KHOA KỸ THUẬT VÀ CÔNG NGHỆ

**BỘ MÔN CÔNG NGHỆ THÔNG TIN**

****

THỰC TẬP ĐỒ ÁN CƠ SỞ NGÀNH

HỌC KỲ I, NĂM HỌC 2024-2025

**VIẾT CHƯƠNG TRÌNH MÔ PHỎNG GIẢI THUẬT KRUSKAL**

*Sinh viên thực hiện:*

Họ tên: Nguyễn Phi Hùng

MSSV: 110122079

Lớp: DA22TTD

***Giảng viên hướng dẫn:***

**Trầm Hoàng Nam**

***Trà Vinh, tháng 12 năm 2024***

KHOA KỸ THUẬT VÀ CÔNG NGHỆ

**BỘ MÔN CÔNG NGHỆ THÔNG TIN**

****

THỰC TẬP ĐỒ ÁN CƠ SỞ NGÀNH

HỌC KỲ I, NĂM HỌC 2024-2025

**VIẾT CHƯƠNG TRÌNH MÔ PHỎNG GIẢI THUẬT KRUSKAL**

*Sinh viên thực hiện:*

Họ tên: Nguyễn Phi Hùng

MSSV: 110122079

Lớp: DA22TTD

***Giảng viên hướng dẫn:***

**Trầm Hoàng Nam**

***Trà Vinh, tháng 12 năm 2024***

**NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN**

.............

......

*Trà Vinh, ngày ….. tháng …… năm ……*

**Giáo viên hướng dẫn**

*(Ký tên và ghi rõ họ tên)*

*Trà Vinh, ngày ….. tháng …… năm …*

**Giáo viên hướng dẫn**

*(Ký tên và ghi rõ họ tên)*

**LỜI CẢM ƠN**

**NHẬN XÉT CỦA THÀNH VIÊN HỘI ĐỒNG**

*Trà Vinh, ngày ….. tháng …… năm ……*

**Thành viên hội đồng**

*(Ký tên và ghi rõ họ tên)*

Trước tiên em xin cảm ơn thầy Trầm Hoàng Nam, giảng viên khoa Kỹ thuật và Công nghệ trường Đại học Trà Vinh đã tận tình hướng dẫn em trong việc thực hiện đồ án cơ sở ngành, qua đó tạo cho em thêm nhiều kiến thức mới, kỹ năng và tư duy lập trình. Tuy đồ án cơ sở ngành của em còn nhiều thiếu xót và em rất mong nhận được những ý kiến của quý Thầy Cô để rút ra kinh nghiệm và hoàn thiện cho các bài báo cáo trong tương lai. Và em cũng cảm ơn các Thầy Cô đã tận tụy giảng dạy để em có thể tích luỹ kiến thức cho tất cả môn học, tạo tiền đề cho chặng đường học tập phía trước.

**MỤC LỤC**

[CHƯƠNG 1: TỔNG QUAN 12](#_Toc11642)

[1.1 Giới thiệu vấn đề 12](#_Toc24567)

[1.2. Mục đích nghiên cứu 12](#_Toc31970)

[CHƯƠNG 2: NGHIÊN CỨU LÝ THUYẾT 15](#_Toc22382)

[2.1. Lý thuyết đồ thị 13](#_Toc2194)

[2.1.1. Định nghĩa đồ thị 13](#_Toc29885)

[2.1.2.Cây khung và cây khung tối thiểu (Minimum Spanning Tree - MST) 14](#_Toc11170)

[2.2. Giải thuật Kruskal 15](#_Toc26163)

[2.2.1. Khái niệm 15](#_Toc25713)

[2.2.2. Cấu trúc dữ liệu hỗ trợ: 15](#_Toc15568)

[2.2.3. Ưu điểm 15](#_Toc25498)

[2.2.5. Nhược điểm 16](#_Toc9208)

[2.3. Ngôn ngữ lập trình Python 16](#_Toc14168)

[2.3.1.Tổng quan về Python 16](#_Toc26558)

[2.4. Các thư viện hỗ trợ 19](#_Toc7982)

[2.4.1.Tkinter 19](#_Toc18020)

[2.4.2 Networkx 20](#_Toc12768)

[CHƯƠNG 3: HIỆN THỰC HÓA NGHIÊN CỨU 21](#_Toc12039)

[3.1. Quy trình nghiên cứu 21](#_Toc24216)

[3.1.1. Nghiên cứu giải thuật Kruskal 21](#_Toc5505)

[3.1.2. Xác định yêu cầu và phạm vi nghiên cứu 21](#_Toc28774)

[3.1.3. Lựa chọn công cụ và môi trường phát triển 21](#_Toc13238)

[3.2. Phân tích và thiết kế hệ thống 21](#_Toc30075)

[3.2.1. Đặc tả chức năng 21](#_Toc13819)

[3.2.2. Đặc tả phi chức năng 22](#_Toc10301)

[3.3. Cài đặt chương trình 23](#_Toc9892)

[3.3.1. Khởi tạo lớp 23](#_Toc6811)

[3.3.2. Tạo giao diện nhập thông tin đồ thị 23](#_Toc25889)

[3.3.3. Tạo ô chứa các cạnh đã nhập và kết quả cây khung nhỏ nhất 24](#_Toc30334)

[3.3.4. Vẽ đồ thị ban đầu 25](#_Toc781)

[3.3.5. Tạo các đỉnh 25](#_Toc8478)

[3.3.6. Vẽ cạnh đồ thị 26](#_Toc10912)

[3.3.6. Tìm cây khung nhỏ nhất 26](#_Toc5680)

[3.3.7. Vẽ cây khung nhỏ nhất 26](#_Toc25001)

[3.3.8. Làm mới dữ liệu 27](#_Toc8556)

[3.3.9. Khởi tạo chương trình 28](#_Toc21898)

[CHƯƠNG 4: KẾT QUẢ NGHIÊN CỨU 29](#_Toc4398)

[4.1. Giao diện chính 29](#_Toc32703)

[4.2. Nhập cạnh đồ thị 30](#_Toc4932)

[4.3. Tìm cây khung nhỏ nhất 31](#_Toc12283)

[4.4. Làm mới dữ liệu 32](#_Toc13618)

[CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 33](#_Toc12264)

[5.1. Kết luận 33](#_Toc2918)

[5.2. Hướng phát triển 33](#_Toc5629)

[DANH MỤC TÀI LIỆU THAM KHẢO 34](#_Toc24371)

**DANH MỤC HÌNH ẢNH**

[Hình 1 : Đồ thị vô hướng có trọng số 13](#_Toc18090)

[Hình 2 : Cây khung tối thiểu 14](#_Toc31982)

[Hình 3 : Logo ngôn ngữ lập trình Python 17](#_Toc23901)

[Hình 4 : Tạo giao diện nhập thông tin đồ thị 23](#_Toc18941)

[Hình 5 : Tạo giao diện nhập số đỉnh và số cạnh 23](#_Toc16024)

[Hình 6 : Tạo giao điện nhập cạnh u ,v và trọng số w 24](#_Toc2950)

[Hình 7 : Thêm cạnh u, v và trọng số w 24](#_Toc4536)

[Hình 8 : Tạo ô hiển thị đã nhập 25](#_Toc31227)

[Hình 9 : Hàm vẽ đồ thị ban đầu 25](#_Toc17674)

[Hình 10 :Hàm bố trí các đỉnh 25](#_Toc28915)

[Hình 11 : Hàm vẽ cạnh đồ thị 26](#_Toc28098)

[Hình 12 : Hàm tìm cây khung nhỏ nhất 26](#_Toc17422)

[Hình 13 : Hàm vẽ cây khung nhỏ nhất 27](#_Toc28758)

[Hình 14 : Hàm làm mới dữ liệu 27](#_Toc7148)

[Hình 15 : Hàm khởi tạo chương trình 28](#_Toc27161)

[Hình 16 : Giao diện khi bắt đầu chương trình 29](#_Toc25494)

[Hình 17 : Giao diện nhập cạnh u v và trọng số w 30](#_Toc6793)

[Hình 18 : Giao diện sau khi nhập một cạnh 30](#_Toc13221)

[Hình 19 : Giao diện sau khi đã nhập đủ cạnh 31](#_Toc9510)

[Hình 20 : Giao diện sau khi chạy giải thuật Kruskal 32](#_Toc17078)

[Hình 21 : Giao diện sau sau khi làm mới. 32](#_Toc9605)

**TÓM TẮT ĐỒ ÁN CƠ SỞ NGÀNH**

**- Vấn đề nghiên cứu:** Đồ án "**Viết chương trình mô phỏng giải thuật Kruskal**" tập trung vào việc xây dựng một chương trình máy tính mô phỏng thuật toán Kruskal để tìm cây khung nhỏ nhất (Minimum Spanning Tree - MST) trong một đồ thị liên thông có trọng số. Đây là bài toán cơ bản trong lý thuyết đồ thị, với nhiều ứng dụng thực tiễn như tối ưu hóa mạng lưới giao thông, mạng máy tính, và nhiều lĩnh vực khác. Tuy nhiên, việc hiểu và triển khai giải thuật không chỉ dừng lại ở lý thuyết mà cần phải mô phỏng và kiểm chứng hiệu quả của thuật toán Kruskal thông qua việc thực hiện và đánh giá kết quả trên các đồ thị khác nhau.

**- Các hướng tiếp cận:**

**+ Nghiên cứu lý thuyết:** Tìm hiểu nguyên lý hoạt động của giải thuật Kruskal, các khái niệm liên quan như đồ thị, cây khung, trọng số cạnh. Cách sắp xếp các cạnh của đồ thị theo thứ tự trọng số tăng dần, sử dụng cấu trúc dữ liệu Union-Find (Disjoint Set Union) để kiểm tra và hợp nhất các thành phần liên thông. Lựa chọn các cạnh không tạo chu trình và thêm vào MST.

**+ Chọn ngôn ngữ lập trình:** Sử dụng ngôn ngữ lập trình Python và thư viện Tkinter trong Python để triển khai thuật toán.

**+ Mô phỏng chương trình:** Sử dụng ngôn ngữ lập trình để xây dựng giao diện đồ họa thân thiện, cho phép người dùng nhập liệu, xuất đồ thị trên giao diện và quan sát kết quả.

