**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC NHA TRANG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO THỰC TẬP CƠ SỞ**

**CÀI ĐẶT THUẬT TOÁN TÌM CÂY KHUNG NHỎ NHẤT CỦA ĐỒ THỊ THEO THUẬT TOÁN KRUSKAL**

Giảng viên hướng dẫn: **NGUYỄN THUỶ ĐOAN TRANG**

Sinh viên thực tập: **NGUYỄN HỮU VINH QUANG**

Lớp: **60.CNTT-2**

MSSV: **60130835**

**Năm học: 2020**

**NHẬN XÉT KẾT QUẢ THỰC TẬP**

Họ và tên sinh viên: .

Mã số sinh viên: …………………… Lớp ……………… Hệ:

Khoa Công nghệ Thông tin, Trường Đại học Nha Trang.

Nội dung thực tập:

Đánh giá kết quả thực tập:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **STT** | **Khả năng/kỹ năng** | **Tốt** | **Khá** | **TB** | **Yếu** | **Kém** |
| 1 | Kiến thức 1 (lý thuyết, công nghệ) |  |  |  |  |  |
| 2 | Kiến thức 2 (thực hành) |  |  |  |  |  |
| 3 | Làm việc nhóm |  |  |  |  |  |
| 4 | Tổ chức công việc và làm việc độc lập |  |  |  |  |  |
| 5 | Giao tiếp và thái độ làm việc |  |  |  |  |  |

Nhận xét chung:

*………………… ngày …. tháng … năm ……*

Giáo viên hướng dẫn

# **LỜI MỞ ĐẦU**

Trên thực tế có nhiều bài toán liên quan tới một tập các đối tượng và những mối liên hệ giữa chúng, đòi hỏi toán học phải đặt ra một mô hình biểu diễn một cách chặt chẽ và tổng quát bằng ngôn ngữ ký hiệu, đó là đồ thị. Những ý tưởng cơ bản của nó được đưa ra từ thế kỷ thứ XVIII bởi nhà toán học Thuỵ Sĩ Leonhard Euler, ông đã dùng mô hình đồ thị để giải bài toán về những cây cầu Konigsberg nổi tiếng.

Mặc dù Lý thuyết đồ thị đã được khoa học phát triển từ rất lâu nhưng lại có nhiều ứng dụng hiện đại. Đặc biệt trong khoảng vài mươi năm trở lại đây, cùng với sự ra đời của máy tính điện tử và sự phát triển nhanh chóng của Tin học, Lý thuyết đồ thị càng được quan tâm đến nhiều hơn. Đặc biệt là các thuật toán trên đồ thị đã có nhiều ứng dụng trong nhiều lĩnh vực khác nhau như: Mạng máy tính, Lý thuyết mã, Tối ưu hoá, Kinh tế học v.v... Chẳng hạn như trả lời câu hỏi: Hai máy tính trong mạng có thể liên hệ được với nhau hay không ?; hay vấn đề phân biệt hai hợp chất hoá học có cùng công thức phân tử nhưng lại khác nhau về công thức cấu tạo cũng được giải quyết nhờ mô hình đồ thị. Hiện nay, môn học này là một trong những kiến thức cơ sở của bộ môn khoa học máy tính.

Việc biểu diễn đồ thị và xử lý nó là rất cần thiết, các thao tác như: tìm cây khung nhỏ nhất, tìm đường đi ngắn nhất,… trong đó việc tìm kiếm cây khung nhỏ nhất là việc cần thiết để tìm tổng chi phí cả đồ thị để có thể tính toán hợp lí hơn. Áp dụng nhiều trong việc xây dựng và việc tính toán,…

Sau thời gian tìm hiểu và xây dựng, chương trình đã giải quyết được việc mất thời gian trong việc tìm kiếm cây khung nhỏ nhất bằng thuật toán Kruskal. Nếu đồ thị nhỏ thì việc tìm sẽ đơn giản nhưng đối với những đồ thị lớn và rất nhiều cạnh hay nhưng đồ thị đầy đủ - các đỉnh đều nối với nhau, thì việc tìm bằng tay sẽ rất khó và tốn thời gian đó là lý do tại sao chương trình này được xây dựng nên.

