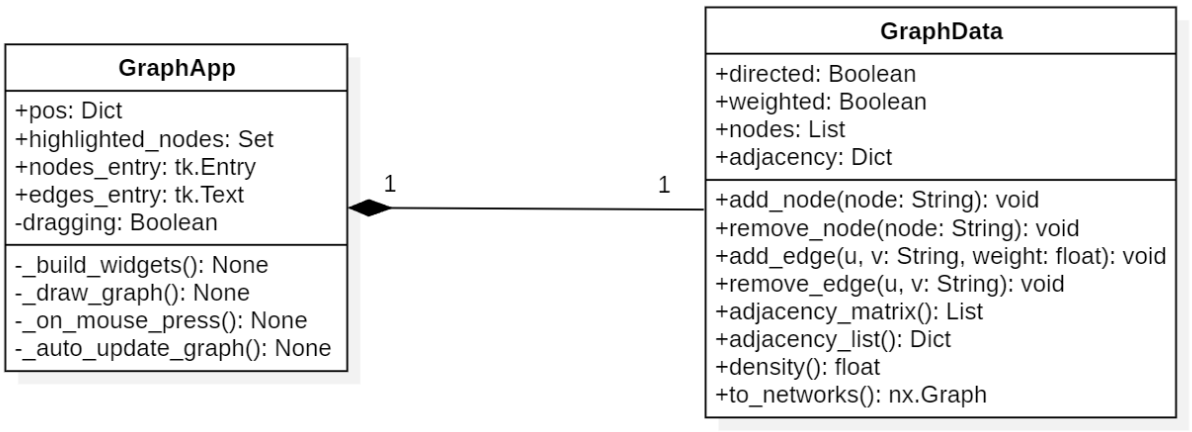
- Nhập liệu và khởi tạo đồ thị: Cho phép người dùng tạo dữ liệu đầu vào bằng nhiều cách khác nhau như nhập trực tiếp từ bàn phím, đọc dữ liệu từ tệp tin file .txt hoặc sử dụng các mẫu dữ liệu có sẵn như Karate Club.
- Quản lý đồ thị: Cung cấp các công cụ để thay đổi cấu trúc đồ thị hiện tại bao gồm thêm hoặc xóa đỉnh, thêm hoặc xóa cạnh và chức năng reset để đưa đồ thị về trạng thái rỗng ban đầu.
- Trực quan hóa đồ thị: Hỗ trợ tương tác trực tiếp trên giao diện đồ họa thông qua việc kéo thả để điều chỉnh vị trí các đỉnh và đánh dấu highlight các thành phần quan trọng.
- Xuất dữ liệu và xem thông tin: Giúp người dùng khai thác thông tin từ đồ thị như xem biểu diễn dưới dạng ma trận kề hoặc danh sách kề, kiểm tra mật độ đồ thị và trích xuất toàn bộ kết quả ra tệp tin để lưu trữ.

3.2 XÂY DỰNG CÁC LỚP ĐỐI TƯỢNG

3.2.1 Xác định các lớp đối tượng

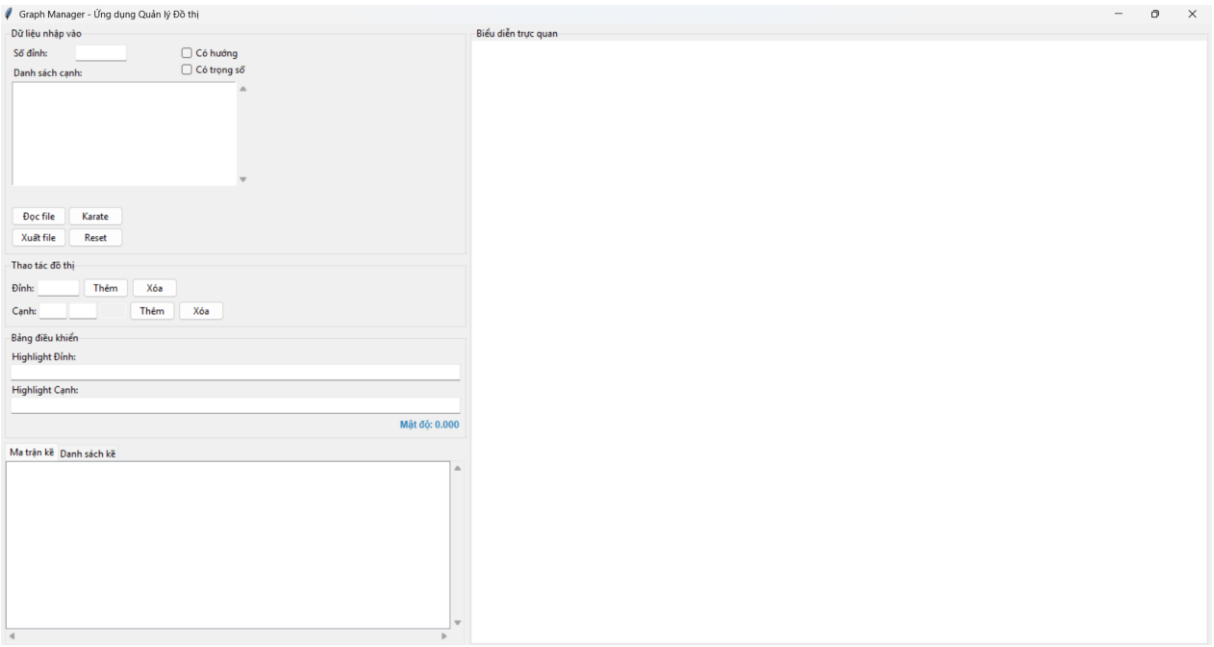
- GraphData: Lớp dữ liệu và logic đóng vai trò là Model quản lý toàn bộ cấu trúc và các phép toán trên đồ thị.
- GraphApp: Lớp giao diện và điều khiển kế thừa từ lớp tk.Tk của thư viện Tkinter. đóng vai trò là View và Controller.