**- Cách giải quyết vấn đề:**

**+ Xây dựng chương trình:** Sử dụng ngôn ngữ lập trình Python kết hợp thư viện Tkinter để thiết kế giao diện người dùng.Triển khai các bước của giải thuật Kruskal dưới dạng mô-đun, dễ dàng tích hợp và mở rộng.Cung cấp các tính năng như nhập dữ liệu đồ thị, xuất đồ thị, hiển thị trực quan quá trình sắp xếp cạnh, kiểm tra chu trình và xây dựng MST.

**+ Kiểm thử và đánh giá:** Thử nghiệm chương trình trên các bộ dữ liệu đồ thị khác nhau từ đơn giản đến phức tạp để đảm bảo tính hiệu quả và chính xác. Đánh giá hiệu suất chương trình thông qua thời gian chạy và khả năng xử lý đồ thị lớn.

**- Một số kết quả đạt được:**

**+ Chương trình hoàn chỉnh:** Giao diện đồ họa thân thiệncho phép người dùng tương tác, nhập đồ thị tùy ý và quan sát từng bước hoạt động,Hiển thị kết quả cuối cùng là cây khung, tổng trọng số và đồ thị MST của giải thuật Kruskal.

**+ Hiệu suất ổn định:** Đảm bảo chương trình hoạt động mượt mà với đồ thị có kích thước trung bình

**+ Tính giáo dục cao:** Hỗ trợ người học hiểu sâu hơn về giải thuật Kruskal thông qua minh hoạ chi tiết và dễ hiểu.

**MỞ ĐẦU**

1. **Lý do chọn đề tài**

Giải thuật Kruskal là một giải thuật quan trọng trong lý thuyết đồ thị, được sử dụng rộng rãi để tìm cây khung nhỏ nhất (Minimum Spanning Tree - MST) của một đồ thị vô hướng có trọng số. Việc hiểu và ứng dụng thuật toán này có ý nghĩa thiết thực lớn trong nhiều lĩnh vực như thiết kế mạng máy tính, xây dựng cơ sở hạ tầng, và nhiều vấn đề tối ưu hóa khác, nơi mà việc tiết kiệm chi phí và tăng cường hiệu quả là rất cần thiết.

Đề tài **“Viết chương trình mô phỏng giải thuật Kruskal”** không chỉ giúp hiểu rõ hơn về cơ chế hoạt động của thuật toán và tạo cơ hội áp dụng nó vào các bài toán thực tế mà còn nhằm mục đích giúp tôi củng cố kiến thức về cấu trúc dữ liệu, thuật toán và lập trình, đồng thời rèn luyện kỹ năng thiết kế, lập trình và kiểm thử chương trình.

1. **Mục đích nghiên cứu**

- Hiểu rõ nguyên lý hoạt động của giải thuật Kruskal, các bước thực hiện thuật toán và ứng dụng thực tế của nó.

- Xây dựng một chương trình mô phỏng trực quan quá trình tìm cây khung nhỏ nhất.

- Ứng dụng chương trình để giải các bài toán tìm cây khung nhỏ nhất trên các đồ thị khác nhau.

- Đảm bảo chương trình chạy ổn định và hiệu quả khi thực hiện các thử nghiệm.

- Phát triển kỹ năng lập trình, phân tích và giải quyết vấn đề.

1. **Đối tượng nghiên cứu**

- Giải thuật Kruskal và các cấu trúc dữ liệu liên quan, như danh sách cạnh, tập hợp rời rạc (Disjoint Set Union - DSU)

- Các công cụ lập trình và hỗ trợ việc mô phỏng (Python, thư viện đồ họa Tkinter, thư viện Networkx hỗ trợ việc xử lý đồ thị).

1. **Phạm vi nghiên cứu**

- Lý thuyết cơ bản về đồ thị vô hướng có trọng số: các khái niệm, định nghĩa và tính chất liên quan đến cây khung nhỏ nhất.

- Xây dựng và triển khai giải thuật Kruskal trên đồ thị không vô hướng có trọng số dương. Có thể giới hạn kích thước đồ thị để đảm bảo tính trực quan và hiệu quả.

- Mô phỏng các bước thực hiện của giải thuật Kruskal trên máy tính, hiển thị trực quan các trạng thái của đồ thị qua từng giai đoạn bằng ngôn ngữ lập trình Python, sử dụng thư viện đồ họa Tkinter để minh họa, và các công cụ hỗ trợ.

# CHƯƠNG 1: TỔNG QUAN

## Giới thiệu vấn đề

Trong lĩnh vực khoa học máy tính và công nghệ thông tin, các bài toán liên quan đến đồ thị đóng vai trò quan trọng, đặc biệt trong các ứng dụng thực tế như thiết kế mạng, giao thông, tối ưu hóa và các hệ thống điện. Một trong những bài toán nổi bật là tìm cây khung nhỏ nhất (Minimum Spanning Tree - MST), có vai trò tối ưu hóa chi phí trong việc kết nối các nút của một đồ thị vô hướng có trọng số.

Giải thuật Kruskal là một trong những phương pháp hiệu quả để giải quyết bài toán này. Với cách tiếp cận dựa trên chiến lược tham lam (greedy), giải thuật này giúp tìm ra cây khung nhỏ nhất bằng cách sắp xếp các cạnh theo trọng số tăng dần và lần lượt chọn các cạnh phù hợp mà không tạo thành chu trình.

Việc nghiên cứu và mô phỏng chương trình giải thuật Kruskal có ý nghĩa quan trọng vì:

- Giúp sinh viên hiểu sâu hơn về cách thức hoạt động của thuật toán Kruskal và ứng dụng của nó.

- Cung cấp công cụ học tập trực quan để minh họa quá trình tìm kiếm cây khung tối thiểu.

- Rèn luyện kỹ năng lập trình và tư duy logic cho sinh viên.

- Tạo cơ hội áp dụng kiến thức lý thuyết vào thực tiễn thông qua việc xây dựng chương trình mô phỏng.

## 1.2. Mục đích nghiên cứu

- Hiểu rõ cơ chế hoạt động của giải thuật Kruskal thông qua việc phân tích lý thuyết và thực nghiệm.

- Phát triển kỹ năng lập trình.

- Ứng dụng giải thuật Kruskal vào các bài toán thực tế, từ đó nhận diện lợi ích và hạn chế của giải thuật.

- Hỗ trợ người dùng dễ dàng tiếp cận và hiểu sâu hơn về cách vận hành của thuật toán thông qua việc tương tác trực tiếp với mô phỏng và có ánh nhìn trực quan.

## 2.1. Lý thuyết đồ thị

### 2.1.1. Định nghĩa đồ thị

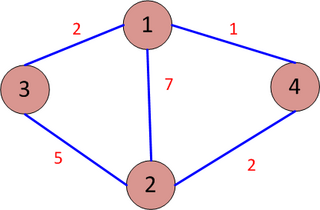
**- Đồ thị G**=(V,E)G = (V, E)G=(V,E) là một cấu trúc bao gồm:

**+ Tập hợp V**: Các đỉnh (vertices).

**+ Tập hợp E:** Các cạnh (edges), mỗi cạnh là một cặp đỉnh (u,v) (u, v) (u,v) với u,v∈Vu, v \in Vu,v∈V.

**- Đồ thị vô hướng:** Là đồ thị mà các cạnh không có hướng.

**- Đồ thị có trọng số:** Là đồ thị mà mỗi cạnh được gán một giá trị số, thường gọi là trọng số.



Hình 1: Đồ thị vô hướng có trọng số

**- Trọng số của cạnh w(u,v):** Giá trị gắn liền với cạnh, có thể biểu diễn chi phí hoặc độ dài.

- **Các ứng dụng thực tế**

**+ Hệ thống mạng máy tính:** Tìm đường đi tối ưu, phát hiện lỗi trong mạng.

**+ Quản lý giao thông:** Tìm tuyến đường ngắn nhất, tối ưu hóa thời

gian chuyển.

**+ Mạng xã hội:** Phân tích kết nối, tìm các cộng đồng.

**+ Lập lịch và tối ưu hóa:** Giải quyết bài toán phân bổ tài nguyên.

****2.1.2. Cây khung và cây khung tối thiểu (Minimum Spanning Tree - MST)****

- **Cây khung (Spanning Tree)**:

+ Là một cây con của đồ thị bao gồm tất cả các đỉnh của đồ thị và một số cạnh sao cho không tạo thành chu trình.

+ Nếu đồ thị ban đầu có nnn đỉnh, cây khung sẽ có đúng n−1n-1n−1 cạnh.

- **Cây khung tối thiểu (MST)**:

+ Là cây khung trong đó tổng trọng số các cạnh là nhỏ nhất.

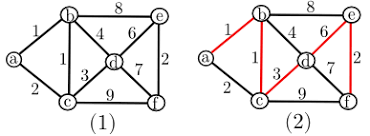
+ Một đồ thị liên thông vô hướng luôn tồn tại ít nhất một cây khung tối thiểu.

- **Ứng dụng của cây khung tối thiểu**:

+ Thiết kế mạng lưới kết nối (mạng điện, mạng viễn thông).

+ Tối ưu hóa đường truyền trong mạng máy tính.

+ Phân vùng trong các thuật toán xử lý hình ảnh.



Hình 2: Cây khung tối thiểu

# CHƯƠNG 2: NGHIÊN CỨU LÝ THUYẾT

## 2.2. Giải thuật Kruskal

2.2.1. Khái niệm

- **Nguyên lý hoạt động**: Thuật toán Kruskal xây dựng cây khung nhỏ nhất bằng cách lần lượt thêm các cạnh có trọng số nhỏ nhất vào cây khung, với điều kiện là việc thêm cạnh đó không tạo thành chu trình.

+ Sắp xếp các cạnh của đồ thị theo thứ tự tăng dần của trọng số.

+ Khởi tạo một tập hợp rừng gồm các cây đơn, mỗi cây chứa một đỉnh.

+ Duyệt qua danh sách các cạnh đã sắp xếp, lần lượt thêm cạnh vào

cây khung nếu nó không tạo thành chu trình, ngược lại bỏ qua cạnh đó

+ Lặp lại cho đến khi cây khung có đủ ∣V∣−1|V| - 1∣V∣−1 cạnh.