# **LÝ DO CHỌN ĐỀ TÀI**

Nhằm giải quyết việc tốn thời gian trong việc tìm kiếm cây khung nhỏ nhất. Giải quyết các bài toán tìm kiếm cây khung nhỏ nhất với đồ thị nhiều đỉnh, nhiều cạnh hay hay các đồ thị có các trọng số lớn, các đồ thị có nhiều cạnh và trọng số lớn,…

**MỤC LỤC**

[**LỜI MỞ ĐẦU** 2](#_Toc62297779)

[**LÝ DO CHỌN ĐỀ TÀI** 2](#_Toc62297780)

[**1.** **CHƯƠNG 1: CƠ SỞ LÝ THUYẾT** 5](#_Toc62297781)

[**1.1** **Các khái niệm đồ thị** 5](#_Toc62297782)

[**1.2** **Định nghĩa đồ thị** 5](#_Toc62297783)

[**1.3** **Các định nghĩa** 5](#_Toc62297784)

[*1.3.1* *Đồ thị vô hướng* 5](#_Toc62297785)

[*1.3.2* *Đồ thị có hướng* 6](#_Toc62297786)

[*1.3.3* *Đơn đồ thị và Đa đồ thị* 6](#_Toc62297787)

[*1.3.4* *Đồ thị hỗn hợp* 6](#_Toc62297788)

[**1.4** **Thuật toán Kruskal** 6](#_Toc62297789)

[*1.4.1* *Khái niệm liên quan* 6](#_Toc62297790)

[*1.4.2* *Bài toán dẫn nhập* 7](#_Toc62297791)

[*1.4.3* *Tư tưởng thuật toán* 7](#_Toc62297792)

[*1.4.4* *Mô tả thuật toán* 7](#_Toc62297793)

[*1.4.5* *Mã giả* 8](#_Toc62297794)

[*1.4.6* *Kỹ thuật đánh nhãn đỉnh* 8](#_Toc62297795)

[*1.4.7* *Ghi chú* 8](#_Toc62297796)

[**1.5** **Ví dụ thuật toán Kruskal** 9](#_Toc62297797)

[**2.** **CHƯƠNG 2: XÂY DỰNG CHƯƠN TRÌNH** 15](#_Toc62297798)

[**2.1** **Phần thuật toán** 15](#_Toc62297799)

[*2.1.1* *Khai báo* 15](#_Toc62297800)

[*2.1.2* *Đầu vào* 15](#_Toc62297801)

[*2.1.3* *Đầu ra* 17](#_Toc62297802)

[*2.1.4* *Chương trình xử lý* 17](#_Toc62297803)

[*2.1.5* *Hàm main* 20](#_Toc62297804)

[**2.2** **Độ phức tạp thuật toán** 20](#_Toc62297805)

[**3.** **CHƯƠNG 3: XÂY DỰNG ĐỒ HỌA** 21](#_Toc62297806)

[**3.1** **Lời dẫn** 21](#_Toc62297807)

[**3.2** **Phần code đồ họa** 21](#_Toc62297808)

[*3.2.1* *Bảng biểu diễn đồ thị* 22](#_Toc62297809)

[*3.2.2* *Khung biểu diễn đầu vào và đầu ra* 23](#_Toc62297810)

[*3.2.3* *Xây dựng các thực thể* 24](#_Toc62297811)

[*3.2.4* *Phần xử lý đồ họa* 26](#_Toc62297812)

[*3.2.5* *Chương trình chính* 28](#_Toc62297813)

[**KẾT LUẬN** 31](#_Toc62297814)

[**TÀI LIỆU THAM KHẢO** 32](#_Toc62297815)

**DANH MỤC CÁC BẢNG, SƠ ĐỒ, HÌNH**

Bảng 1. Liệt kê các cạnh và trọng số

Bảng 2. Danh sách các cạnh và trọng số đã sắp xếp theo trọng số

Hình 1. Đồ thị

Hình 2. Kết quả cây khung

Hình 3. Demo giao diện chương trình

Hình 4. Demo giao diện

Hình 5. Khung chính

Hình 6. Khung phụ

Hình 7. Giao diện bắt đầu

Hình 8. Giao diện kết thúc

# **CHƯƠNG 1: CƠ SỞ LÝ THUYẾT**

## **Các khái niệm đồ thị**

Trong toán học và tin học, lý thuyết đồ thị (tiếng Anh: graph theory) nghiên cứu các tính chất của đồ thị. Một cách không chính thức, đồ thị là một tập các đối tượng được gọi là các đỉnh (hoặc nút) nối với nhau bởi các cạnh (hoặc cung). Cạnh có thể có hướng hoặc vô hướng. Đồ thị thường được vẽ dưới dạng một tập các điểm (các đỉnh nối với nhau bằng các đoạn thẳng (các cạnh).