3.2.2 Thiết kế lớp chi tiết



Hình 3.2. Sơ đồ lớp đối tượng

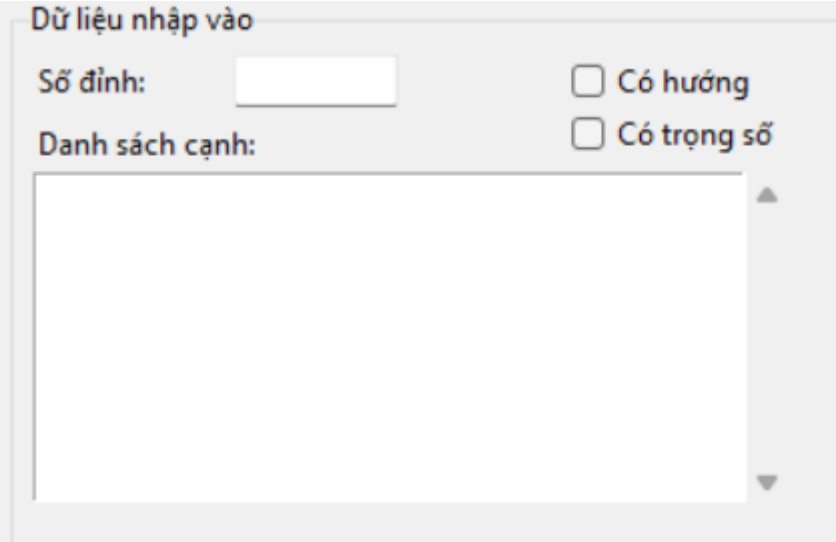
3.3 GIAO DIỆN NGƯỜI DÙNG



Hình 3.3. Giao diện ban đầu của người dùng

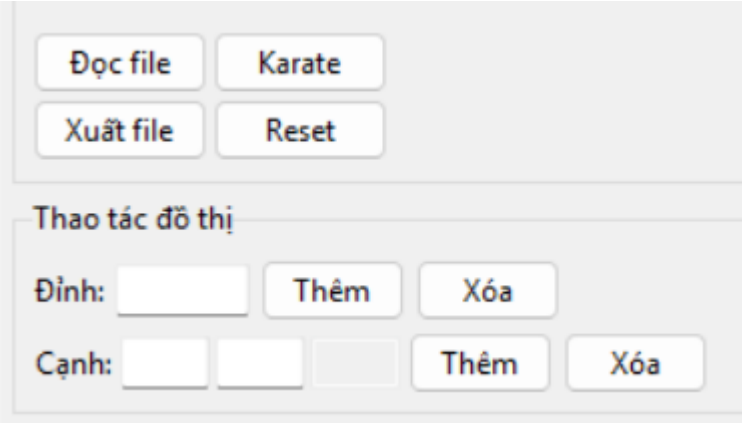
Hệ thống mô phỏng đồ thị được xây dựng nhằm hỗ trợ người dùng trong việc nhập liệu, quản lý, trực quan hóa và xuất dữ liệu đồ thị phục vụ cho học tập và nghiên cứu.

- Nhập dữ liệu đồ thị: Hệ thống cho phép người dùng nhập dữ liệu đồ thị trực tiếp từ bàn phím thông qua việc khai báo số đỉnh và danh sách cạnh. Ngoài ra, người dùng có thể đọc dữ liệu đồ thị từ file hoặc sử dụng dữ liệu mẫu có sẵn. Hệ thống hỗ trợ cả đồ thị có hướng hoặc không hướng, có trọng số hoặc không trọng số.
- Quản lý và chỉnh sửa đồ thị: Người dùng có thể thực hiện các thao tác quản lý đồ thị như thêm hoặc xóa đỉnh, thêm hoặc xóa cạnh, cũng như reset đồ thị để đưa hệ thống về trạng thái ban đầu.
- Trực quan hóa đồ thị: Hệ thống hiển thị đồ thị một cách trực quan trên giao diện người dùng, cho phép kéo thả các đỉnh để thay đổi vị trí hiển thị. Đồng thời, người dùng có thể làm nổi bật highlight các đỉnh hoặc cạnh nhằm hỗ trợ quá trình quan sát và phân tích.
- Hiển thị thông tin đồ thị: Hệ thống cung cấp các thông tin liên quan đến đồ thị như danh sách kề, ma trận kề và mật độ đồ thị, giúp người dùng dễ dàng theo dõi và đánh giá cấu trúc của đồ thị.
- Xuất dữ liệu đồ thị: Người dùng có thể xuất dữ liệu đồ thị ra file để phục vụ cho việc lưu trữ, chia sẻ hoặc sử dụng cho các mục đích khác.



Hình 3.4. Giao diện nhập liệu đồ thị

Khu vực này nằm ở phía bên trái giao diện, cho phép người dùng khởi tạo đồ thị ban đầu. Người dùng có thể nhập số đỉnh để xác định quy mô của đồ thị, đồng thời nhập danh sách cạnh theo định dạng quy ước, chẳng hạn như *u v đối với đồ thị không trọng số* hoặc *u v w đối với đồ thị có trọng số*. Ngoài ra, cho phép người dùng lựa chọn đồ thị có hướng hoặc vô hướng, cũng như xác định đồ thị có sử dụng trọng số cạnh hay không, từ đó phù hợp với từng bài toán và mục đích phân tích khác nhau.



Hình 3.5. Giao diện nút các chức năng

Các nút chức năng hỗ trợ thao tác nhanh với dữ liệu đồ thị:

- Đọc file: Nhập dữ liệu đồ thị từ file có sẵn.
- Xuất file: Lưu đồ thị hiện tại ra file.
- Karate: Tải nhanh đồ thị mẫu Karate Club để minh họa.
- Reset: Xóa toàn bộ dữ liệu và đưa ứng dụng về trạng thái ban đầu.

Cho phép chỉnh sửa đồ thị trực tiếp mà không cần nhập lại toàn bộ dữ liệu:

- Thao tác với đỉnh:
  - Thêm đỉnh mới.
  - Xóa đỉnh đã tồn tại.
- Thao tác với cạnh:
  - Thêm cạnh giữa hai đỉnh (kèm trọng số nếu có).
  - Xóa cạnh bất kỳ trong đồ thị đã tồn tại.

Hình 3.6. Giao diện bảng điều khiển

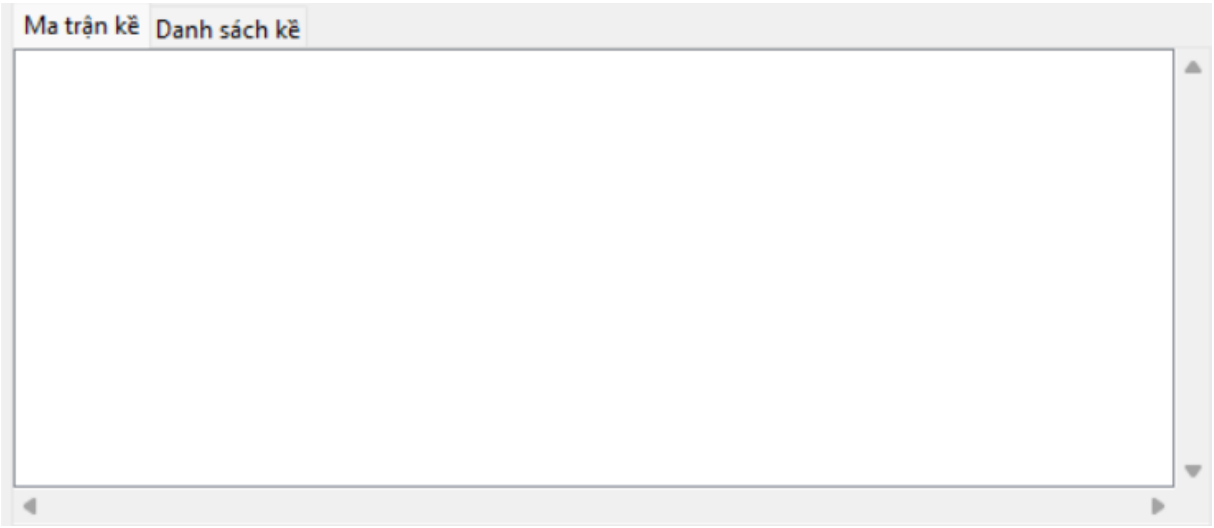
Khu vực này hỗ trợ quan sát và phân tích đồ thị:

- Highlight Đỉnh: Làm nổi bật một hoặc nhiều đỉnh được chỉ định.
- Highlight Cạnh: Làm nổi bật các cạnh được chọn.
- Mật độ đồ thị: Hiển thị mật độ kết nối của đồ thị hiện tại, giúp đánh giá mức độ liên kết giữa các đỉnh.

Hình 3.7. Giao diện biểu diễn trực quan

Nằm ở phía bên phải giao diện:

- Hiển thị đồ thị dưới dạng hình vẽ trực quan.
- Các đỉnh và cạnh được bố trí tự động, dễ quan sát.
- Khi thực hiện thêm hoặc xóa hoặc highlight, vùng này sẽ cập nhật ngay lập tức.



Hình 3.8. Giao diện của ma trận kề và danh sách kề

Nằm ở phía dưới bên trái:

- Ma trận kề: Biểu diễn đồ thị dưới dạng bảng ma trận.
- Danh sách kề: Biểu diễn đồ thị dưới dạng danh sách các đỉnh kề.
- Người dùng có thể chuyển đổi giữa hai cách biểu diễn để phục vụ học tập và phân tích thuật toán.

