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 & 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 & 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 những năm học vừa qua, hỗ trợ em trong suốt quá trình học tập.

Đặc biệt, em xin gửi lời tri âm sâu sắc nhất đến cô TS. Phạm Thị Thu Thúy cô hướng dẫn trực tiếp của em. Với sự tận tâm, những chỉ bảo tỉ mỉ, 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 hiện 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 & 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	07/12/2024	Hoàn thành việc xây dựng cơ bản mô phỏng biểu diễn đồ thị bằng ma trận kề & danh sách kề, viết báo cáo lần 1.	Chỉnh sửa lại giao diện biểu diễn đồ thị trực quan với kích thước lớn hơn.
02	21/12/2024	Hoàn thành bản mô phỏng biểu diễn đồ thị bằng ma trận kề & danh sách kề đã sửa, viết báo cáo lần 2.	
03			

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: ..... **MSSV:** .....

3. Tên đề tài: .....

.....

4. Nhận xét

a) Kết quả thực hiện đề tài

.....

.....

.....

b) Báo cáo thực tập

- 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

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 hiện nay, lý thuyết đồ thị đã và đang trở thành nền tảng không thể thiếu của rất nhiều ứng dụng thực tiễn như mạng xã hội, hệ thống đề xuất, tìm kiếm đường đi ngắn nhất, phân tích mạng máy tính, sinh học tin học và trí tuệ nhân tạo. Chính vì vậy, việc nắm vững cách biểu diễn đồ thị trong máy tính là kỹ năng cơ bản và quan trọng đối với sinh viên ngành Công nghệ thông tin.

Xuất phát từ thực tế đó, em quyết định lựa chọn đề tài **"Mô phỏng biểu diễn đồ thị bằng Ma trận kề và Danh sách kề"** bởi vì đồ thị chính là cấu trúc dữ liệu cốt lõi của môn cấu trúc dữ liệu và giải thuật, đồng thời là nền tảng quan trọng để tiếp cận các thuật toán nâng cao như DFS, BFS, Dijkstra, Kruskal, Prim, Floyd-Warshall. Hơn nữa, khi tự tay cài đặt cả hai cách biểu diễn phổ biến là ma trận kề và danh sách kề, em có thể trực tiếp cảm nhận rõ sự đánh đổi thực tế giữa chúng: ma trận kề tiêu tốn $O(n^2)$ bộ nhớ nhưng kiểm tra cạnh chỉ mất $O(1)$, trong khi danh sách kề chỉ cần $O(n+m)$ bộ nhớ cực kỳ hiệu quả với đồ thị thưa dù thời gian kiểm tra cạnh lại phụ thuộc vào bậc của đỉnh. Việc tự xây dựng và quan sát trực tiếp sự khác biệt này sẽ giúp em cũng như các bạn sinh viên hiểu sâu sắc và thực tế hơn rất nhiều so với chỉ học lý thuyết suông trên lớp.

Bên cạnh đó, đề tài giúp em rèn luyện toàn diện các kỹ năng lập trình từ quản lý bộ nhớ động, xử lý cấu trúc dữ liệu phức tạp, đến việc sử dụng các thư viện trực quan hóa mạnh mẽ như NetworkX, Matplotlib và Graphviz những công cụ mà sinh viên công nghệ thông tin sẽ thường xuyên sử dụng trong công việc sau này. Đặc biệt, việc tự vẽ và tương tác với đồ thị (hiển thị đỉnh, cạnh, highlight các thành phần) giúp chuyển hóa những khái niệm trừu tượng trên giấy thành hình ảnh trực quan sinh động, từ đó củng cố sự hiểu biết về các khái niệm như đỉnh kề, bậc của đỉnh, đồ thị thưa hay dày.

Cuối cùng, sản phẩm của đề tài không chỉ dừng lại ở một chương trình minh họa đơn thuần mà còn có khả năng mở rộng cao, dễ dàng tích hợp thêm các thuật toán duyệt đồ thị, tìm đường, phát hiện chu trình hay bài toán luồng mạng, qua đó trở thành công cụ học tập và nghiên cứu hữu ích cho chính bản thân em cũng như các thế hệ sinh viên sau này tại Khoa Công nghệ thông tin Trường Đại học Nha Trang.

Với những lý do nêu trên, em tin rằng đề tài không chỉ giúp em củng cố vững chắc kiến thức nền tảng mà còn góp phần nhỏ vào việc nâng cao chất lượng giảng dạy và học tập môn học, đồng thời mang lại một sản phẩm phần mềm mang tính giáo dục cao, giúp các bạn sinh viên tiếp cận lý thuyết đồ thị một cách dễ dàng, trực quan và dễ tiếp cận.

1.2 MỤC TIÊU NGHIÊN CỨU

1.2.1 Mục tiêu đề tài

- Hệ thống hóa và củng cố các khái niệm cơ bản về lý thuyết đồ thị bao gồm: đồ thị vô hướng hoặc có hướng, có trọng số hoặc 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 nhất trong máy tính: ma trận kề và danh sách kề.
- Phân tích, so sánh rõ ràng ưu điểm và nhược điểm của từng phương pháp về độ phức tạp bộ nhớ, thời gian truy vấn và khả năng áp dụng thực tiễn.
- Xây dựng chương trình hoàn chỉnh cho phép người dùng nhập đồ thị (từ bàn phím hoặc file), tự động sinh cả hai cấu trúc ma trận kề và danh sách kề, đồng thời hiển thị chúng dưới nhiều dạng: bảng ma trận, danh sách liên kết và đồ thị trực quan (node-edge).
- Rèn luyện kỹ năng lập trình các cấu trúc dữ liệu động, quản lý bộ nhớ hiệu quả và sử dụng thành thạo các thư viện trực quan hóa mạnh mẽ của Python như NetworkX, Matplotlib, Graphviz.
- Thực hiện các chức năng nâng cao thêm hoặc xóa đỉnh-cạnh, xuất file cấu trúc, đọc và hiển thị bộ dữ liệu thực tế Karate Club 34 đỉnh nhằm nâng cao tính tương tác và khả năng mở rộng của phần mềm.
- Tạo ra một công cụ mô phỏng trực quan, sinh động giúp người học dễ dàng "nhìn thấy" cấu trúc đồ thị thay vì chỉ tưởng tượng qua lý thuyết, từ đó hiểu sâu hơn bản chất của các khái niệm trừu tượng.
- 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.
- Xây dựng một sản phẩm phần mềm có 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 các khóa sau tại Khoa Công nghệ thông tin Trường Đại học Nha Trang.