+ Kết quả sẽ là cây khung nhỏ nhất với tổng trọng số của nó

### 2.2.2. Cấu trúc dữ liệu hỗ trợ:

**- Disjoint Set Union (DSU)**:DSU là cấu trúc dữ liệu hỗ trợ kiểm tra và hợp nhất các tập hợp để xác định các thành phần liên thông.

+ Có hai thao tác quan trọng:

* Find(u): Tìm tập hợp chứa đỉnh u.
* Union(u, v): Hợp nhất hai tập hợp chứa u và v.

+ Tối ưu bằng kỹ thuật "nén đường dẫn" và "hợp theo hạng".

**- Sắp xếp các cạnh:** O(ElogE), với E là số cạnh trong đồ thị

**- Union-Find:** Mỗi thao tác **find** và **union** có độ phức tạp gần O(α(V)), với α là hàm đảo ngược của hàm Ackermann (hầu như là hằng số nhỏ).

- **Tổng độ phức tạp:** O(ElogE+Eα(V)), trong đó E là số cạnh và V là số đỉnh.

### 2.2.3. Ưu điểm

- Giải thuật Kruskal rất hiệu quả đối với đồ thị thưa (số cạnh ít so với số đỉnh).

- Ý tưởng của thuật toán khá đơn giản và dễ hiểu, việc triển khai bằng code cũng không quá phức tạp.

- Quy trình của Kruskal rất trực quan (sắp xếp cạnh và thêm cạnh nếu không tạo chu trình). Điều này làm cho thuật toán dễ lập trình hơn so với các thuật toán khác như Prim.

- Kruskal không yêu cầu chọn đỉnh gốc, do đó thuận lợi khi áp dụng trên đồ thị mà không cần quan tâm vị trí của các đỉnh.

- Sử dụng Union-Find để kiểm tra chu trình giúp thuật toán hoạt động nhanh và hiệu quả, đặc biệt khi thực hiện nhiều lần trên các đồ thị khác nhau.

### 2.2.5. Nhược điểm

- Khi đồ thị có số cạnh rất lớn, bước sắp xếp cạnh

sẽ tốn kém hơn so với giải thuật Prim với ma trận trọng số.

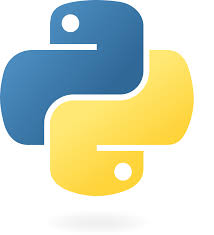
- Thuật toán cần sử dụng bộ nhớ để lưu trữ danh sách các cạnh.

- Để kiểm tra chu trình, giải thuật Kruskal yêu cầu triển khai Union- Find, điều này có thể phức tạp hơn đối với người mới học hoặc khi làm việc với đồ thị lớn.

## 2.3. Ngôn ngữ lập trình Python

### 2.3.1.Tổng quan về Python

Python là một ngôn ngữ lập trình bậc cao,với thiết kế cú pháp rõ ràng, linhoạt, đơn giản và đa năng, giúp các lập trình viên tập trung vào giải quyết vấn đề hơn là các cấu trúc ngôn ngữ phức tạp. Python được sử dụng rộng rãi trong nhiều lĩnh vực, từ phát triển web, khoa học dữ liệu, trí tuệ nhân tạo cho đến tự động hóa và nhiều hơn nữa. Python đã nhanh chóng thu hút sự chú ý rộng rãi và trở thành một trong những ngôn ngữ lập trình phổ biến nhất trên thế giới.



Hình 3: Logo ngôn ngữ lập trình Python

**- Lịch sử hình thành và phát triển:**

**+ Người sáng lập:** ngôn ngữ lập trình Python được tạo ra bởi **Guido van Rossum** vào tháng 12 năm 1989 tại Cebtrum Wiskunde & Informatica (CWI) ở Hà Lan.

**+ Mục tiêu ban đầu:** **Guido van Rossum muốn tạo ra một ngôn ngữ lập trình dễ đọc, dễ học và dễ viết, phù hợp cho cả người mới bắt đầu và lập trình viên chuyên nghiệp**

**+ Nguồc gốc tên gọi : tên “Python” được lấy cảm hứng từ nhóm hài kịch Anh “Monty Python’s Flying Circus” mà Guido yêu thích, chứ không phải tên từ loài rắn.**

**+ Các giai đoạn phát triển**

* **1991: Python 0.9.0**
* **Đây là phiên bản đầu tiên của Python được phát hành vào tháng 2 năm 1991.**
* **Hỗ trợ các tính năng cơ bản như: hàm, vòng lặp, xử lý ngoại lệ, module, và một số kiểu dữ liệu như str, list, dict, tuple.**
* **1994: Python 1.0**
* **Phiên bản ổn định đầu tiên ra mắt vào tháng 1 năm 1994.**
* **Các tính năng mới như: Hỗ trợ lập trình hướng đối tượng, hàm lambda, cấu trúc lớp(class),** hệ thống xử lý lỗi ngoại lệ.
* Đây là phiên bản đầu tiên thu hút sự chú ý rộng rãi trong cộng đồng lập trình.
* **2000: Python 2.0**
* **Python 2.0 được ra mắt vào tháng 10 năm 2000.**
* **Ra mắt với nhiều cải tiến, nổi bật như:**
* **List comprehensions:** Viết danh sách một cách gắn gọn hơn.
* **Garbage collection:** Quản lý bộ nhớ tự động bằng cách dùng thuật toán tham chiếu.
* Mặc dù Python 2.0 được sử dụng rộng rãi, nhưng nó đã dần bị thây thế bởi Python 3.0 do những hạn chế về tính tương thích ngược.
* **2008: Python 3.0**
* **Python 3.0 được phát hành vào tháng 12 năm 2008.**
* **Đây là phiên bản mang tính cách mạng trong lịch sử Python, và không tương thích ngược với Python 2.0**
* **Các cải tiến lớn của Python 3.0:**
* **Thống nhất các kiểu chuỗi ( str và unicode)**
* **Thay đổi cú pháp in: Print() trở thành một hàm thay vì câu lệnh.**
* **Loại bỏ các tính năng cũ và lỗi thời.**
* **2010: Python trở thành ngôn ngữ hàng đầu trong các lĩnh vực:**
* **Phân tích dữ liệu: với các thư viện như Pandas, Numpy.**
* **Trí tuệ nhân tạo(AI) và học máy:** TensorFlow, PyTorch, Scikit-learn.
* **Phát triển web:** Django, Flask.
* **Xử lý ngôn ngữ tự nhiên:** NLTK, SpaCy.
* Tự động hóa và DevOps
* **2020: Python ngừng hỗ trợ Python 2.0**
* Python 2.0 chính thức ngừng hỗ trợ vào ngày **1 tháng 1 năm 2020**.
* Tất cả hệ thống và dự án sẽ được hoàn toàn chuyển sang Python 3.0.
* **2024: Python tiếp tục dẫn đầu trong các bảng xếp hạng ngôn ngữ lập trình và là một trong những ngôn ngữ lập trình được sử dụng phổ biến nhất.**

**- Những yếu tố đóng góp vào thành công của Python**

**+ Cú pháp đơn giản và dễ đọc: G**iúp các lập trình viên dễ dàng học và sử dụng.

+ T**hư viện phong phú:** Python có một hệ sinh thái thư viện khổng lồ, bao gồm NumPy, Pandas, Networkx, Matplotlib, TensorFlow, và nhiều thư viện khác, hỗ trợ các ứng dụng trong nhiều lĩnh vực khác nhau.

+ **Đa nền tảng:** Python có thể chạy trên nhiều hệ điều hành khác nhau, từ Windows đến Linux và macOS.

+ **Cộng đồng lớn:** Python có một cộng đồng sử dụng toàn cầu đóng góp nhiều thư viện mã nguồn mở.

## 2.4. Các thư viện hỗ trợ

### 2.4.1.Tkinter

**- Định nghĩa:** Tkinkerlà một thư viện chuẩn đi kèm với Python,giúp tạo giao diện đồ họa người dùng GUI(Graphical User Interface)

**- Các thành phần cơ bản của Tkinter**

**+ Cửa sổ chính(Tk)**

* Đây là cửa sổ gốc nơi các thành phần khác sẽ được thêm vào.
* Dùng phương thức mainloop() để bắt đầu vòng lặp xử lý sự kiện GUI.

**+ Widget(Các thành phần giao diện)**

* **Label:** Hiển thị văn bản.
* **Button:** Nút bấm để thực hiện các hành động.
* **Entry:** Ô nhập liệu cho người dùng nhập dữ liệu.
* **Text:** Vùng nhập văn bản đa dòng.
* **Canvas:** Khu vực để vẽ đồ họa hoặc đặt các widget khác.

**- Ưu và nhược điểm của Tkinter**

**+ Ưu điểm:**

* Tích hợp sẵn không cần cài đặt thêm.
* Đơn giản và phù hợp cho người mới bắt đầu.
* Chạy trên đa nền tảng như Window, macOS, và Linux.

**+ Nhược điểm:**

* Giao diện Tkinter khá đơn giản, không hiện đại và hấp dẫn như các framework khác như Qt, Kivy.
* Không phù hợp cho các ứng dụng lớn và phức tạp.
* Khả năng xử lý đồ họa của Tkinter không được mạnh mẽ.

### 2.4.2 Networkx

**- Định nghĩa: Networkx** Là một thư viện mạnh mẽ dùng để tạo, thao tác và phân tích các đồ thị (graph) và mạng lưới(network). Được sử dụng rộng rãi trong các lĩnh vực như khoa học dữ liệu, phân tích mạng xã hội, mạng giao thông, mạng máy tính, nghiên cứu đường đi, tối ưu hóa, và các mô hình hệ thống phức tạp.

+ Các loại đồ thị trong Networkx:

* Graph: Đồ thị vô hướng.
* DiGraph: đồ thị có hướng.
* MultiGraph: Đồ thị vô hướng với nhiều cạnh giữa hai nút.
* MultiDiGraph: Đồ thị có hướng với nhiều cạnh giữa hai nút.

+ Ưu và nhược điểm của Networkx:

* Ưu điểm:
* Cú pháp trực quan, dễ hiểu
* Tích hợp nhiều thuật toán phân tích đồ thị.
* Hỗ trợ nhiều loại đồ thị với kích thước lớn vừa phải.
* Nhược điểm:
* Hiệu suất không tốt với đồ thị cực lớn
* Không tối ưu cho các phép toán song song.