Đồ thị biểu diễn được rất nhiều cấu trúc, nhiều bài toán thực tế có thể được biểu diễn bằng đồ thị. Ví dụ, cấu trúc liên kết của một website có thể được biểu diễn bằng một đồ thị có hướng như sau: các đỉnh là các trang web hiện có tại website, tồn tại một cạnh có hướng nối từ trang A tới trang B khi và chỉ khi A có chứa 1 liên kết tới B. Do vậy, sự phát triển của các thuật toán xử lý đồ thị là một trong các mối quan tâm chính của khoa học máy tính.

Cấu trúc đồ thị có thể được mở rộng bằng cách gán trọng số cho mỗi cạnh. Có thể sử dụng đồ thị có trọng số để biểu diễn nhiều khái niệm khác nhau. Ví dụ, nếu đồ thị biểu diễn một mạng đường giao thông, các trọng số có thể là độ dài của mỗi con đường. Một cách khác để mở rộng đồ thị cơ bản là quy định hướng cho các cạnh của đồ thị (như đối với các trang web, A liên kết tới B, nhưng B không nhất thiết cũng liên kết tới A). Loại đồ thị này được gọi là đồ thị có hướng. Một đồ thị có hướng với các cạnh có trọng số được gọi là một lưới.

Các lưới có nhiều ứng dụng trong khía cạnh thực tiễn của lý thuyết đồ thị, chẳng hạn, phân tích lưới có thể dùng để mô hình hoá và phân tích mạng lưới giao thông hoặc nhằm "phát hiện" hình dáng của Internet.

## **Định nghĩa đồ thị**

Trong toán học và tin học, đồ thị là đối tượng nghiên cứu cơ bản của lý thuyết đồ thị. Một cách không chính thức, đồ thị là một tập các đối tượng gọi là đỉnh nối với nhau bởi các cạnh. Thông thường, đồ thị được vẽ dưới dạng một tập các điểm (đỉnh, nút) nối với nhau bởi các đoạn thẳng (cạnh). Tùy theo ứng dụng mà một số cạnh có thể có hướng.

## **Các định nghĩa**

### *Đồ thị vô hướng*

Đồ thị vô hướng hoặc đồ thị G là một cặp không có thứ tự (unordered pair) G:=(V, E), trong đó V, tập các đỉnh hoặc nút,

E, 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.

Trong nhiều tài liệu, tập các cạnh bao gồm cả các cặp đỉnh không phân biệt, các cạnh này được gọi là các khuyên. V (và E) thường là các tập hữu hạn, phần lớn các kết quả nghiên cứu đã biết không đúng (hoặc khác) khi áp dụng cho đồ thị vô hạn (infinite graph) vì nhiều luận cứ không dùng được trong trường hợp vô hạn.

### *Đồ thị có hướng*

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

- V, tập các đỉnh hoặc nút,

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

### *Đơn đồ thị và Đa đồ thị*

* Đơn đồ thị là đồ thị mà không có khuyên và không có cạnh song song.
* Đa đồ thị là đồ thị mà không thỏa đơn đồ thị.
* Đa đồ thị có hướng là một đồ thị có hướng, trong đó, nếu x và y là hai đỉnh thì đồ thị được phép có cả hai cung (x, y) và (y, x).
* Đơn đồ thị có hướng là một đồ thị có hướng, trong đó, nếu x và y là hai đỉnh thì đồ thị chỉ được phép có tối đa một trong hai cung (x, y) hoặc (y, x).

### *Đồ thị hỗn hợp*

Đồ thị hỗn hợp G là một bộ ba có thứ tự G:= (V,E,A) với V, E và A được định nghĩa như trên.