1.2.2 Đối tượng và phạm vi nghiên cứu

- Đối tượng nghiên cứu: Ma trận kề và danh sách kề.
- Phạm vi nghiên cứu:
  - Các loại đồ thị cơ bản: vô hướng, có hướng, có trọng số và không trọng số.
  - Hai cách biểu diễn đồ thị trong bộ nhớ máy tính: ma trận kề và danh sách kề.
  - So sánh ưu nhược điểm về bộ nhớ, tốc độ truy vấn và tính phù hợp với đồ thị thưa hoặc đồ thị dày.
  - Trực quan hóa đồ thị bằng hình vẽ (node edge) và các chức năng tương tác cơ bản.

1.2.3 Công nghệ sử dụng

STT	Công nghệ hoặc thư viện	Mục đích
01	Python	Ngôn ngữ lập trình chính.
02	NetworkX	Tạo, quản lý và thao tác đồ thị, cung cấp bộ dữ liệu mẫu Karate Club.
03	Matplotlib	Vẽ đồ thị trực quan (node edge), tùy chỉnh màu sắc, kích thước.
04	Graphviz	Bố cục đồ thị đẹp, chuyên nghiệp.
05	Tkinter	Xây dựng giao diện người dùng đồ họa đơn giản.
06	Text document file	Lưu trữ và đọc dữ liệu đồ thị từ file.
07	Visual Studio Code	Môi trường phát triển .
08	Git và GitHub	Quản lý mã nguồn và phiên bản.

Chương 2. CƠ SỞ LÝ THUYẾT

2.1 KHÁI NIỆM ĐỒ THỊ

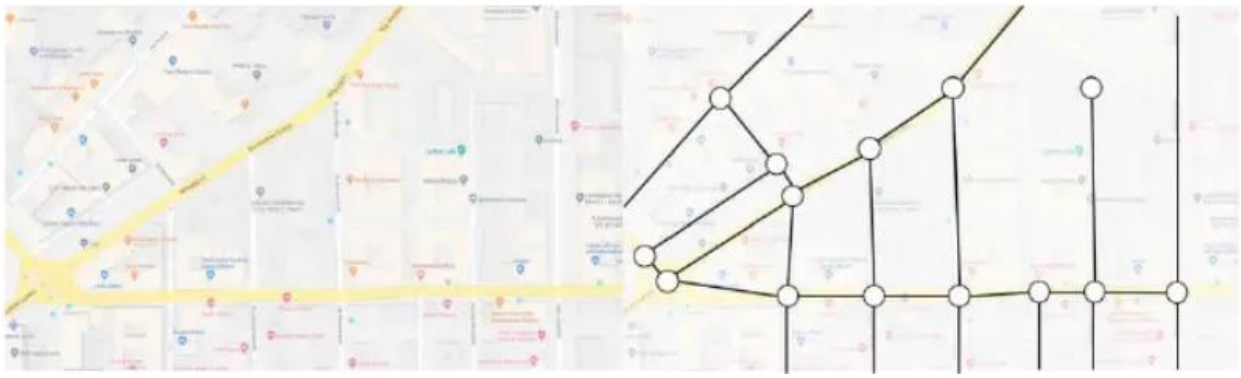
2.1.1 Định nghĩa đồ thị

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), 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:*

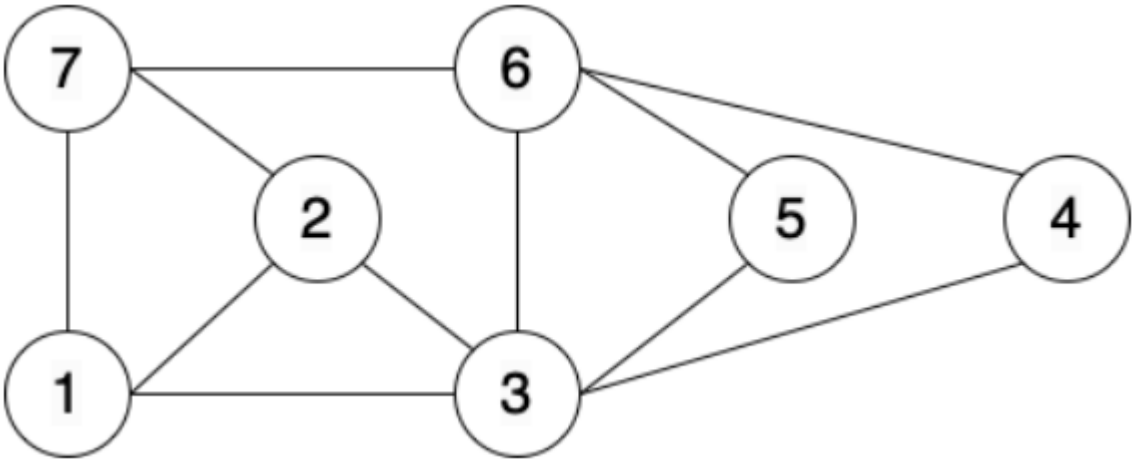
V là tập các đỉnh.