**CHƯƠNG 3: HIỆN THỰC HÓA NGHIÊN CỨU**

## 3.1. Quy trình nghiên cứu

### 3.1.1. Nghiên cứu giải thuật Kruskal

- Nghiên cứu chi tiết về giải thuật Kruskal

+ Tìm hiểu về các khái niệm liên quan đến đồ thị: đồ thị vô hướng, đồ thị có trọng số, cây khung nhỏ nhất (Minimum Spanning Tree - MST).

+Hiểu rõ các tính chất của cây khung nhỏ nhất, cách xác định cây khung nhỏ nhất từ đồ thị.

+ Xem xét các cấu trúc dữ liệu phù hợp (danh sách cạnh, Disjoint Set - Union- Find).

- Phân tích giải thuật Kruskal và cách thức hoạt động

+ Giải thuật Kruskal hoạt động dựa trên nguyên lý tham lam (greedy), chọn các cạnh có trọng số nhỏ nhất theo thứ tự không tạo chu trình.

+ Sử dụng cấu trúc dữ liệu **Disjoint Set (Union-Find)** để kiểm tra và hợp nhất các tập hợp đỉnh, đảm bảo không có chu trình.

### 3.1.2. Xác định yêu cầu và phạm vi nghiên cứu

- Xác định các chức năng cần thiết như nhập dữ liệu đồ thị, thực hiện thuật toán, và hiển thị kết quả.

- Đặc tả yêu cầu phi chức năng như hiệu suất, khả năng xử lý đồ thị lớn, và tính thân thiện với người dùng.

- Đồ thị đầu vào là đồ thị vô hướng, có trọng số.

- Số đỉnh đồ thị tối đa được hỗ trợ: 40.

- Nhập liệu từ giao diện trực tiếp hoặc

### 3.1.3. Lựa chọn công cụ và môi trường phát triển

- Ngôn ngữ sử dụng để viết chương trình là ngôn ngữ Python vì dễ triển khai và có nhiều thư viện hỗ trợ trực quan hóa.

- Thư viện hỗ trợ gồm:

+ Tkinter: Tạo giao diện người dùng

+ Networkx: Mô hình hóa và sử lý đồ thị

+ Matplolib: Vẽ đồ thị trực quan.

- Môi trường phát triển: Visual Studio Code.

## 3.2. Phân tích và thiết kế hệ thống

3.2.1. Đặc tả chức năng

- **Nhập dữ liệu đồ thị**:

+ Người dùng có thể nhập đồ thị thông qua:.

+ Giao diện nhập liệu trực tiếp

+ Dữ liệu đồ thị bao gồm danh sách các đỉnh, cạnh và trọng số tương ứng.

**Ví dụ dữ liệu đầu vào:**

5 6

1 2 3

1 3 4

2 3 1

3 4 2

4 5 6

5 1 5

- **Hiển thị thông tin đồ thị**:

+ Hiển thị danh sách các đỉnh, cạnh và trọng số tương ứng.

+ Hiển thị đồ thị dưới dạng minh họa trực quan

- **Thực hiện giải thuật Kruskal**:

+ Sắp xếp các cạnh theo trọng số tăng dần.

+ Lần lượt thêm các cạnh vào cây khung không tạo chu trình

+ Ghi lại trạng thái của cây khung tại mỗi bước xử lý.

- **Hiển thị kết quả**:

+ Danh sách các cạnh thuộc cây khung nhỏ nhất.

+ Tổng trọng số của cây khung.

+ Minh họa trực quan cây khung nhỏ nhất trên đồ thị.

Ví dụ kết quả đầu ra:

2 - 3: 1

3 - 4: 2

1 - 2: 3

4 - 5: 6

Tổng trọng số: 12

- **Xuất kết quả**:

+ Xuất danh sách các cạnh thuộc cây khung nhỏ nhất và tổng trọng số ra tệp.

## 3.2.2. Đặc tả phi chức năng

**-** Đảm bảo thời gian thực thi nhanh chóng, ngay cả với đồ thị có số đỉnh và số cạnh lớn (hàng nghìn đỉnh/cạnh).

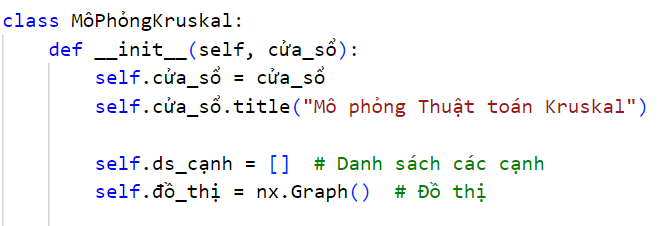
· - Tối ưu hóa thuật toán để đạt độ phức tạp \O(ElogV, trong đó E là số cạnh và V là số đỉnh.

## 3.3. Cài đặt chương trình

### 3.3.1. Khởi tạo lớp

- Lớp Mô Phỏng Kruskal được khởi tạo với cửa sổ chính cửa\_sổ, trong đó tạo các widget như các trường nhập liệu cho số lượng đỉnh và cạnh, các nút để tạo các cạnh và tìm MST.

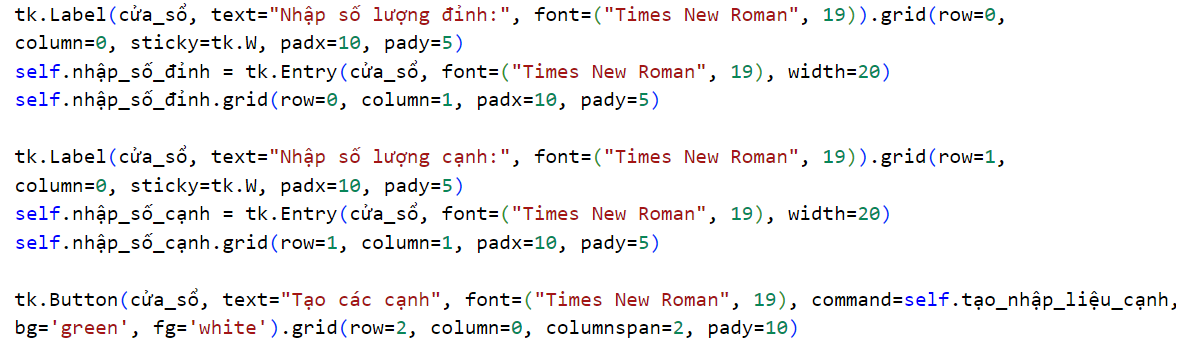
- Các đối tượng cần thiết như danh sách cạnh, đồ thị và khung vẽ được khởi tạo trong \_\_init\_\_.



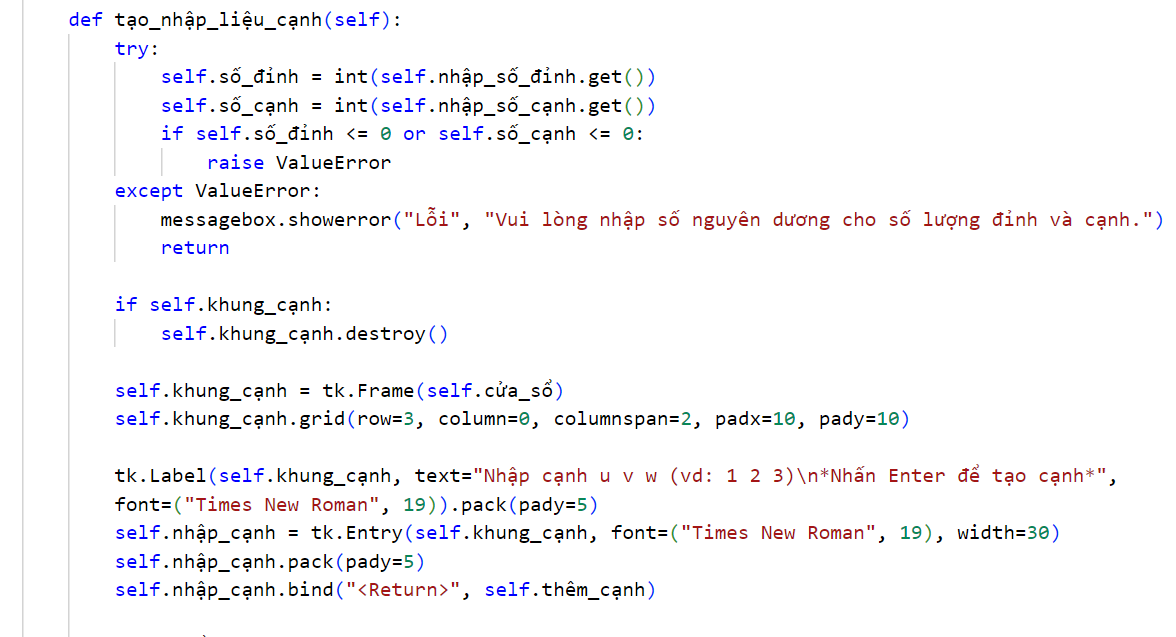
Hình 4: Tạo giao diện nhập thông tin đồ thị

- Giao diện cho phép người dùng nhập số lượng đỉnh và cạnh của đồ thị, và sau đó nhập các cạnh với trọng số theo định dạng u v w (với u, v là đỉnh và w là trọng số).