Luồng thao tác chính của người dùng ứng dụng như sau:

- Người dùng nhập số đỉnh và danh sách cạnh (hoặc đọc từ file).
- Chọn loại đồ thị có hướng hoặc vô hướng và có trọng số hoặc không có trọng số.
- Quan sát đồ thị trên vùng biểu diễn trực quan.
- Thực hiện thêm hoặc xóa đỉnh, cạnh hoặc highlight để phân tích.
- Xem dữ liệu tương ứng dưới dạng ma trận kề hoặc danh sách kề.
- Xuất đồ thị ra file nếu cần lưu trữ.

Chương 4. CÀI ĐẶT CHƯƠNG TRÌNH

4.1 CẤU TRÚC SOURCE CODE

Bảng 4.1. Mô tả các module trong hệ thống

Module	Vai trò trong hệ thống	Mô tả
app.py	Controller và View	Thành phần chính điều khiển ứng dụng. Xây dựng giao diện Tkinter, tiếp nhận sự kiện từ người dùng và phối hợp giữa Model và đồ họa.
graph_data.py	Model	Quản lý cấu trúc dữ liệu cốt lõi của đồ thị. Chứa lớp GraphData để lưu trữ danh sách đỉnh, ma trận kề và thực hiện các phép toán logic trên đồ thị.
graph_io.py	I/O Utility	Nhập hoặc xuất dữ liệu chứa các hàm đọc file .txt theo định dạng quy định, xuất file và nạp các đồ thị mẫu như Karate Club.
benchmark.py	Testing	Công cụ dòng lệnh trên terminal - để đo hiệu năng.
__init__.py	Package Init	Đánh dấu thư mục graph-app là một python package, giúp việc import module.

4.2 THUẬT TOÁN XỬ LÝ CÁC CHỨC NĂNG CHÍNH

4.2.1 Đọc dữ liệu từ file

```
Algorithm ReadGraphFromFile(path)
    Input: path – đường dẫn tới file dữ liệu đồ thị
    Output: G – đồ thị được tạo
    Read all content from file at path
    Split content into lines
    num_vertices <- first line
    is_directed <- second line
    Initialize graph G with num_vertices and is_directed
    for each line i from line 3 to end of file do
        Parse u, v, weight from line i
        if weight is not provided then
            weight <- 1
        end if
        AddEdge(G, u, v, weight)
    end for
    return G
End Algorithm
```

4.2.2 Thêm cạnh

```
Algorithm AddEdge(G, u, v, weight)
    Input: G – đồ thị
           u, v – hai đỉnh
           weight – trọng số cạnh
    if u does not exist in G then
        Add vertex u to G
```

```
    end if
    if v does not exist in G then
        Add vertex v to G
    end if
    adjacency_matrix[u][v] <- weight
    if G is undirected then
        adjacency_matrix[v][u] <- weight
    end if
End Algorithm
```

4.2.3 Tự động cập nhật đồ thị

```
Algorithm RefreshViews(G)
    Input: G – đồ thị hiện tại
    matrix <- GetAdjacencyMatrix(G)
    UpdateMatrixView(matrix)
    list <- GetAdjacencyList(G)
    UpdateAdjacencyListView(list)
    d <- CalculateGraphDensity(G)
    DisplayGraphDensity(d)
    DrawGraph(G)
End Algorithm
```

4.2.4 Vẽ đồ thị

```
Algorithm DrawGraph(G)
    Input: G – đồ thị cần hiển thị
    ClearCanvas()
    drawableGraph <- ConvertToDrawableModel(G)
    positions <- ComputeForceDirectedLayout(drawableGraph)
```

AssignVisualAttributes(drawableGraph)

DrawVertices(drawableGraph, positions)

DrawEdges(drawableGraph, positions)

DrawLabels(drawableGraph)

RefreshCanvas()

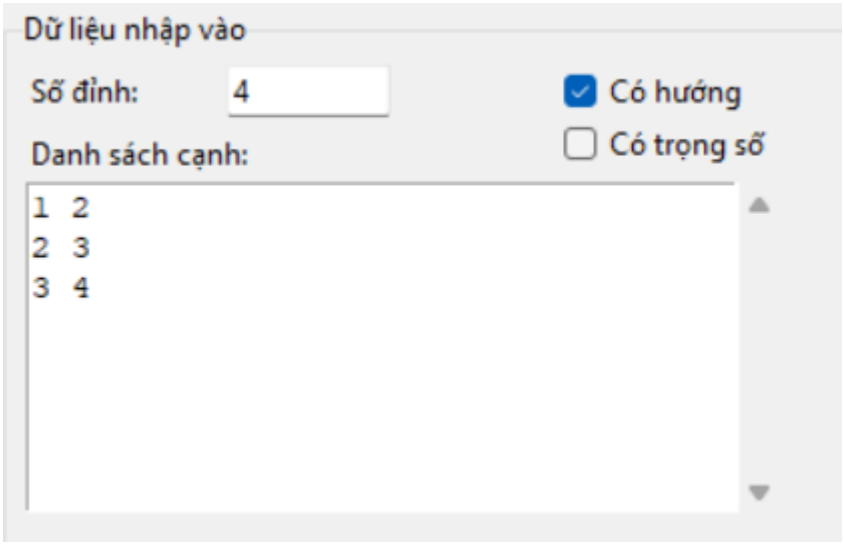
End Algorithm

4.3 MÃ NGUỒN

Mã nguồn đầy đủ của ứng dụng được lưu trữ và quản lý trên repository GitHub cá nhân, có thể tham khảo tại đường dẫn sau:

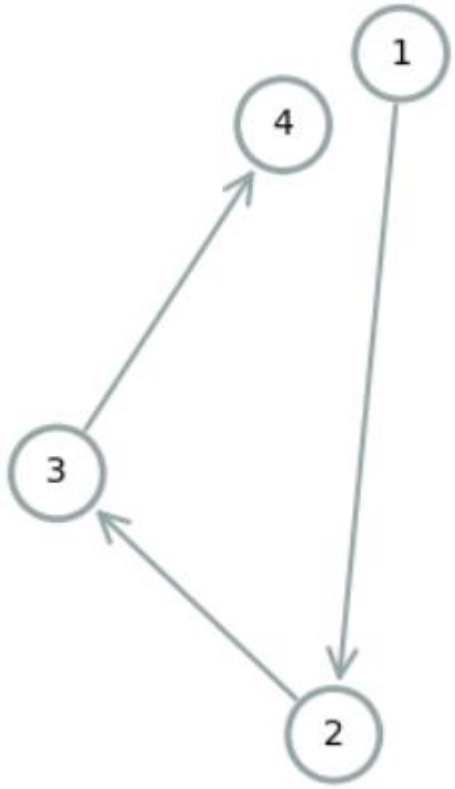
https://github.com/duy-debug/graph-visualization