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 các đỉnh đầu 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:*

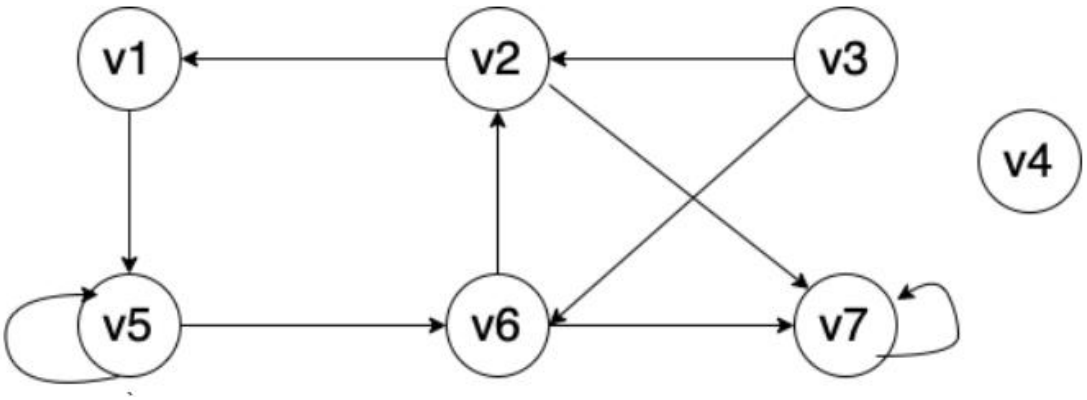
V là tập các đỉnh.

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 vô 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)

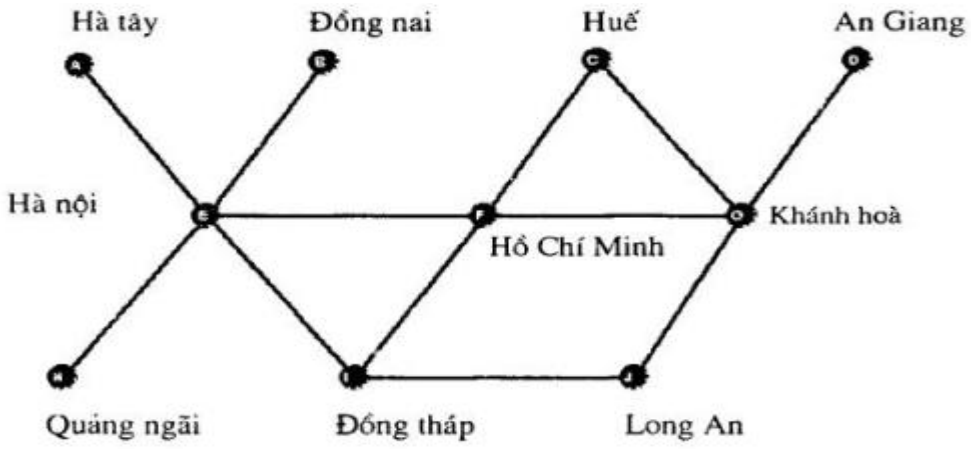
Hơn nữa, chúng ta có thể phân loại đồ thị dựa vào số cạnh nối giữa hai đỉnh xác định đơn đồ thị hay đa đồ thị.

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.

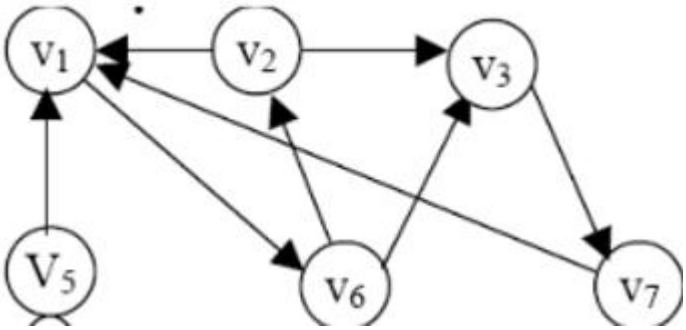
Không có cạnh song song: tức là giữa hai đỉnh, không có nhiều hơn một cung cùng chiều



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.*

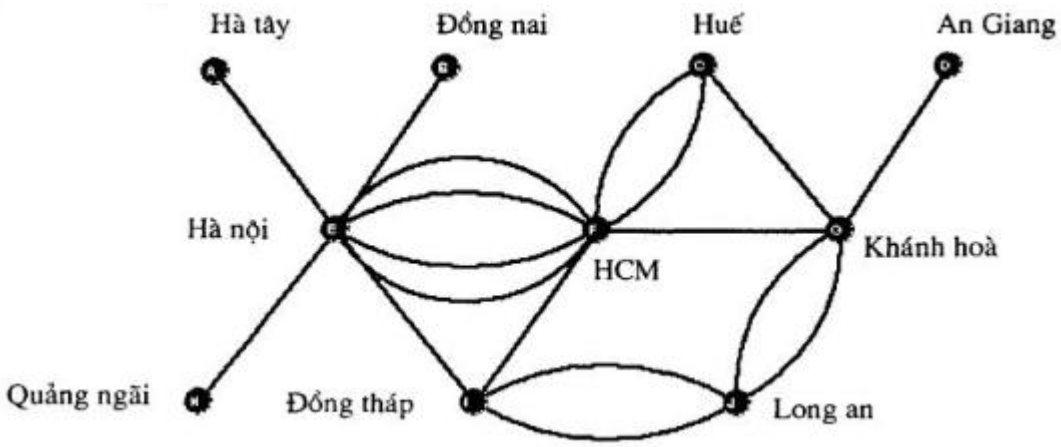
Đơn đồ thị có hướng G = (V, E) bao gồm V là tập các đỉnh khác rỗng, và E là tập các cặp có thứ tự gồm hai phần tử khác nhau của V gọi là các cung, không có khuyên và không có cạnh song song.