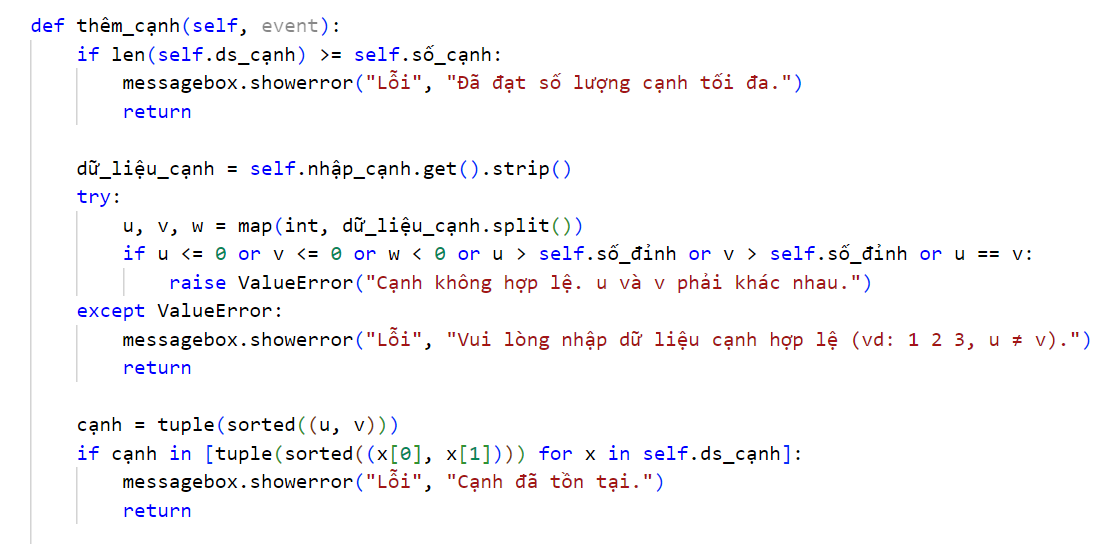
- Nút “Tạo cách cạnh” khi được nhấn sẽ mở ra giao diện nhập cạnh u, v và trọng số w, đồng thời sẽ vẽ đỉnh trên giao diện chứa đồ thị



Hình 5: Tạo giao diện nhập số đỉnh và số cạnh



Hình 6: Tạo giao điện nhập cạnh u ,v và trọng số w

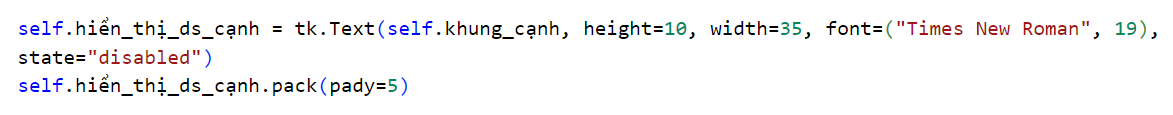


Hình 7: Thêm cạnh u, v và trọng số w

3.3.3. Tạo ô chứa các cạnh đã nhập và kết quả cây khung nhỏ nhất

- Hàm tạo ô chứa các cạnh đã nhập sẽ hiển thị lần lượt cách cạnh đã thêm đễ dễ quản lý cạnh

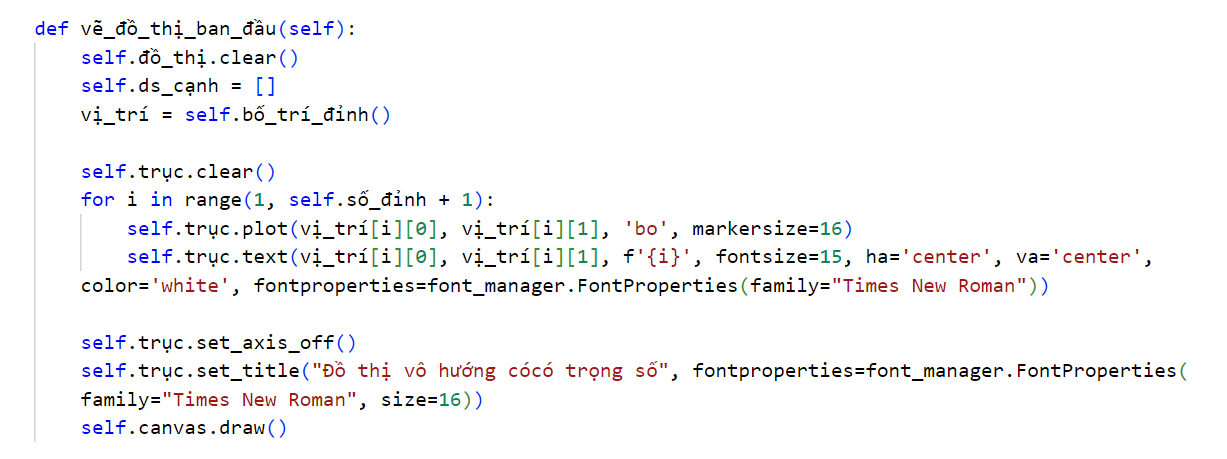
- Hàm tạo ô chứa kết quả cây khung nhỏ nhất sẽ hiển thị kết quả là các cạnh trong cây khung nhỏ nhất và tổng trọng số của nó.



Hình 8: Tạo ô hiển thị đã nhập

3.3.4. Vẽ đồ thị ban đầu

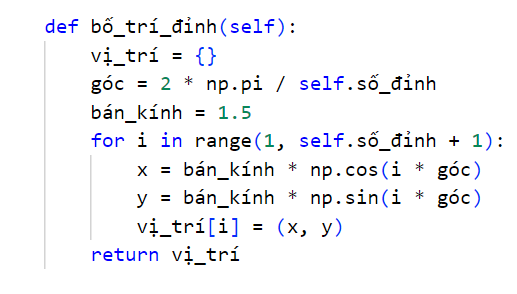
**-**Hàm này vẽ đồ thị ban đầu, trong đó các đỉnh được sắp xếp theo hình tròn và các cạnh chưa được vẽ.



Hình 9: Hàm vẽ đồ thị ban đầu

3.3.5. Tạo các đỉnh

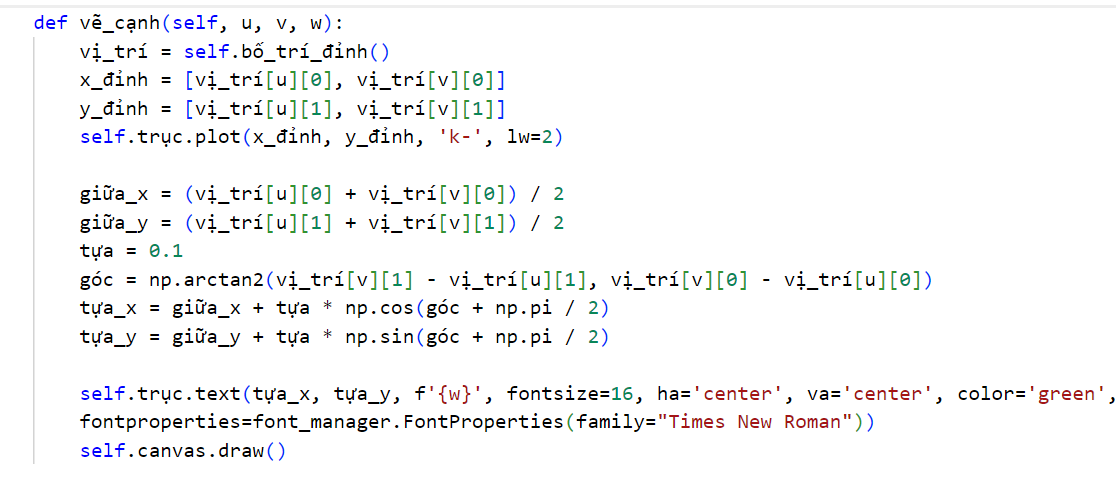
**-** Hàm này tạo ra vị trí cho các đỉnh trong bố cục hình tròn, dùng để trực quan hóa.



Hình 10: Hàm bố trí các đỉnh

### 3.3.6. Vẽ cạnh đồ thị

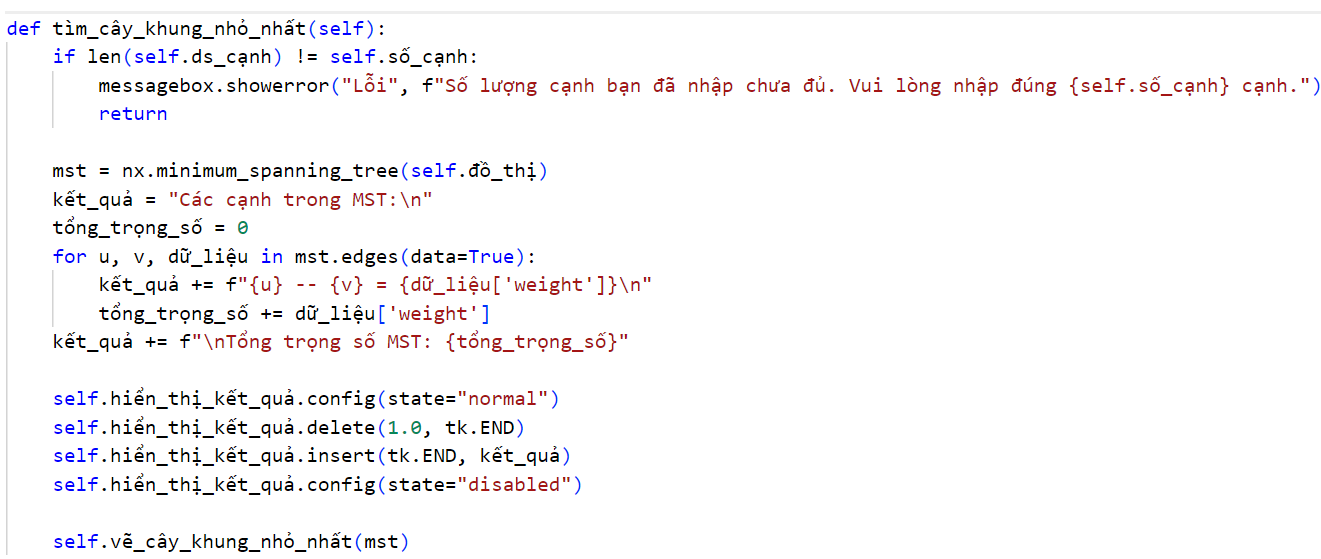
**-** Sử dụng matplotlib để vẽ cạnh đồ thị ( đường kẻ kết nối u và v)



Hình 11: Hàm vẽ cạnh đồ thị

3.3.6. Tìm cây khung nhỏ nhất

**-** Hàm này tính toán Cây Khung Nhỏ Nhất (MST) bằng Thuật toán Kruskal và hiển thị kết quả trong ô hiển thị kết quả. Nó cũng vẽ MST trên một khung vẽ riêng.