Chương 5. THỰC THI CHƯƠNG TRÌNH

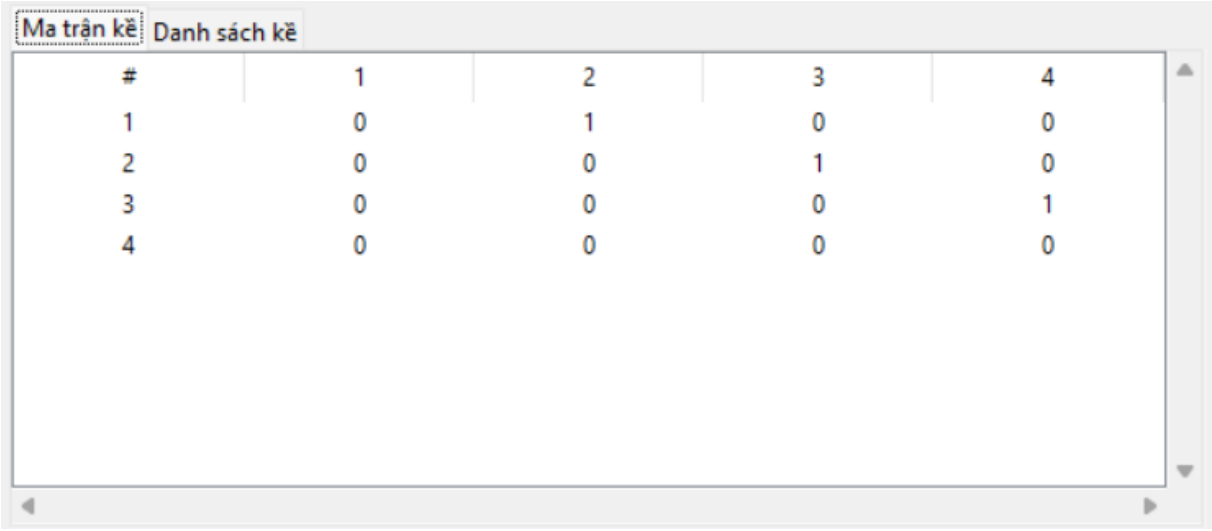


Hình 5.1. Nhập liệu mẫu cho đồ thị

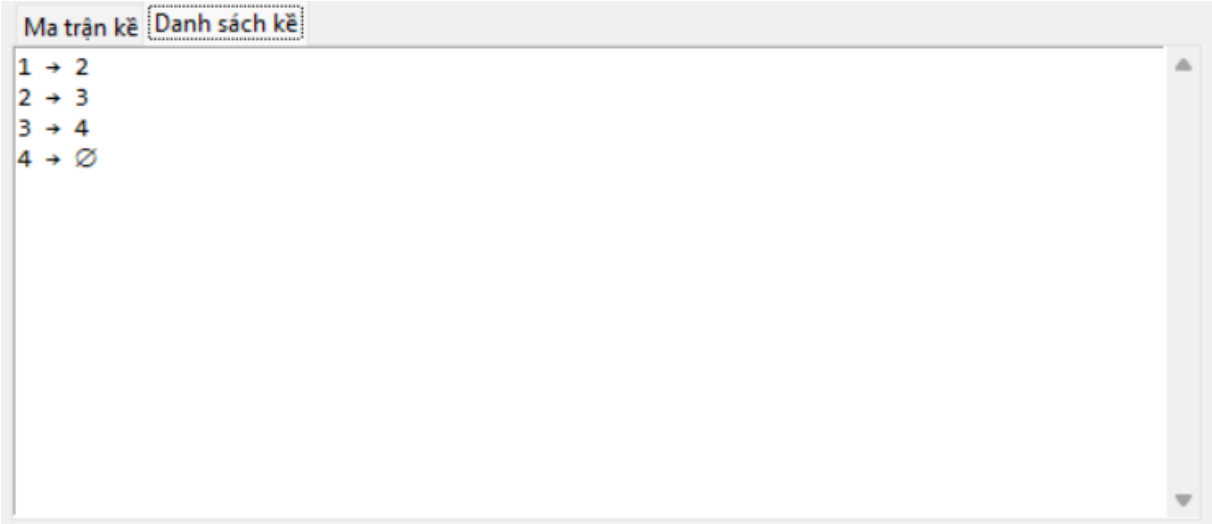
Input: Số đỉnh là 4, loại đồ thị có hướng và không có trọng số, các đỉnh lần lượt là *1, 2, 3, 4*, danh sách cạnh *1->2, 2->3, 3->4* tương ứng với 3 cạnh.



Hình 5.2. Đồ thị biểu diễn trực quan tương ứng với input đã nhập

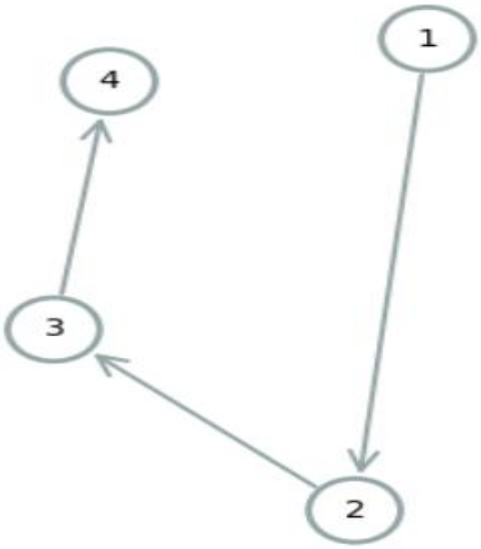


Hình 5.3. Ma trận kề tương ứng với input đã nhập

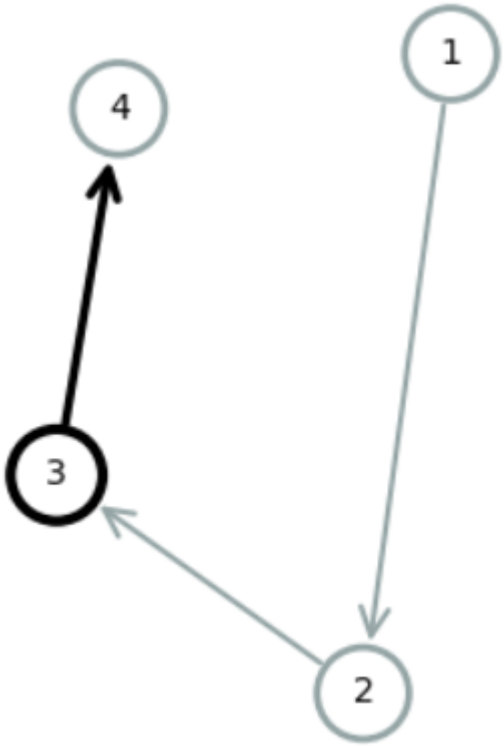


Hình 5.4. Danh sách kề tương ứng với input đã nhập

Hình 5.5. Chương trình tính toán mật độ đồ thị tương ứng với trên input đã nhập



Hình 5.6. Người dùng dùng chuột kéo đỉnh di chuyển trực tiếp trên đồ thị biểu diễn trực quan



Hình 5.7. Người dùng dùng chuột highlight vào đỉnh hoặc cạnh trực tiếp trên đồ thị biểu diễn trực quan

Hình 5.8. Dữ liệu tự động cập nhật sau khi người dùng highlight đỉnh và cạnh

```
Số lượng đỉnh: 4
Đồ thị: có hướng
Trọng số: không

Danh sách cạnh:
1 2
2 3
3 4

Ma trận kề:
#       1       2       3       4
1       0       1       0       0
2       0       0       1       0
3       0       0       0       1
4       0       0       0       0

Danh sách kề:
1 -> 2
2 -> 3
3 -> 4
4 -> Ø
```

Hình 5.9. Dữ liệu sau khi xuất file .txt

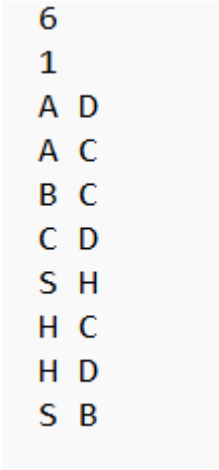
Chương trình có thể đọc từ file .txt, tệp tin cấu trúc đồ thị phải theo quy ước của các quy định sau:

- Dòng 1: Ghi một số nguyên đại diện cho số lượng đỉnh của đồ thị.
- Dòng 2: Ghi cờ loại đồ thị (nhập 1 nếu là đồ thị có hướng, 0 nếu là đồ thị vô hướng).
- Dòng 3 trở đi: Danh sách các cạnh theo định dạng

đỉnh_nguồn đỉnh_đích [trọng_số]

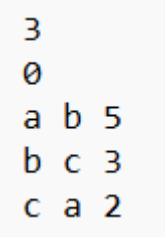
- Tên đỉnh có thể là số hoặc chữ (không chứa khoảng trắng).
- Trọng số là tùy chọn. Nếu có ít nhất một dòng có trọng số, hệ thống sẽ tự động chuyển sang chế độ "Đồ thị có trọng số". Ngược lại khi không để trọng số đồ thị mặc định "Đồ thị không có trọng số".