Hình 2.6. Đơn đồ thị có hướng

b) Đa đồ thị

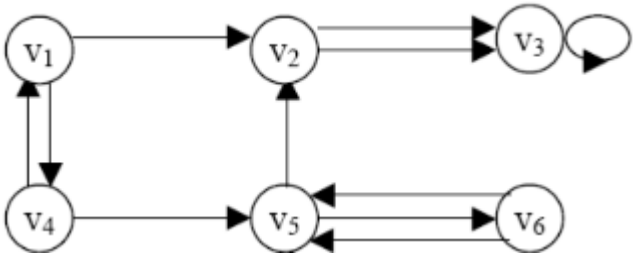
Đa đồ thị vô hướng G = (V, E) bao gồm V là tập các đỉnh khác rỗng và E là tập các cặp không có thứ tự gồm hai phần tử khác nhau của V gọi là các cạnh. Hai cạnh e1, e2 tương ứng với cùng một cặp đỉnh được gọi là cạnh lặp.



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

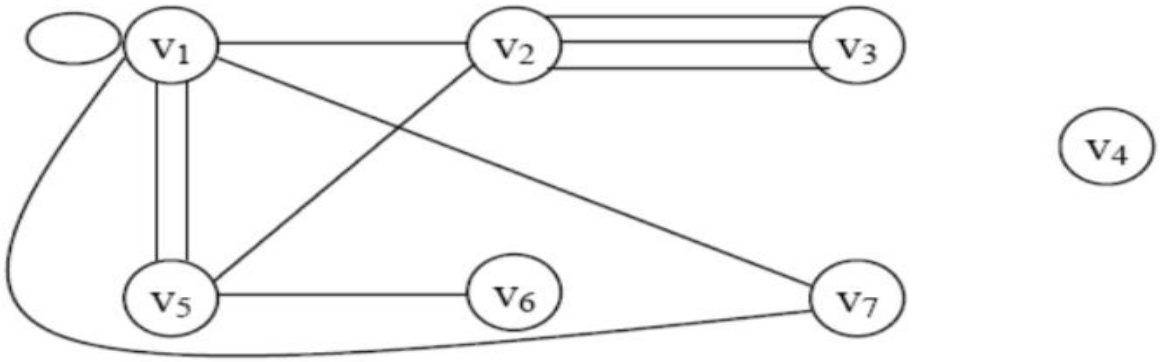
Trong mục này, chúng ta sẽ tìm hiểu và trình bày một số khái niệm nền tảng của lĩnh vực lý thuyết đồ thị. Những khái niệm này đóng vai trò quan trọng giúp ta mô tả, phân tích và hiểu được cấu trúc của các loại đồ thị khác nhau. Để bắt đầu, chúng ta sẽ

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 = (V, E) được gọi là liền kề nhau nếu (u, v) là cạnh của đồ thị G hay (u,v) ∈ E. Nếu e = (u,v) thì gọi e là cạnh liên thuộc với các đỉnh u và v hoặc có thể nói cạnh e cũng được gọi là cạnh nối đỉnh u và v. Đồng thời, các đỉnh u và v được gọi là các điểm đầu mút của cạnh e.*

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. *Bậc của định v trong đồ thị G = (V, E), ký hiệu deg(v) là số cạnh liên thuộc với nó, riêng khuyên tại một đỉnh được tính hai lần cho bậc của nó.*



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. Khi đó tổng bậc của tất cả các đỉnh bằng hai lần số cạnh.*

$$2m = a_0 + \sum_{v \in V} \deg(v)$$

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.*

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

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

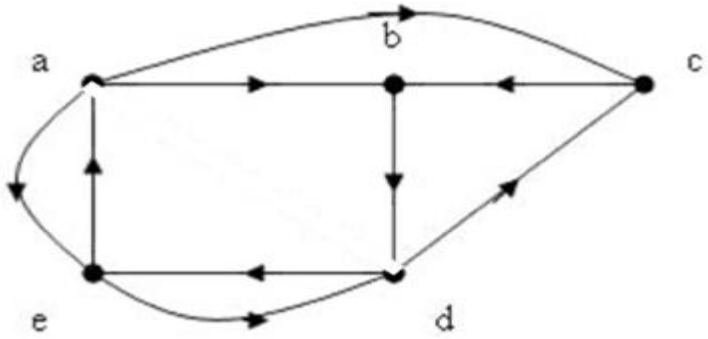
Do deg(v) là chẵn với v là đỉnh trong U nên tổng thứ nhất ở trên là số chẵn. Suy ra tổng thứ hai (chính là tổng bậc của các đỉnh bậc lẻ) cũng phải là số chẵn. Do tất cả các số hạng của nó là số lẻ, nên tổng này phải gồm một số chẵn các số hạng. Vì vậy, số đỉnh bậc lẻ phải là số chẵn.

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).*

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 $deg^+(v)$ ($deg^-(v)$).*



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. Khi đó:*

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

2.2 PHƯƠNG PHÁP BIỂU DIỄN ĐỒ THỊ

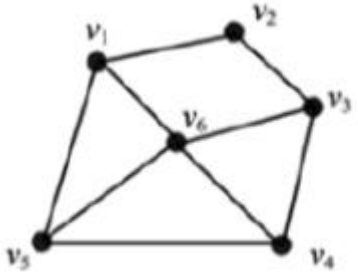
Biểu diễn đồ thị là cơ sở để định nghĩa đồ thị như một cấu trúc dữ liệu trong các ngôn ngữ lập trình, từ đó có thể triển khai các thuật toán, chương trình xử lý tính toán dựa trên đồ thị trong các ứng dụng thực tế.