Hình 12: Hàm tìm cây khung nhỏ nhất

3.3.7. Vẽ cây khung nhỏ nhất

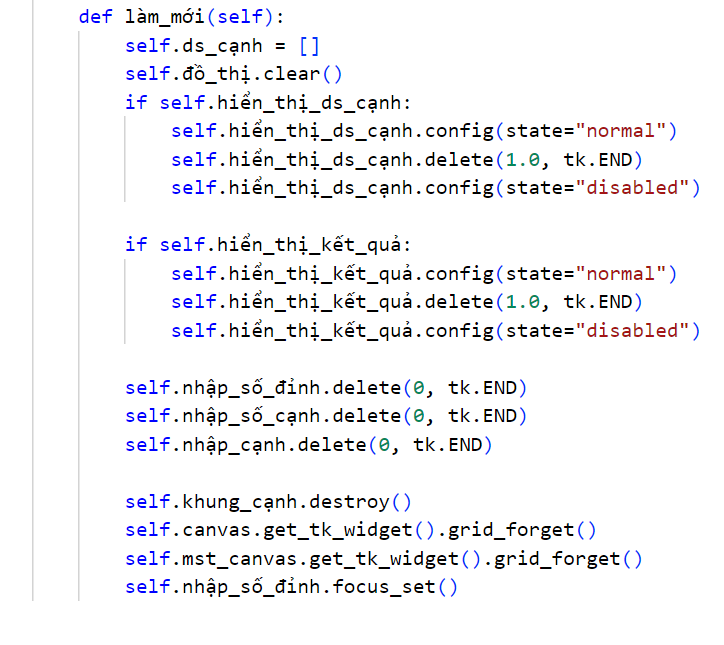
- Thiết kê hàm trực quan hóa MST bằng cách chỉ vẽ các cạnh là một phần của MST bằng màu đỏ và các cạnh khác bằng màu đen.



Hình 13: Hàm vẽ cây khung nhỏ nhất

3.3.8. Làm mới dữ liệu

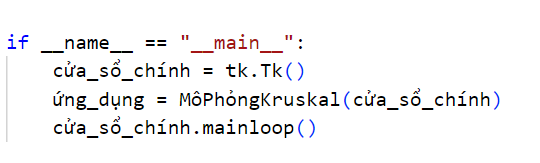
- Thiết kế hàm làm\_mớidọn dẹp tất cả các dữ liệu đầu vào, xóa các cạnh đã nhập và vẽ lại sơ đồ từ đầu.



Hình 14: Hàm làm mới dữ liệu

3.3.9. Khởi tạo chương trình

- Phần khởi động này tạo chính cửa sổ chạy ứng dụng và vòng lặp chính của Tkinter ( mainloop) để hoạt động giao diện người dùng.

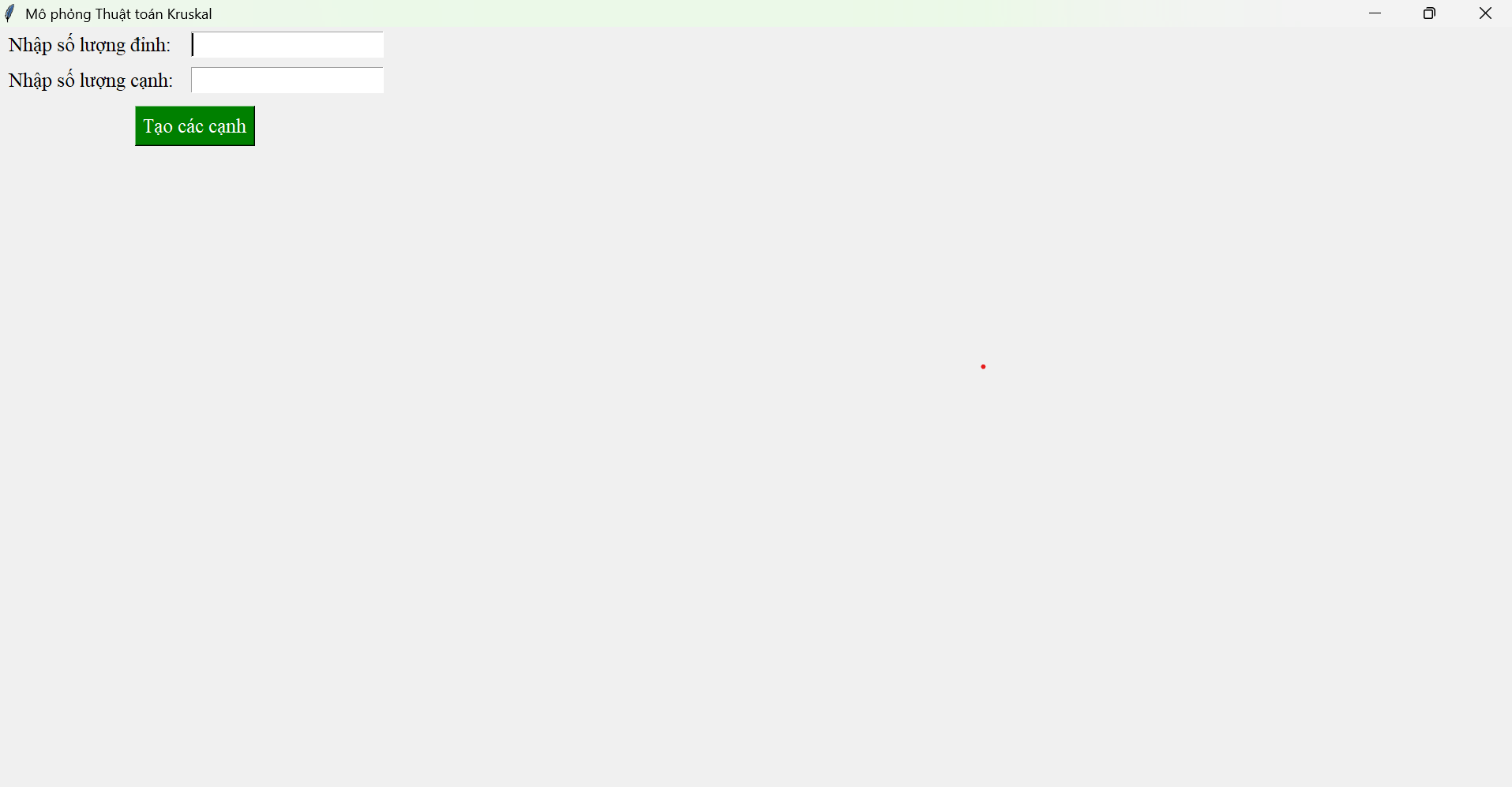


Hình 15: Hàm khởi tạo chương trình

CHƯƠNG 4: KẾT QUẢ NGHIÊN CỨU

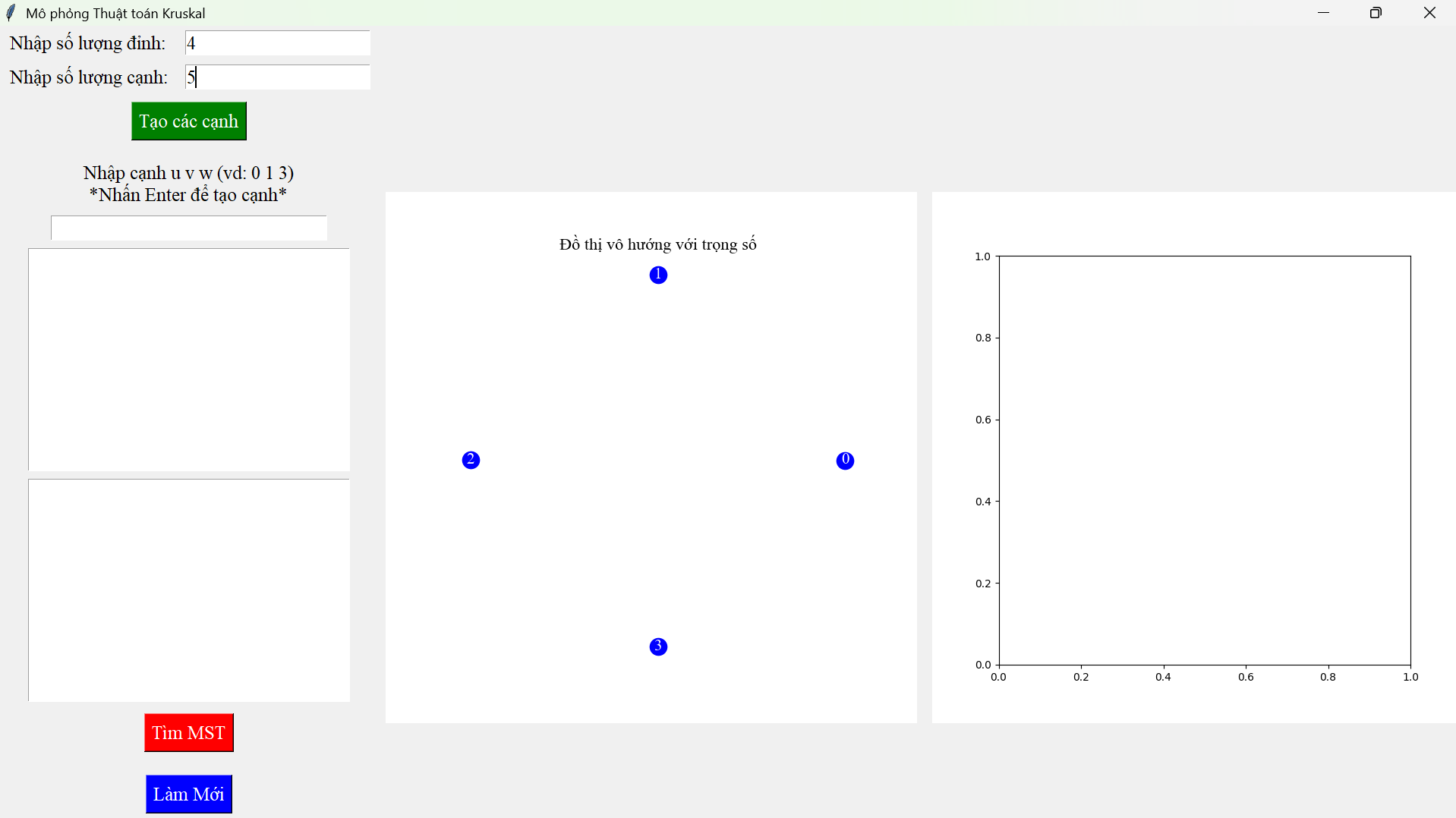
## 4.1. Giao diện chính

Giao diện chính của chương trình cho phép người dùng thao tác với các chức năng cơ bản như nhập dữ liệu đầu vào, thực thi giải thuật Kruskal, hiển thị đồ thị và hiển thị kết quả.