Lưu ý: Với đồ thị vô hướng, khi nhập một cạnh từ a -> b, chương trình sẽ tự động tạo cạnh ngược b -> a trong cấu trúc dữ liệu để đảm bảo tính đối xứng của ma trận kề.



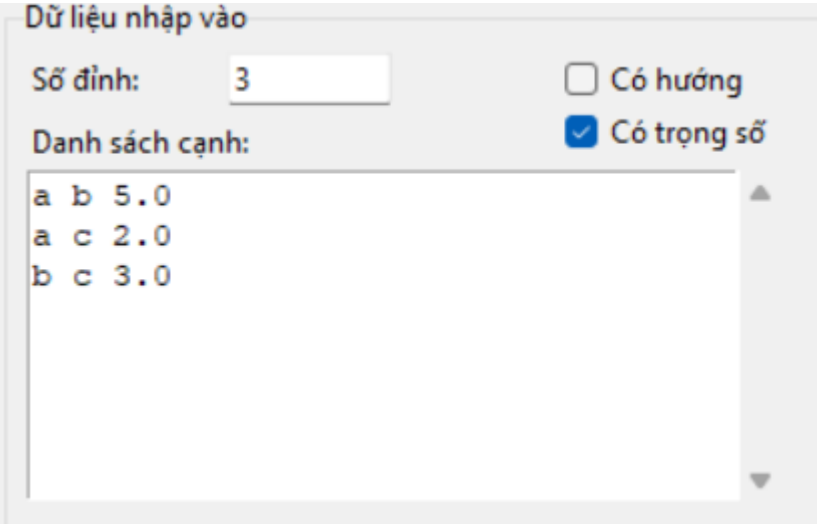
Hình 5.10. Dữ liệu nhập mẫu file .txt ứng với đồ thị có hướng không có trọng số

Input: Số đỉnh là 6, loại đồ thị có hướng ứng với cờ = 1 và không có trọng số, các đỉnh lần lượt là *A, B, C, D, S, H* danh sách cạnh *A->D, A->C, B->C, C->D, S->H, H->C, H->D, S->B.*

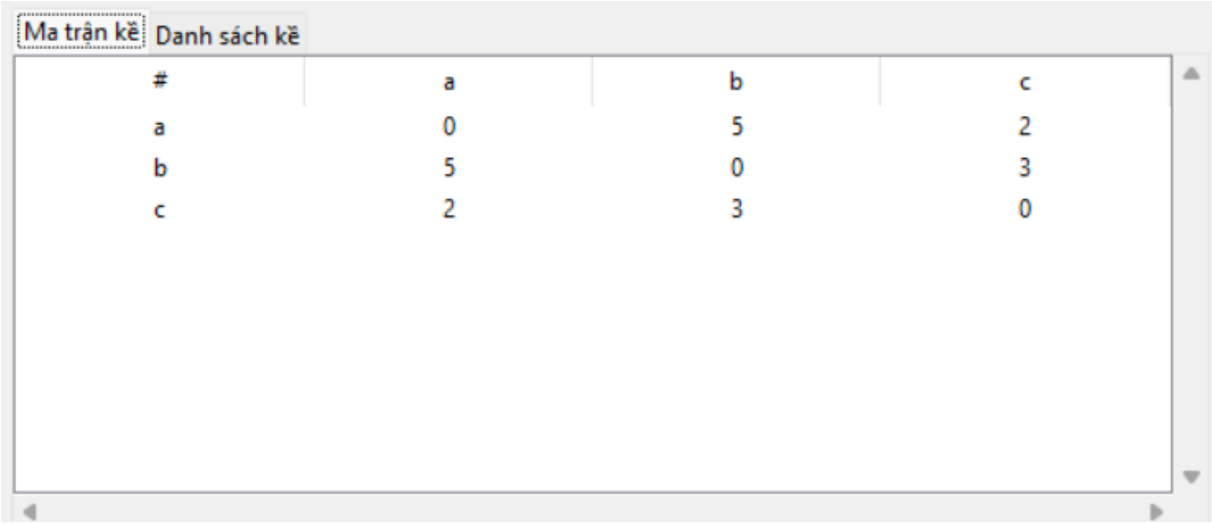


Hình 5.11. Dữ liệu nhập mẫu file .txt ứng với đồ thị vô hướng có trọng số

Input: Số đỉnh là 3, loại đồ thị vô hướng ứng với cờ = 0 và có trọng số, các đỉnh lần lượt là *a, b, c* danh sách cạnh *a-b w=5, b-c w=3, c-a w=2,* tương ứng với 3 cạnh trọng có trọng số lần lượt sau các cạnh *5, 3, 2*.

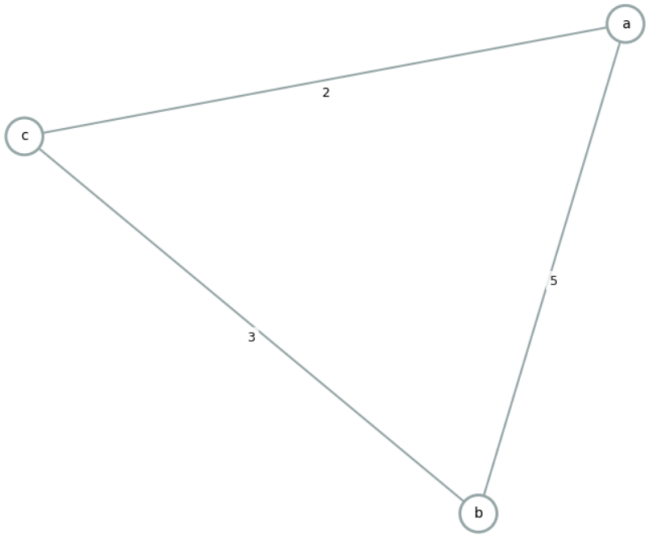


Hình 5.12. Dữ liệu được cập nhật vào ô nhập và checkbox sau khi đọc file .txt hình 5.11