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

Giả sử ta xét đơn đồ thị vô hướng *G = (V, E)*, 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.

$$a_{ij} = \begin{cases} 0, & nếu\ (i,j) \notin E \\ 1, & nếu\ (i,j) \in E \end{cases} \quad i, j = 1,2,\ldots, n.$$

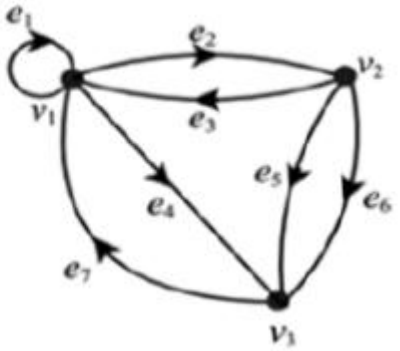


$$A_G = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$

Hình 2.11. Ma trận kề của đồ thị vô hướng G

Ta thấy, trong Hình 1.11 ma trận A_G là ma trận kề biểu diễn đồ thị vô hướng G theo cáo đỉnh 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. Ma trận kề của đồ thị có hướng G

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

Các tính chất của ma trận kề:

1) Ma trận kề của đồ thị vô hướng là ma trận đối xứng, nghĩa là
$$a[i, j] = a[i, j], i, j = 1, 2,\dots n.$$
2) Ma trận kề của đồ thị có hướng không phải là ma trân 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 x 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

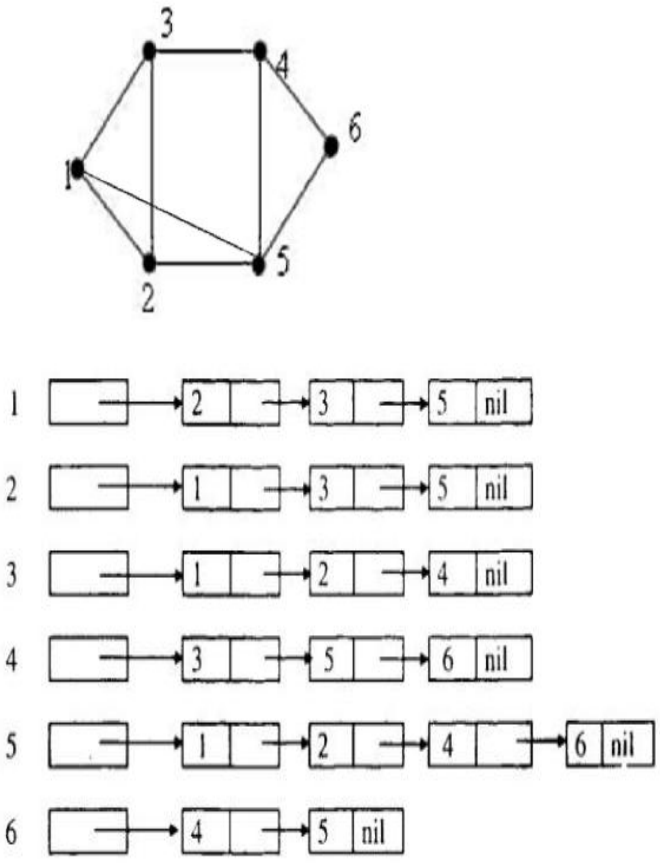
Trong quá trình biểu diễn và xử lý đồ thị, danh sách kề là một trong những cấu trúc dữ liệu được sử dụng phổ biến nhờ tính hiệu quả về bộ nhớ và khả năng truy xuất nhanh

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.

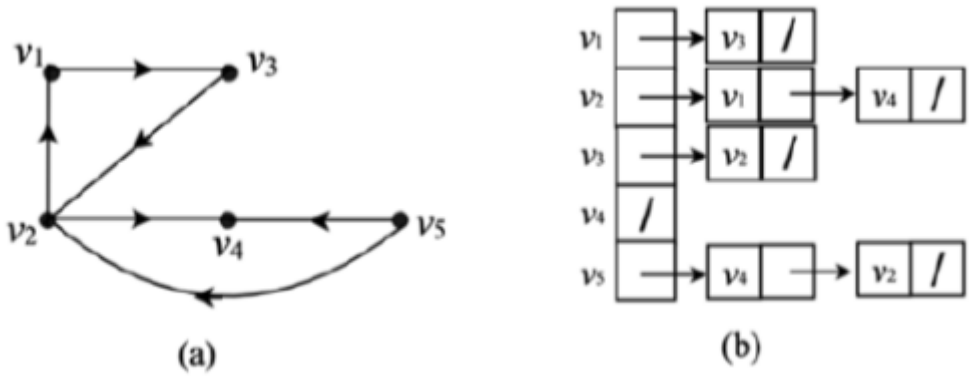
Tập cạnh của một đồ thị có thể được xác định bởi một tập các danh sách các đỉnh kề của mỗi đỉnh của đồ thị. Từ đó chúng ta có thể dùng danh sách (như là danh sách liên kết trong các ngôn ngữ lập trình) để biểu diễn đồ thị. Tập các đỉnh kề của một đỉnh của đồ thị được định nghĩa như sau.

Định nghĩa 2.2.2.1. *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*

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



Hình 2.13. Danh sách kề (phải) của đồ thị vô hướng (trái)



Hình 2.14. 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 2.15. 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.

2.4 NGÔN NGỮ LẬP TRÌNH PYTHON

2.4.1 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 2.16. 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.

2.4.2 Thư viện sử dụng chính trong Python

a) Tkinter

Tkinter là thư viện giao diện đồ họa (GUI) 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.

Tài liệu: 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 (centrality), 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 graphvi