## **Thuật toán Kruskal**

### *Khái niệm liên quan*

Khái niệm cây khung nhỏ nhất: Với một đồ thị liên thông, vô hướng cho trước, cây bao trùm của nó là một đồ thị con có dạng cây và có tất cả các đỉnh liên thông với nhau. Một đồ thị có thể có nhiều cây bao phủ khác nhau. Chúng ta cũng có thể gán một trọng số cho mỗi cạnh, là con số biểu thị sự "không ưa thích" và dùng nó để tính toán trọng số của một cây bao trùm bằng cách cộng tất cả trọng số của cạnh trong cây bao trùm đó. Khi đó, một cây bao trùm nhỏ nhất là một cây bao trùm có trọng số bé hơn bằng trọng số của tất cả các cây bao trùm khác.

*Tính chất:*

* Có thể có vô số: Có thể có một vài cây bao trùm nhỏ nhất có cùng trọng số và có số cạnh nhỏ nhất; cụ thể hơn, nếu tất cả các cạnh của một đồ thị đều có trọng số bằng nhau, thì tất cả các cây bao trùm của đồ thị đó đều là nhỏ nhất.
* Tính duy nhất: Nếu mỗi cạnh có trọng số riêng biệt thì sẽ chỉ có một, và chỉ một cây bao trùm nhỏ nhất. Có thể chứng minh phát biểu này bằng quy nạp hoặc phản chứng. Điều này đúng trong nhiều trường hợp thực tế, như ví dụ về công ty truyền hình cáp ở trên chẳng hạn, khi đó rất hiếm khi hai con đường lại có chính xác cùng một chi phí. Phát biểu này cũng được tổng quát hóa cho rừng bao trùm.
* Đồ thị có chi phí nhỏ nhất: Nếu trọng số là số dương, thì một cây bao trùm nhỏ nhất cũng chính là đồ thị con có chi phí nhỏ nhất kết nối tất cả đỉnh, vì các đồ thị con có chứa chu trình bao giờ cũng có tổng trọng số lớn hơn.
* Tính chất vòng: Với một chu trình C bất kỳ trong đồ thị, nếu trọng số của cạnh e nào đó của C lớn hơn trọng số của các cạnh còn lại của C, thì cạnh đó không thể thuộc về cây bao trùm nhỏ nhất.
* Tính chất cắt: Với nhát cắt C bất kỳ trong đồ thị, nếu trọng số của một cạnh e của C nhỏ hơn trọng số của các cạnh còn lại của C, thì cạnh này thuộc về tất cả các cây bao trùm nhỏ nhất của đồ thị.
* Cạnh có chi phí nhỏ nhất: Nếu một cạnh của đồ thị với chi phí nhỏ nhất e là duy nhất, thì cạnh này sẽ thuộc về bất kỳ một cây bao trùm nhỏ nhất nào

### *Bài toán dẫn nhập*

Cho một đồ thị có trọng số với n đỉnh. Yêu cầu tìm ra cây khung nhỏ nhất.

### *Tư tưởng thuật toán*

Thuật toán Kruskal dựa trên mô hình xây dựng cây khung nhỏ nhất bằng thuật toán hợp nhất.

* Thuật toán không xét các cạnh với thứ tự tuỳ ý.
* Thuật toán xét các cạnh theo thứ tự đã sắp xếp theo trọng số.
* Để xây dựng tập n-1 cạnh của cây khung nhỏ nhất - tạm gọi là tập K, Kruskal đề nghị cách kết nạp lần lượt các cạnh vào tập đó theo nguyên tắc như sau:
* Ưu tiên các cạnh có trọng số nhỏ hơn.
* Kết nạp cạnh khi nó không tạo chu trình với tập cạnh đã kết nạp trước đó.

Đó là một nguyên tắc chính xác và đúng đắn, đảm bảo tập K nếu thu đủ n - 1 cạnh sẽ là cây khung nhỏ nhất.

### *Mô tả thuật toán*

Giả sử ta cần tìm cây bao trùm nhỏ nhất của đồ thị G. Thuật toán bao gồm các bước sau.

* Khởi tạo rừng F (tập hợp các cây), trong đó mỗi đỉnh của G tạo thành một cây riêng biệt
* Khởi tạo tập S chứa tất cả các cạnh của G
* Chừng nào S còn khác rỗng và F gồm hơn một cây
* Xóa cạnh nhỏ nhất trong S
* Nếu cạnh đó nối hai cây khác nhau trong F, thì thêm nó vào F và hợp hai cây kề với nó làm một
* Nếu không thì loại bỏ cạnh đó.