Hình 5.13. Ma trận kề của đồ thị được đọc dữ liệu đầu vào từ file .txt hình 5.11

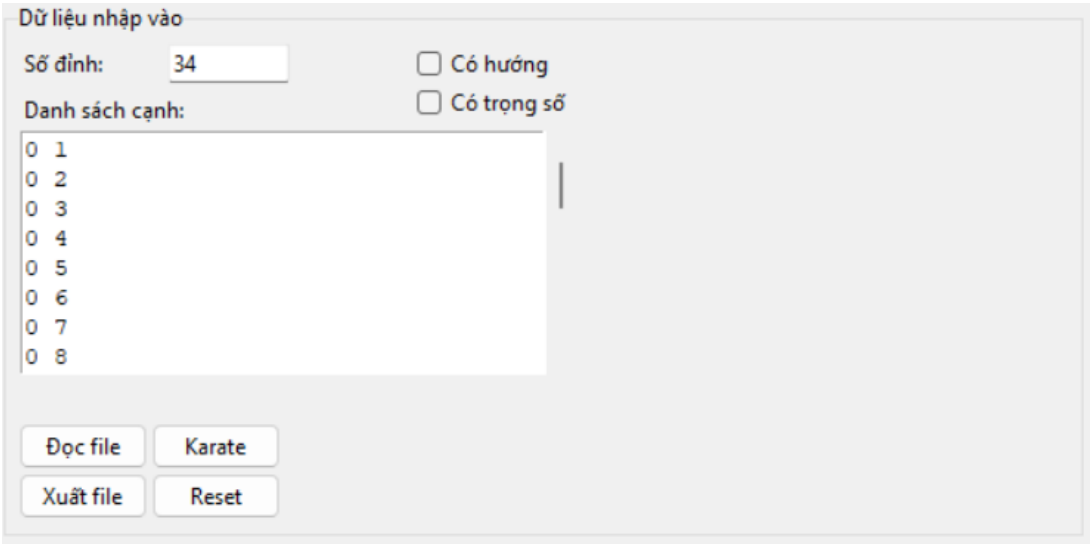
Hình 5.14. Danh sách kề của đồ thị được đọc dữ liệu đầu vào từ file .txt hình 5.11



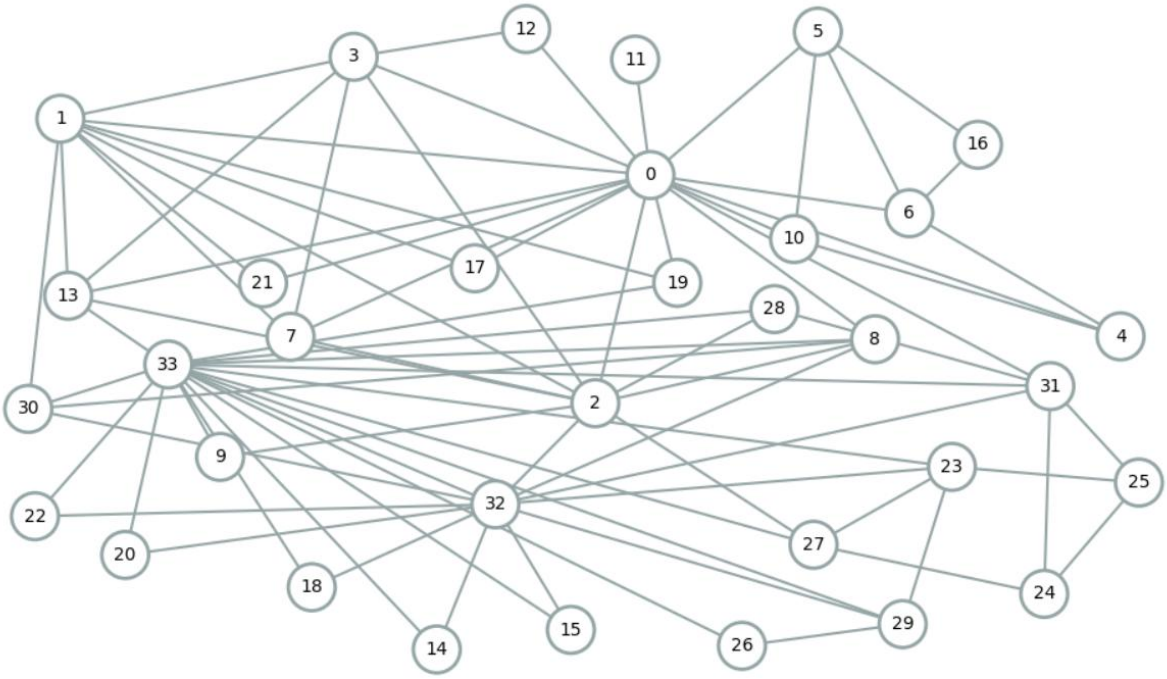
Hình 5.15. Đồ thị biểu diễn trực quan tương ứng với dữ liệu đầu vào đã đọc từ file .txt hình 5.11

Hình 5.16. Mật độ đồ thị được tính toán sau khi đọc số cạnh và đỉnh từ file .txt hình 5.11

Thử nghiệm bộ dữ liệu có sẵn từ thư viện NetworkX là Karate club.



Hình 5.17. Dữ liệu đầu vào tự động cập nhật khi người dùng trỏ vào button Karate



Hình 5.18. Đồ thị biểu diễn trực quan của bộ dữ liệu Karate club

Ma trận kề | Danh sách kề

#	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	0	1	1	1	1	1	1	1	1	0	1	1	1	1	0
1	1	0	1	1	0	0	0	1	0	0	0	0	0	1	0
2	1	1	0	1	0	0	0	1	1	1	0	0	0	1	0
3	1	1	1	0	0	0	0	1	0	0	0	0	1	1	0
4	1	0	0	0	0	0	1	0	0	0	1	0	0	0	0
5	1	0	0	0	0	0	1	0	0	0	1	0	0	0	0
6	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0
7	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
8	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0

Hình 5.19. Ma trận kề của đồ thị ứng với bộ dữ liệu Karate club

Ma trận kề | Danh sách kề

```
0 → 1, 2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 13, 17, 19, 21, 31
1 → 0, 2, 3, 7, 13, 17, 19, 21, 30
2 → 0, 1, 3, 7, 8, 9, 13, 27, 28, 32
3 → 0, 1, 2, 7, 12, 13
4 → 0, 6, 10
5 → 0, 6, 10, 16
6 → 0, 4, 5, 16
7 → 0, 1, 2, 3
8 → 0, 2, 30, 32, 33
9 → 2, 33
10 → 0, 4, 5
11 → 0
12 → 0, 3
13 → 0, 1, 2, 3, 33
14 → 32, 33
```

Hình 5.20. Danh sách kề của đồ thị ứng với bộ dữ liệu Karate club

Bảng điều khiển

Highlight Đỉnh:

Highlight Cạnh:

Mật độ: 0.139 (Đồ thị thưa)

Hình 5.21. Chương trình tính toán mật độ đồ thị ứng với bộ dữ liệu Karate club

Thông qua quá trình thực thi và kiểm thử với các bộ dữ liệu đa dạng từ đồ thị thưa đơn giản đến đồ thị dày phức tạp có hướng và trọng số chương trình đã chứng minh được tính ổn định và độ chính xác cao trong xử lý dữ liệu. Các kết quả hiển thị trực quan trên biểu diễn đồ thị trực quan, bảng ma trận kề và danh sách kề đều có sự đồng bộ tuyệt đối với dữ liệu nhập từ tệp tin .txt. Điều này khẳng định ứng dụng không chỉ đáp ứng tốt các yêu cầu chức năng đặt ra mà còn đảm bảo được tính thực tiễn, hỗ trợ người dùng quản lý và phân tích cấu trúc đồ thị một cách hiệu quả và tin cậy.

Kết thúc quá trình thực thi, có thể khẳng định chương trình đã giải quyết tốt bài toán trực quan hóa đồ thị với độ chính xác về mặt toán học cao. Việc xử lý thành công các ràng buộc về định dạng file, cùng khả năng tương tác linh hoạt trên giao diện, đã tạo nên một công cụ hỗ trợ học tập và nghiên cứu đồ thị trực quan. Kết quả kiểm thử đạt yêu cầu chính là minh chứng rõ nét nhất cho khả năng ứng dụng của sản phẩm trong các bài toán mô hình hóa mạng lưới thực tế.