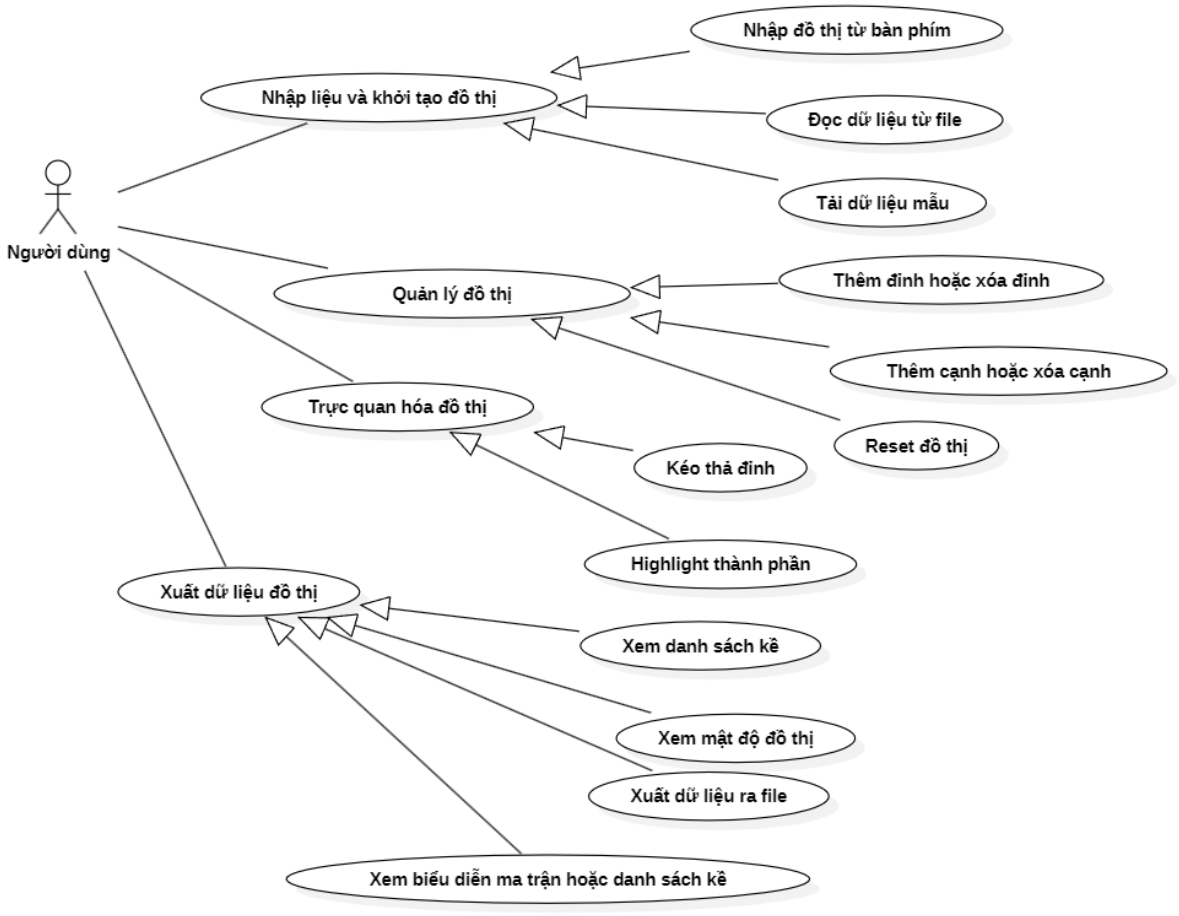
Chương 3. PHÂN TÍCH THIẾT KẾ HỆ THỐNG

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:

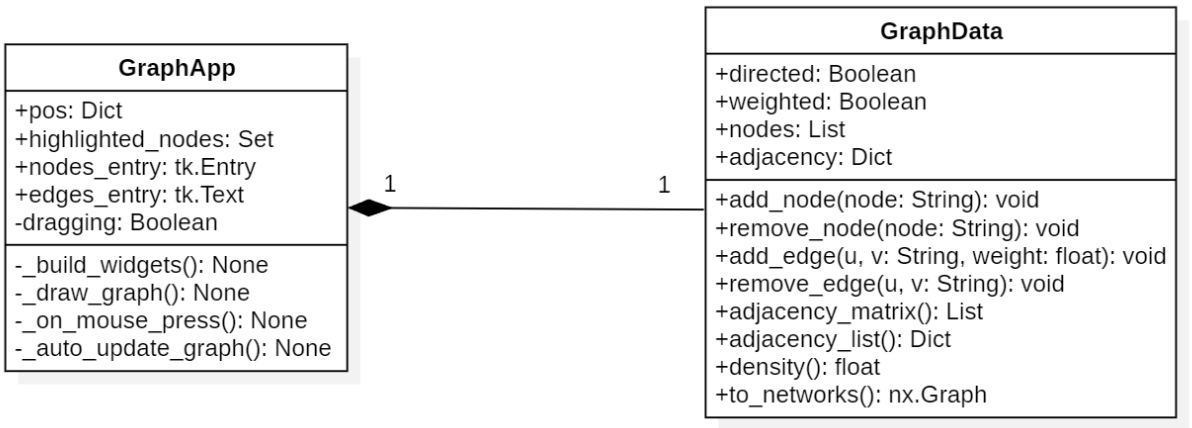
- Nhập liệu và khởi tạo đồ thị: Cho phép người dùng tạo dữ liệu đầu vào bằng nhiều cách khác nhau như nhập trực tiếp từ bàn phím, đọc dữ liệu từ tệp tin file .txt hoặc sử dụng các mẫu dữ liệu có sẵn như Karate Club.
- Quản lý đồ thị**:** Cung cấp các công cụ để thay đổi cấu trúc đồ thị hiện tại bao gồm 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.