Khi thuật toán kết thúc, rừng chỉ gồm đúng một cây và đó là một cây bao trùm nhỏ nhất của đồ thị G.

### *Mã giả*

Cho đồ thị **G = (X, E).**

Bước 1: Sắp xếp các cạnh của đồ thị theo thứ tự trọng số tăng dần.

Bước 2: Khởi tạo T:= Ø

Bước 3: Lần lượt lấy từng cạnh thuộc danh sách đã sắp xếp. Nếu T+{e} không chứa chu trình thì gán T:=T+{e}.

Bước 4: Nếu T đủ n-1 phần tử thì dừng, ngược lại làm tiếp bước 3.

### *Kỹ thuật đánh nhãn đỉnh*

Kỹ thuật đánh nhãn đỉnh Trong thuật toán Kruskal, để kiểm tra xem T + {e} có chứa chu trình hay không ta có thể dùng kỹ thuật gắn nhãn đỉnh, kỹ thuật này khá đơn giản và hiệu quả.

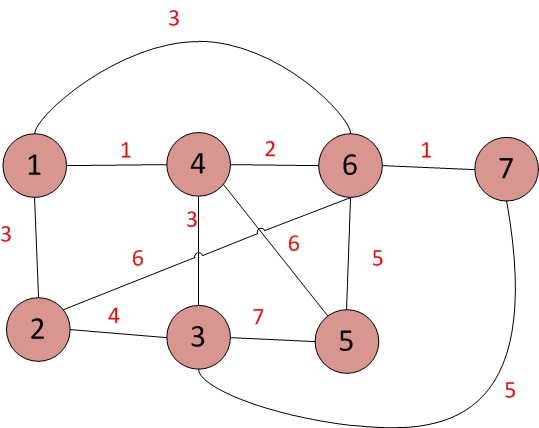
* Ngay sau bước 1 của thuật toán, ta gắn đỉnh i của đồ thị một nhãn là i
* Trong bước 2:
* Nếu hai đầu cạnh e có cùng nhãn (tức là nhãn của e.v1 và nhãn của e.v2 bằng nhau) thì T+{e} tạo chu trình, ta không đưa e vào T.
* Ngược lại [nếu Label(e.v1)!= Label(e.v2) ] thì ta đưa e vào T và thực hiện công việc ghép nhãn bằng cách:
* lab1 = Min(Label(e.v1), Label (e.v2))
* lab2 = Max(Label(e.v1), Label (e.v2))
* Sửa nhãn của tất cả các đỉnh nào có nhãn là lab2 thành nhãn lab1

### *Ghi chú*

* Trong quá trình xây dựng T thì các cạnh có thể không liên thông nhau lúc đó T chỉ là rừng chứ chưa trở thành cây.
* Khi thuật toán dừng:
* Nếu T chưa đủ n - 1 cạnh thì đồ thị G không liên thông(không có cây khung)

Ngược lại thì T là cây khung cần tìm

## **Ví dụ thuật toán Kruskal**



*Hình 1. Đồ thị*

**Bước 1: Liệt kê tất cả cạnh với trọng số của cạnh đó**: Dựa vào đồ thị ta liệt kê ra các cạnh gồm đỉnh đầu, đỉnh cuối và trọng số:

|  |  |  |
| --- | --- | --- |
| **Điểm đầu** | **Điểm cuối** | **Trọng số** |
| 1 | 2 | 3 |
| 1 | 4 | 1 |
| 1 | 6 | 3 |
| 2 | 3 | 4 |
| 2 | 6 | 6 |
| 3 | 4 | 3 |
| 3 | 5 | 7 |
| 3 | 7 | 5 |
| 4 | 5 | 6 |
| 4 | 6 | 2 |
| 5 | 6 | 5 |
| 6 | 7 | 1 |