Chương 6. ĐÁNH GIÁ HIỆU NĂNG HỆ THỐNG

6.1 MỤC ĐÍCH ĐÁNH GIÁ

Để đánh giá hiệu năng hoạt động của ứng dụng quản lý đồ thị, nghiên cứu đã tiến hành các bài kiểm tra với nhiều kích thước và mật độ đồ thị khác nhau. Mục tiêu đánh giá bao gồm:

- Đo thời gian tạo cấu trúc dữ liệu đồ thị.
- Đo thời gian thực hiện các thao tác truy vấn.
- Đo thời gian tạo ma trận kề và danh sách kề.
- So sánh thời gian xử lý cấu trúc dữ liệu với thời gian vẽ đồ thị.

6.2 PHƯƠNG PHÁP ĐÁNH GIÁ

Nghiên cứu sử dụng đồ thị ngẫu nhiên với các tham số

- Số lượng đỉnh (n): 50, 200, 500 đỉnh.
- Mật độ đồ thị: 10%, 30%, 50%.
- Loại đồ thị: Vô hướng, không có trọng số.

Công thức tính số cạnh dựa trên mật độ:

$$Số\ cạnh = Mật\ độ \times [n \times (n - 1) / 2]$$

Các thao tác truy vấn được thực hiện 1000 lần để đảm bảo độ chính xác của phép đo.

6.3 KẾT QUẢ ĐÁNH GIÁ

```
Đang test: 50 đỉnh, mật độ 10%... Hoàn thành (122 cạnh)
Đang test: 50 đỉnh, mật độ 30%... Hoàn thành (367 cạnh)
Đang test: 50 đỉnh, mật độ 50%... Hoàn thành (612 cạnh)
Đang test: 200 đỉnh, mật độ 10%... Hoàn thành (1990 cạnh)
Đang test: 200 đỉnh, mật độ 30%... Hoàn thành (5970 cạnh)
Đang test: 200 đỉnh, mật độ 50%... Hoàn thành (9950 cạnh)
Đang test: 500 đỉnh, mật độ 10%... Hoàn thành (12475 cạnh)
Đang test: 500 đỉnh, mật độ 30%... Hoàn thành (37425 cạnh)
Đang test: 500 đỉnh, mật độ 50%... Hoàn thành (62375 cạnh)
```

Hình 6.1. Quá trình thực thi đánh giá hiệu năng với các cấu trúc đồ thị khác nhau

Hình 6.1 cho thấy các bộ test được chạy thành công với:

- 3 kích thước đồ thị: 50, 200, 500 đỉnh.

- 3 mức mật độ: 10%, 30%, 50%.
- Tổng cộng 9 trường hợp kiểm thử.

Số cạnh tăng theo công thức:

- Mật độ 10%: 122 -> 1990 -> 12475 cạnh.
- Mật độ 30%: 367 -> 5970 -> 37425 cạnh.
- Mật độ 50%: 612 -> 9950 -> 62375 cạnh.

Đỉnh	Mật độ	Số cạnh	Tạo CTDL (ms)	Check cạnh (1000x, ms)	Lấy kề (1000x, ms)	Ma trận kề (ms)	DS kề (ms)	Vẽ (ms)
50	10%	122	0.131	0.057	0.111	0.166	0.011	26.14
50	30%	367	0.341	0.063	0.158	0.182	0.012	19.43
50	50%	612	0.534	0.066	0.244	0.185	0.018	17.48
200	10%	1990	4.148	0.085	0.222	2.633	0.061	74.29
200	30%	5970	10.599	0.071	0.683	3.770	0.179	96.28
200	50%	9950	18.066	0.093	0.889	2.699	0.211	108.66
500	10%	12475	50.011	0.083	0.445	15.716	0.269	N/A
500	30%	37425	147.418	0.221	1.573	19.380	0.937	N/A
500	50%	62375	257.254	0.194	3.167	18.781	1.359	N/A

Hình 6.2. Kết quả đo thời gian xử lý (đơn vị: mili-giây)

Thời gian tạo cấu trúc dữ liệu:

- Thời gian tăng tuyến tính theo số cạnh (O(m)).
- Khi số cạnh tăng 511 lần (122->62375), thời gian tăng 1963 lần (0.131->257.254ms).
- Nguyên nhân: Phải duyệt qua tất cả các cạnh để xây dựng đồ thị.

Thời gian kiểm tra cạnh:

- Thời gian gần như không đổi (~0.06-0.19ms cho 1000 lần kiểm tra).
- Đạt độ phức tạp O(1) nhờ sử dụng Dictionary.
- Kết luận: Hiệu quả cao, không phụ thuộc kích thước đồ thị.

Thời gian lấy kề:

- Thời gian tăng theo số đỉnh kề trung bình.
- Với mật độ 50%, mỗi đỉnh có nhiều đỉnh kề hơn -> tốn thời gian hơn.
- Độ phức tạp: O(deg(v)) - phụ thuộc bậc của đỉnh.

Thời gian tạo ma trận kề:

- Thời gian phụ thuộc chủ yếu vào n^2 (số đỉnh).
- Tăng 101 lần khi đỉnh tăng 10 lần (50->500) ~ 10^2.
- Độ phức tạp: $O(n^2)$ - phải khởi tạo ma trận kích thước n×n.

Thời gian tạo danh sách kề:

- Nhanh hơn rất nhiều so với ma trận kề.
- Chỉ lưu trữ các cạnh thực sự tồn tại.
- Độ phức tạp: O(n + m).

Thời gian vẽ đồ thị:

- Chậm nhất trong tất cả các thao tác.
- Phụ thuộc vào thư viện đồ họa Matplotlib hoặc NetworkX.
- Đây là điểm nghẽn hiệu năng của hệ thống.

```
PHÂN TÍCH THEO KÍCH THƯỚC ĐỒ THỊ:
------------------------------------------------------------
  - 50 đỉnh: Tạo CTDL TB = 0.335 ms, Ma trận kề TB = 0.178 ms
  - 200 đỉnh: Tạo CTDL TB = 10.938 ms, Ma trận kề TB = 3.034 ms
  - 500 đỉnh: Tạo CTDL TB = 151.561 ms, Ma trận kề TB = 17.959 ms
```

Hình 6.3. Thời gian trung bình theo kích thước đồ thị

Tạo cấu trúc dữ liệu tăng nhanh hơn vì phụ thuộc số cạnh tăng theo n^2. Ma trận kề tăng đều đặn theo n^2 ứng với tỷ lệ:

- 50->200 đỉnh: tăng ~ 33x (tạo CTDL), ~ 17x (ma trận).
- 200->500 đỉnh: tăng ~ 14x (tạo CTDL), ~ 6x (ma trận).

```
PHÂN TÍCH THEO MẬT ĐỘ:
------------------------------------------------------------
  - Mật độ 10%: Số cạnh TB = 4862, Tạo CTDL TB = 18.097 ms
  - Mật độ 30%: Số cạnh TB = 14587, Tạo CTDL TB = 52.786 ms
  - Mật độ 50%: Số cạnh TB = 24312, Tạo CTDL TB = 91.951 ms
```