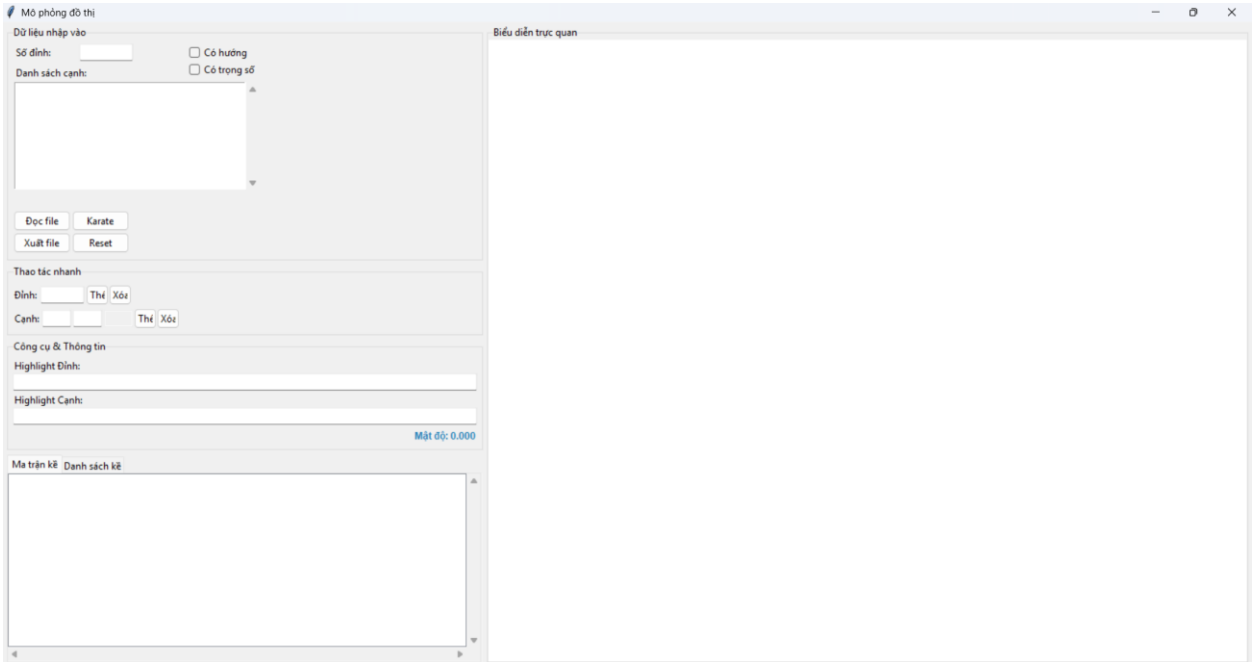
3.2.2 Thiết kế lớp chi tiết



Hình 3.2. Sơ đồ lớp đối tượng

3.4 GIAO DIỆN NGƯỜI DÙNG

3.4.1 Giao diện ban đầu



Hình 3.3. Giao diện ban đầu của người dùng

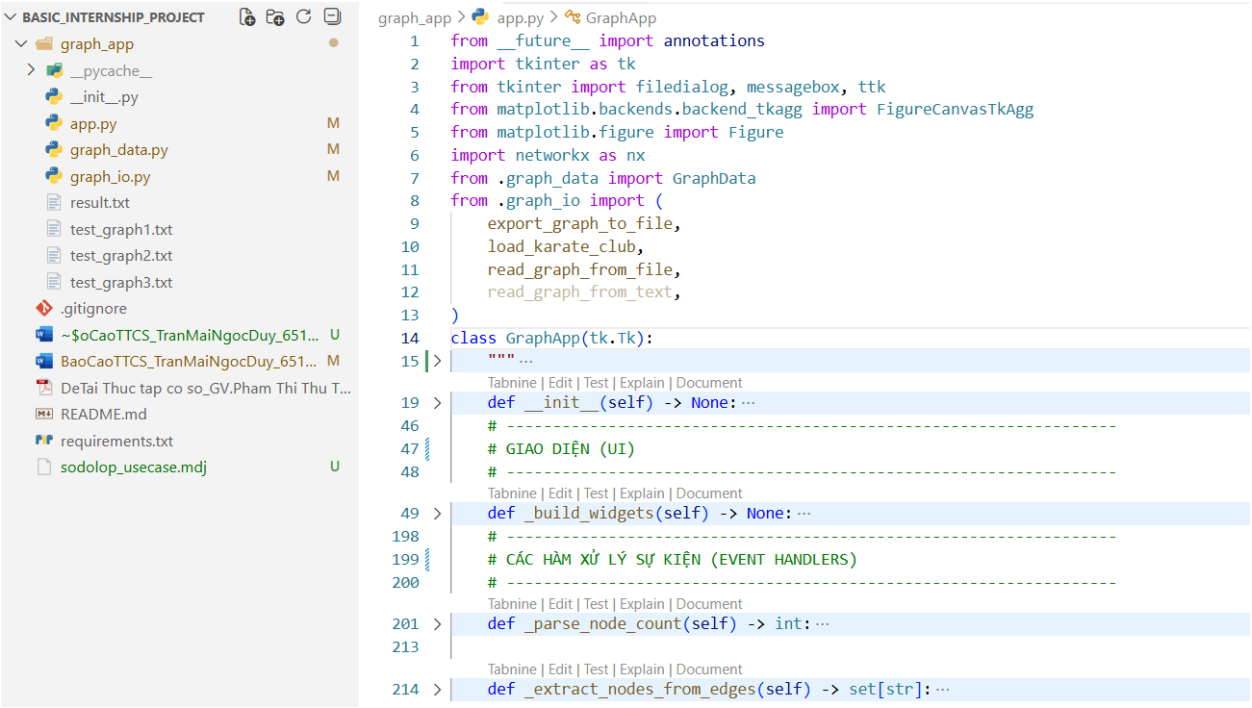
Hệ thống mô phỏng đồ thị được xây dựng nhằm hỗ trợ người dùng trong việc nhập liệu, quản lý, trực quan hóa và xuất dữ liệu đồ thị phục vụ cho học tập và nghiên cứu.

- 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.