Hình 16: Giao diện khi bắt đầu chương trình

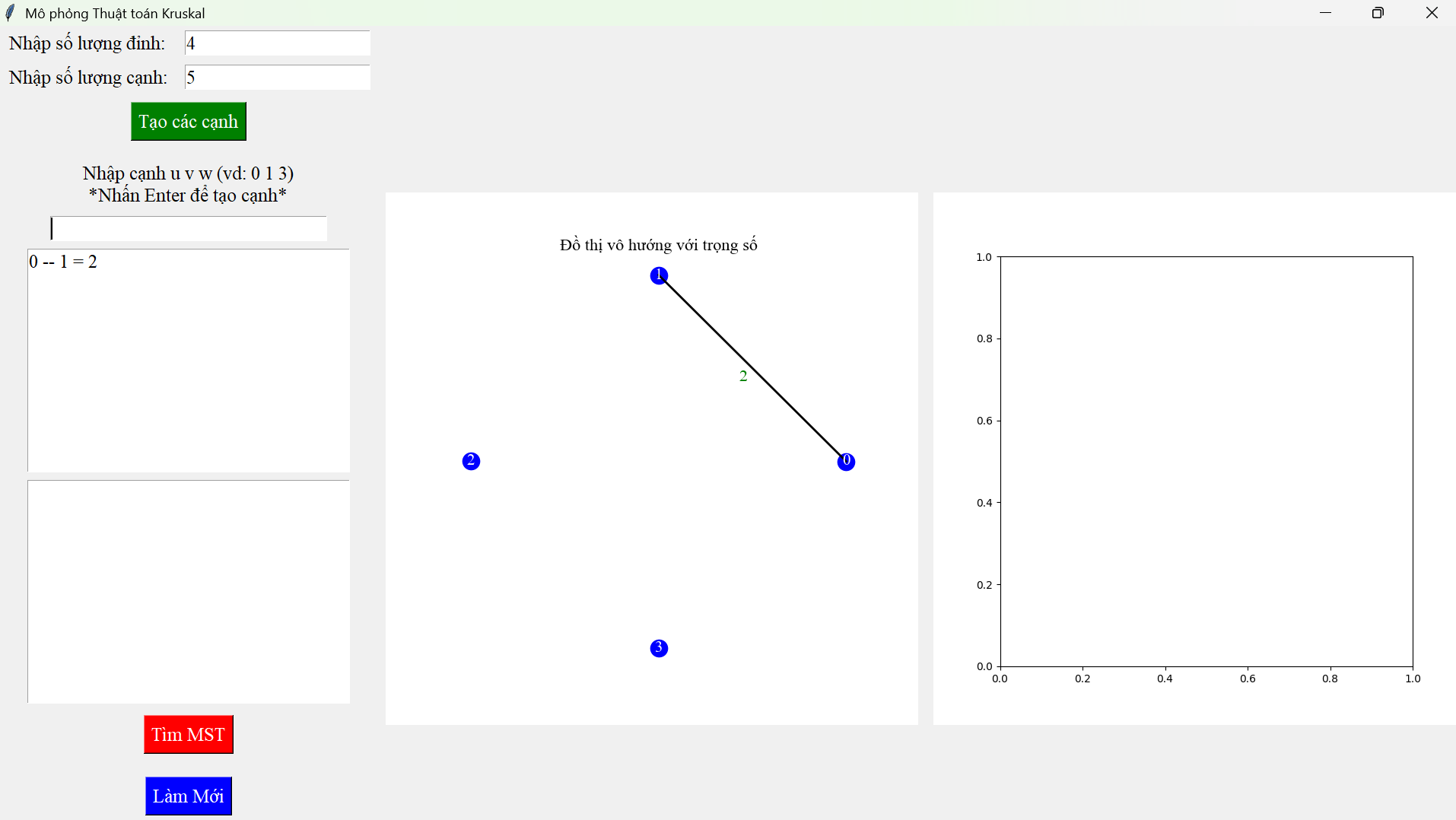
- Sau khi đã nhập số lượng đỉnh và cạnh và nhấn vào nút “Tạo các cạnh” thì sẽ hiện ra giao diện nhập cạnh các ô hiển thị và hiển thị đỉnh đã tạo. Ví dụ: nếu nhập số lượng đỉnh là 4 và số lượng cạnh là 5 thì ô đồ thị sẽ tạo 4 đỉnh.



Hình 17: Giao diện nhập cạnh u v và trọng số w

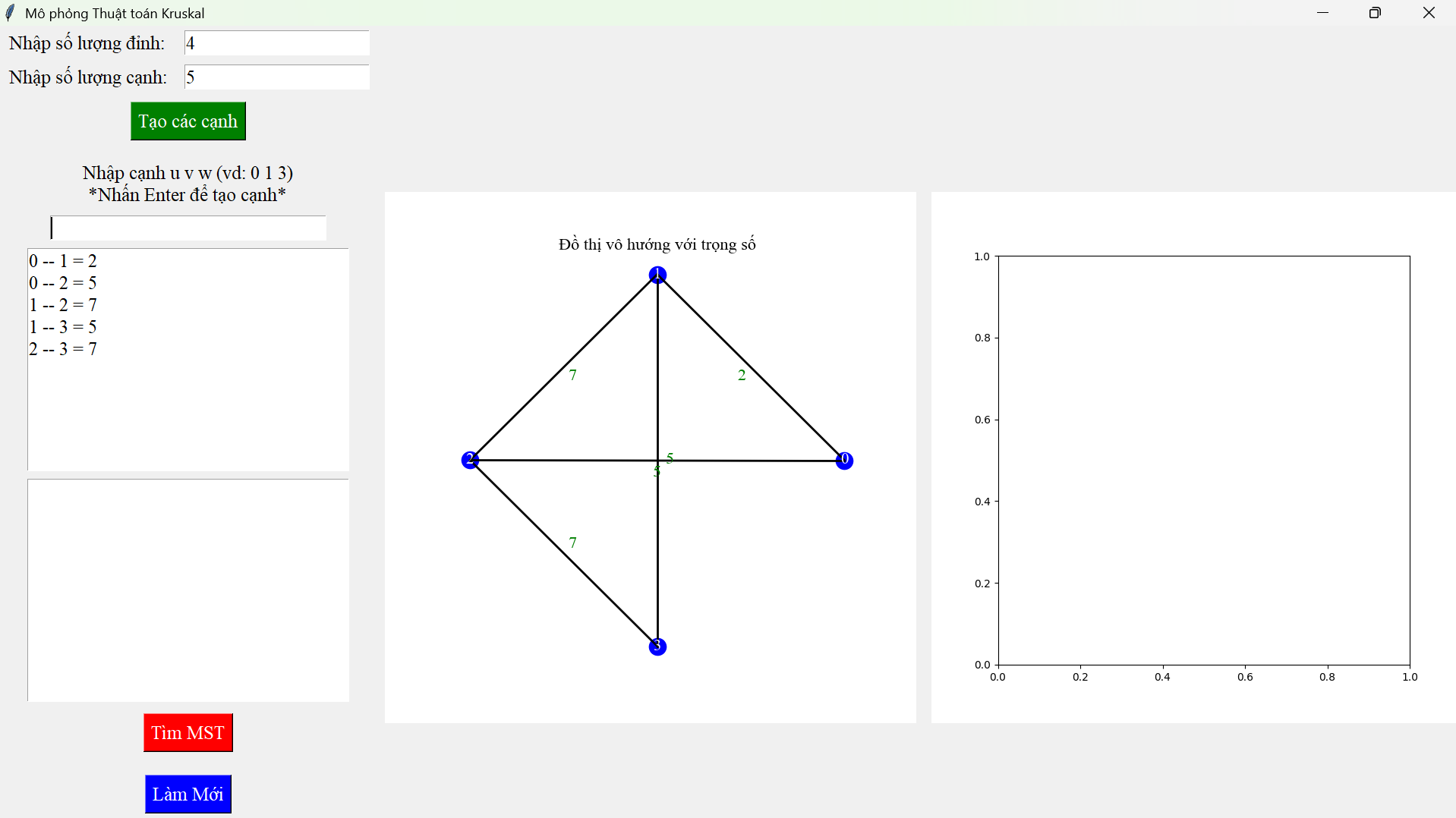
## 4.2. Nhập cạnh đồ thị

- Người dùng sẽ nhập cạnh u v và trọng số w và ô nhập cạnh và ấn phím Enter từ bàn phím để lưu cạnh vào ô chứa cạnh và ô đồ thị sẽ vẽ thêm cạnh tương ứng.