Hình 6.4. Thời gian trung bình theo mật độ

Thời gian tăng tuyến tính với số cạnh . Mật độ 50% (đồ thị dày) chậm gấp ~5 lần so với mật độ 10% (đồ thị thưa). Khuyến khích với đồ thị dày nên dùng ma trận kề, với đồ thị thưa nên dùng danh sách kề.

```
SO SÁNH THỜI GIAN XỬ LÝ DỮ LIỆU vs VẼ ĐỒ THỊ:
------------------------------------------------------------
  • 50 đỉnh, 10%: Xử lý = 0.31 ms, Vẽ = 26.14 ms (gấp 84.8x)
  • 50 đỉnh, 30%: Xử lý = 0.53 ms, Vẽ = 19.43 ms (gấp 36.4x)
  • 50 đỉnh, 50%: Xử lý = 0.74 ms, Vẽ = 17.48 ms (gấp 23.7x)
  • 200 đỉnh, 10%: Xử lý = 6.84 ms, Vẽ = 74.29 ms (gấp 10.9x)
  • 200 đỉnh, 30%: Xử lý = 14.55 ms, Vẽ = 96.28 ms (gấp 6.6x)
  • 200 đỉnh, 50%: Xử lý = 20.98 ms, Vẽ = 108.66 ms (gấp 5.2x)
```

Hình 6.5. So sánh thời gian xử lý dữ liệu và vẽ đồ thị

Kết quả cho thấy thời gian vẽ đồ thị sử dụng Matplotlib và NetworkX lớn hơn khoảng từ 5.2 đến 84.8 lần so với thời gian xử lý cấu trúc dữ liệu. Điều này chứng minh thuật toán và cấu trúc dữ liệu được thiết kế trong ứng dụng hoạt động rất hiệu quả. Phần hiển thị đồ họa là điểm nghẽn hiệu năng do phụ thuộc vào thư viện bên ngoài.

Chương 7. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

7.1 KẾT QUẢ ĐẠT ĐƯỢC

Ứng dụng đã xây dựng bộ công cụ quản lý đồ thị hoàn chỉnh với kiến trúc mã nguồn rõ ràng, tách biệt giữa xử lý dữ liệu và giao diện. Phần mềm hỗ trợ linh hoạt nhiều loại đồ thị có hướng, vô hướng, có trọng số hoặc không có trọng số cùng hệ thống nhập xuất dữ liệu đa dạng từ file văn bản đến nhập liệu thủ công. Bên cạnh đó điểm nổi bật là khả năng trực quan hóa tương tác, cho phép người dùng kéo thả các đỉnh và cho phép highlight trực tiếp lên các đỉnh và cạnh, đồng thời tự động sinh cấu trúc ma trận kề, danh sách kề và tính toán hiển thị mật độ của đồ thị.

Về mặt hiệu năng, hệ thống đã được kiểm thử thành công với đồ thị lên đến 500 đỉnh và 62.375 cạnh. Kết quả đánh giá cho thấy các thao tác cốt lõi đạt độ phức tạp lý thuyết kiểm tra cạnh đạt O(1), tạo ma trận kề đạt $O(n^2)$, và tạo danh sách kề đạt O(n+m). Đặc biệt, thời gian xử lý cấu trúc dữ liệu chỉ chiếm 5-20% tổng thời gian, chứng tỏ thuật toán được tối ưu tốt. Đề tài đã đạt được các mục tiêu nghiên cứu đề ra, bao gồm hệ thống hóa kiến thức về đồ thị và hai phương pháp biểu diễn chính, xây dựng thành công phần mềm với đầy đủ chức năng quản lý và trực quan hóa, thực hiện đánh giá so sánh hiệu năng giữa ma trận kề và danh sách kề, kiểm thử với dữ liệu thực tế. Sản phẩm có thể được sử dụng làm công cụ hỗ trợ giảng dạy và học tập môn Cấu trúc dữ liệu và giải thuật tại Khoa Công nghệ thông tin, Trường Đại học Nha Trang.

Bên cạnh các kết quả đạt được về mặt lý thuyết và triển khai, toàn bộ mã nguồn của hệ thống đã được xây dựng đầy đủ và công khai, giúp đảm bảo tính minh bạch và khả năng tái hiện của đề tài. Phần cài đặt chi tiết đã được trình bày cụ thể trong Chương 4 - Mục 4.3, qua đó người đọc có thể dễ dàng tiếp cận, kiểm thử và phát triển thêm hệ thống trong các nghiên cứu hoặc ứng dụng tiếp theo.

7.2 NHỮNG HẠN CHẾ

Về hiệu năng, thời gian vẽ đồ thị chiếm 80-95% tổng thời gian xử lý do phụ thuộc vào thư viện Matplotlib và NetworkX. Điều này khiến việc trực quan hóa các đồ thị lớn trên 500 đỉnh trở nên chậm và ảnh hưởng đến trải nghiệm người dùng. Hơn nữa, thuật toán bố trí đồ thị tự động đôi khi tạo ra các cạnh chồng chéo, gây khó khăn trong quan sát với đồ thị phức tạp. Về chức năng, hệ thống mới chỉ hỗ trợ biểu diễn và quản lý đồ

thị cơ bản, chưa tích hợp các thuật toán duyệt và tìm đường như Breadth-First Search (BFS), Depth-First Search (DFS), Dijkstra.

Giao diện người dùng tuy đơn giản dễ sử dụng nhưng chưa được tối ưu về mặt thẩm mỹ và thiếu một số tính năng nâng cao như undo hoặc redo, lưu trạng thái làm việc, hoặc xuất hình ảnh đồ thị chất lượng cao. Về mặt kỹ thuật, ứng dụng hiện chỉ chạy trên môi trường desktop, hạn chế khả năng tiếp cận và chia sẻ.

7.3 HƯỚNG PHÁT TRIỂN

Trong thời gian tới, ứng dụng có thể được mở rộng bằng cách tích hợp các thuật toán đồ thị cơ bản như Breadth-First Search (BFS), Depth-First Search (DFS), Dijkstra mô phỏng hóa quá trình chạy thuật toán dưới dạng hoạt ảnh để nâng cao tính giáo dục và phân tích để phục vụ học tập và nghiên cứu. Đồng thời, cần bổ sung chức năng đo hiệu năng về bộ nhớ và thời gian chạy nhằm đánh giá tính hiệu quả của từng cách biểu diễn. Ngoài ra, một hướng phát triển quan trọng khác là chuyển đổi ứng dụng từ nền tảng desktop sang nền tảng Web. Việc triển khai trên Web sẽ giúp ứng dụng dễ dàng truy cập trên nhiều thiết bị, không phụ thuộc vào hệ điều hành, đồng thời tạo điều kiện thuận lợi cho việc mở rộng tính năng chia sẻ mã nguồn cho phép người khác đóng góp mã nguồn.

TÀI LIỆU THAM KHẢO

[1] NetworkX Developers (2025). *NetworkX – Software for Complex Networks*. Truy cập tại: https://networkx.org/en/. Ngày truy cập: 24/11/2025.

[2] Graphviz Authors (2025). *Graphviz – Graph Drawing Software*. Official documentation. Truy cập tại: https://graphviz.readthedocs.io/en/stable/index.html. Ngày truy cập: 24/11/2025.

[3] Viblo Asia (2024). *Giới thiệu về Lý thuyết đồ thị*. Truy cập tại: https://viblo.asia/p/gioi-thieu-ve-ly-thuyet-do-thi-07LKXQ1eZV4. Ngày truy cập: 24/11/2025.

[4] Nguyễn Đình Hưng (2023). *Lập trình Python*. Truy cập tại: https://github.com/nd-hung/python-programming. Ngày truy cập: 24/11/2025.

[5] Nguyễn Đức Nghĩa, Nguyễn Tô Thành (2009). *Giáo trình Toán rời rạc*. Thư viện Trường Đại học Nha Trang, Nha Trang.

[6] Nguyễn Đức Nghĩa (2013). *Cấu trúc dữ liệu và thuật toán*. Thư viện Trường Đại học Nha Trang, Nha Trang.

[7] Phạm Hữu Lợi (2024). *Biểu diễn dữ liệu dạng điểm với thư viện Matplotlib của Python*. Thư viện Trường Đại học Nha Trang, Nha Trang.