Chương 4. CÀI ĐẶT CHƯƠNG TRÌNH

4.1 CẤU TRÚC SOURCE CODE



Hình 4.1. Cấu trúc chương trình

4.2 CODE MẪU MỘT SỐ CHỨC NĂNG CHÍNH

4.2.1 Quản lý cấu trúc đồ thị

```
Tabnine | Edit | Test | Explain | Document
def add_edge(self, u: str, v: str, weight: float = 1.0) -> None:
    """Thêm hoặc cập nhật một cạnh giữa hai đỉnh u và v."""
    self.ensure_node(u)
    self.ensure_node(v)
    w = float(weight) if self.weighted else 1.0
    self.adjacency[u][v] = w
    # Nếu là đồ thị vô hướng, thêm cạnh ngược lại
    if not self.directed:
        self.adjacency[v][u] = w
```

Hình 4.2. Xử lý thêm cạnh cho cả 2 trường hợp đồ thị có hướng và vô hướng

4.2.2 Đọc dữ liệu từ file

```
6   def read_graph_from_text(
37      # 3. Đọc danh sách các cạnh
38      edges = []
39      nodes_set = set()
40      has_weight = False
41      for i, line in enumerate(lines[2:], start=3):
42          parts = line.strip().split()
43          if len(parts) < 2:
44              # Nếu dòng chỉ có 1 phần tử, coi đó là đỉnh đơn lẻ
45              nodes_set.add(parts[0])
46              continue
47          u, v = parts[0], parts[1]
48          nodes_set.add(u)
49          nodes_set.add(v)
50          # Kiểm tra sự tồn tại của trọng số
51          if len(parts) >= 3:
52              try:
53                  weight = float(parts[2])
54                  has_weight = True
55              except ValueError:
56                  raise ValueError(f"Lỗi dòng {i}: Trọng số '{parts[2]}' không hợp lệ (phải là số).")
57          else:
58              weight = 0.0  # Gán mặc định là 0 nếu không nhập trọng số
59
60          edges.append((u, v, weight))
61
62      # Tự động gán trạng thái Weighted nếu có ít nhất một cạnh có trọng số
63      weighted = has_weight
64
65      # Tạo danh sách các đỉnh theo thứ tự nhất quán
```

Hình 4.3. Xử lý file và bóc tách chuỗi

4.2.3 Tự động cập nhật đồ thị

```
14  class GraphApp(tk.Tk):
        Tabnine | Edit | Test | Explain | Document
307     def _auto_update_graph(self) -> None:
308         """Tự động cập nhật đồ thị khi nhập danh sách cạnh"""
309         try:
310             # Xóa thông báo lỗi cũ
311             self.error_label_var.set("")
312             # Trích xuất tất cả đỉnh từ danh sách cạnh
313             edges_nodes = self._extract_nodes_from_edges()
314             if not edges_nodes:
315                 # Nếu không có cạnh nào, reset đồ thị
316                 self.graph = GraphData(
317                     directed=self.options_var["directed"].get(),
318                     weighted=self.options_var["weighted"].get(),
319                 )
320                 self.pos = None  # Reset vị trí đỉnh
321                 self._refresh_views()
322                 return
323             # Sắp xếp danh sách đỉnh
324             nodes_list = sorted(edges_nodes)
325             # Tự động cập nhật số lượng đỉnh
326             actual_count = len(nodes_list)
327             current_count = self.nodes_entry.get().strip()
328             if current_count != str(actual_count):
329                 self.nodes_entry.delete(0, tk.END)
330                 self.nodes_entry.insert(0, str(actual_count))
331             # Tự động phát hiện trọng số và validate
332             has_weight = False
333             edges_text = self.edges_entry.get("1.0", tk.END).strip()
334             line_num = 0
335             for line in edges_text.splitlines():
336                 line = line.strip()
337                 line_num += 1
338                 if not line:
339                     continue
340                 parts = line.split()
341                 if len(parts) >= 3:
```

Hình 4.4. Tự động liên tục đồ thị khi thay đổi

4.2.4 Vẽ đồ thị bằng Matplotlib

```
14   class GraphApp(tk.Tk):
683  >    def _update_highlights_from_inputs(self):···
          Tabnine | Edit | Test | Explain | Document
723      def _draw_graph(self) -> None:
724          """Vẽ đồ thị lên khung Canvas sử dụng Matplotlib và NetworkX."""
725          self.ax.clear()
726          nx_graph = self.graph.to_networkx()
727          # Chỉ tính toán lại layout nếu chưa có hoặc số đỉnh thay đổi
728          if self.pos is None or set(self.pos.keys()) != set(nx_graph.nodes()):
729              self.pos = nx.spring_layout(nx_graph, seed=42)
730          highlight_nodes, highlight_edges = self._parse_highlights()
731          # Màu nền đỉnh: luôn trắng
732          node_colors = ["white" for _ in nx_graph.nodes()]
733          # Viền đỉnh: đỉnh highlight có viền đen đậm, đỉnh thường viền xám nhạt
734          default_edge_color = "#95a5a6"
735          node_edgecolors = []
736          node_linewidths = []
737          for node in nx_graph.nodes():
738              if node in highlight_nodes:
739                  node_edgecolors.append("black")
740                  node_linewidths.append(3.0)
741              else:
742                  node_edgecolors.append(default_edge_color)
743                  node_linewidths.append(2.0)
744          # Cạnh highlight: màu đen đậm, dày hơn; cạnh thường xám nhạt, mỏng hơn
745          edge_colors = []
746          edge_widths = []
747          for u, v in nx_graph.edges():
748              is_highlighted = (u, v) in highlight_edges or (
749                  not self.graph.directed and (v, u) in highlight_edges
750              )
751              if is_highlighted:
752                  edge_colors.append("black")
753                  edge_widths.append(2.5)
754              else:
755                  edge_colors.append(default_edge_color)
756                  edge_widths.append(1.5)
```

Hình 4.5. Sử dụng thư viện Matplotlib để trực quan hóa dữ liệu

Chương 5. THỰC THI CHƯƠNG TRÌNH

Chương 6. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

6.1 KẾT QUẢ ĐẠT ĐƯỢC

6.2 NHỮNG HẠN CHẾ

6.3 HƯỚNG PHÁT TRIỂN

TÀI LIỆU THAM KHẢO

[1] https://networkx.org/en/ (NetworkX, 24/11/2025, Software for Complex Networks).

[2] https://graphviz.readthedocs.io/en/stable/index.html (Graphviz, Graph Drawing Software).

[3] https://viblo.asia/p/gioi-thieu-ve-ly-thuyet-do-thi-07LKXQ1eZV4 (Viblo, 24/01/2024, Giới thiệu về Lý thuyết đồ thị).

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

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