Hình 18: Giao diện sau khi nhập một cạnh

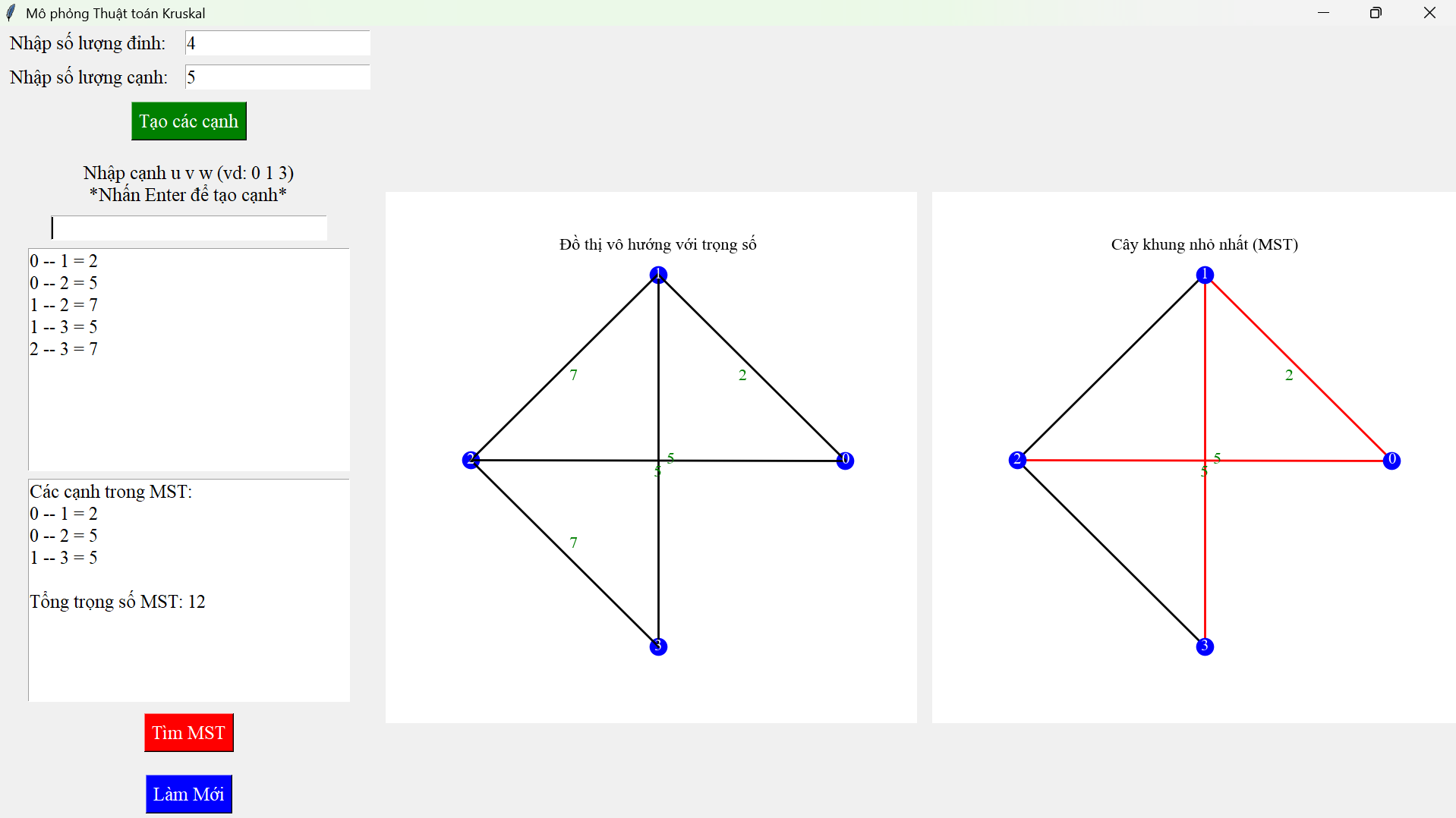
- Người dùng chỉ được nhập đúng số lượng cạnh nhập đã nhập ở trên. Ví dụ ô nhập số lượng cạnh ở trên là 4 thì người dùng chỉ có thể nhập được 4 cạnh



Hình 19: Giao diện sau khi đã nhập đủ cạnh

## 4.3. Tìm cây khung nhỏ nhất

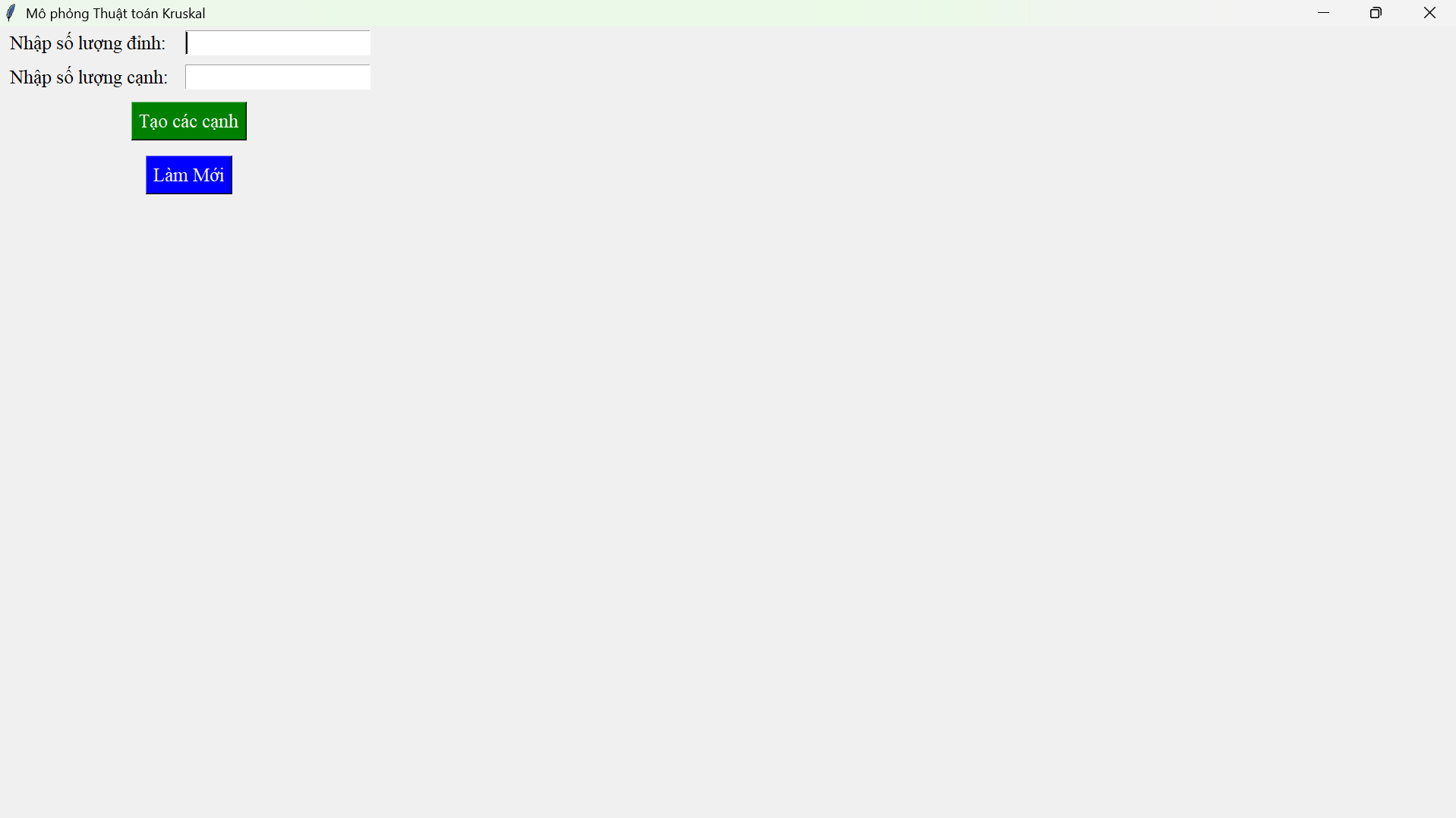
- Để tim được cây khung nhỏ nhất thì người dùng sẽ nhấn vào nút Tìm MST, thì giao diện sẽ xuất kết quả ở ô phía bên dưới ô chứa cạnh, và sẽ vẽ đồ thị cây khung nhỏ nhất ở bên phải đồ thị ban đã vẽ, đồ thị cây khung nhỏ nhất vẫn sẽ giữ cấu trúc giống với đồ thị ban đầu đã vẽ, chỉ thay đổi ở chổ sẽ tô đỏ cạnh nếu là là cây khung nhỏ nhất.



Hình 20: Giao diện sau khi chạy giải thuật Kruskal

## 4.4. Làm mới dữ liệu

- Khi người dùng nhấn vào nút “Làm mới” thì giao diện sẽ được làm mới và trở về như lúc mới chạy chương trình.



Hình 21: Giao diện sau khi làm mới.

# CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

## 5.1. Kết luận

**- Xây dựng thành công chương trình mô phỏng:** Chương trình đã được kiểm nghiệm và hoạt động đúng với các trường hợp đồ thị đầu vào.

**- Hiển thị trực quan:** Chương trình cung cấp giao diện mô phỏng giúp người dùng dễ dàng theo dõi quá trình hoạt động của giải thuật.

**- Tính ổn định và hiệu quả:** Chương trình xử lý được các đồ thị có kích thước vừa và nhỏ với thời gian xử lý hợp lý.

**- Ứng dụng thực tế:** Đồ án giúp người dùng hiểu rõ hơn về hoạt động của giải thuật Kruskal và ứng dụng của nó trong thực tế, đặc biệt là trong lĩnh vực mạng và tối ưu hóa.

## 5.2. Hướng phát triển

**- Tối ưu hóa thuật toán:** Áp dụng các kỹ thuật lập trình nâng cao để cải thiện hiệu suất của chương trình với đồ thị lớn.

**- Hỗ trợ các định dạng đầu vào đa dạng:** Bổ sung khả năng nhập đồ thị từ các tệp dữ liệu như CSV, JSON hoặc XML.

**- Tích hợp giải thuật khác:** Phát triển chương trình mô phỏng thêm các giải thuật tìm cây khung nhỏ nhất khác như Prim, Borůvka để so sánh.

**- Mở rộng ứng dụng:** Áp dụng vào các bài toán thực tế, chẳng hạn như thiết kế mạng hoặc tối ưu hóa chi phí giao thông.

**- Nâng cấp giao diện người dùng:** Phát triển giao diện người dùng trực quan hơn, hỗ trợ các công cụ tương tác như kéo, thả, và chỉnh sửa đồ thị trực tiếp.

**- Phát triển phiên bản web:** Tạo một ứng dụng web để người dùng có thể truy cập và sử dụng chương trình trực tuyến.

DANH MỤC TÀI LIỆU THAM KHẢO

1. Tìm hiểu về giải thuật Krukal:<https://www.vietjack.com/cau-truc-du-lieu-va-giai-thuat/giai-thuat-kruskal.jsp>
2. Tìm hiểu về Python: <https://aws.amazon.com/vi/what-is/python/>
3. Tìm hiểu về Tkinter:<https://www.icantech.vn/kham-pha/tkinter>
4. Tìm hiểu về Networkx:<https://www.analyticsvidhya.com/blog/2022/04/all-about-popular-graph-network-tools-in-natural-language-processing/>
5. Tìm hiểu về Matplotlib:<https://viblo.asia/p/gioi-thieu-ve-matplotlib-mot-thu-vien-rat-huu-ich-cua-python-dung-de-ve-do-thi-yMnKMN6gZ7P>
6. Youtuble
7. Tài liệu giảng dạy môn Lý thuyết đồ thị.