Bảng 1. Liệt kê các cạnh và trọng số

**Bước 2: Sắp xếp các cạnh theo trọng số tăng dần:**

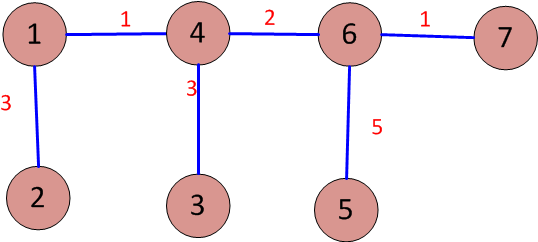
|  |  |  |
| --- | --- | --- |
| **Điểm đầu** | **Điểm cuối** | **Trọng số** |
| 1 | 4 | 1 |
| 6 | 7 | 1 |
| 4 | 6 | 2 |
| 1 | 2 | 3 |
| 1 | 6 | 3 |
| 3 | 4 | 3 |
| 2 | 3 | 4 |
| 3 | 7 | 5 |
| 5 | 6 | 5 |
| 2 | 6 | 6 |
| 4 | 5 | 6 |
| 3 | 5 | 7 |

Bảng 2. Danh sách các cạnh và trọng số đã sắp xếp theo trọng số

**Bước 3: Dựa vào kết quả ở bước 2. Ta tiến hành tìm cây khung bằng thuật toán Kruskal**

|  |  |
| --- | --- |
| **Kết quả** | **Cạnh đang xét** |
| [https://upload.wikimedia.org/wikipedia/commons/thumb/4/41/Do_thi_G_tung_buoc_%281%29.png/200px-Do_thi_G_tung_buoc_%281%29.png](https://vi.wikipedia.org/wiki/T%E1%BA%ADp_tin:Do_thi_G_tung_buoc_(1).png)  Đồ thị G | 1-4-1: Ta nhận thấy cạnh 1-4 không tạo ra một chu trình nào. Vì vậy, thêm 1-4 vào tập hợp |
| [https://upload.wikimedia.org/wikipedia/commons/thumb/b/b0/Do_thi_G_tung_buoc_%282%29.png/200px-Do_thi_G_tung_buoc_%282%29.png](https://vi.wikipedia.org/wiki/T%E1%BA%ADp_tin:Do_thi_G_tung_buoc_(2).png)  Đồ thị G | 6-7-1: Ta nhận thấy cạnh 6-7 không tạo ra một chu trình nào. Vì vậy, thêm 6-7 vào tập hợp |
| [https://upload.wikimedia.org/wikipedia/commons/thumb/6/6f/Do_thi_G_tung_buoc_%283%29.png/200px-Do_thi_G_tung_buoc_%283%29.png](https://vi.wikipedia.org/wiki/T%E1%BA%ADp_tin:Do_thi_G_tung_buoc_(3).png)  Đồ thị G | 4-6-2: Ta nhận thấy cạnh 4-6 không tạo ra một chu trình nào. Vì vậy, thêm 4-6 vào tập hợp |
| [https://upload.wikimedia.org/wikipedia/commons/thumb/c/c6/Do_thi_G_tung_buoc_%284%29.png/200px-Do_thi_G_tung_buoc_%284%29.png](https://vi.wikipedia.org/wiki/T%E1%BA%ADp_tin:Do_thi_G_tung_buoc_(4).png)  Đồ thị G | 1-2-3: Ta nhận thấy cạnh 1-2 không tạo ra một chu trình nào. Vì vậy, thêm 1-2 vào tập hợp |
| [https://upload.wikimedia.org/wikipedia/commons/thumb/b/b7/Do_thi_G_tung_buoc_%285%29.png/200px-Do_thi_G_tung_buoc_%285%29.png](https://vi.wikipedia.org/wiki/T%E1%BA%ADp_tin:Do_thi_G_tung_buoc_(5).png)  Đồ thị G | 1-6-3: Ta nhận thấy cạnh 1-6 tạo ra một chu trình. Không thêm vào tập hợp.  3-4-3: Ta nhận thấy cạnh 3-4 không tạo ra một chu trình. Vì vậy, thêm 3-4 vào tập hợp |
| [https://upload.wikimedia.org/wikipedia/commons/thumb/0/04/Do_thi_G_tung_buoc_%286%29.png/200px-Do_thi_G_tung_buoc_%286%29.png](https://vi.wikipedia.org/wiki/T%E1%BA%ADp_tin:Do_thi_G_tung_buoc_(6).png)  Đồ thị G | 2-3-4: Ta nhận thấy cạnh 2-3 tạo ra một chu trình. Không thêm vào tập hợp. |
| [https://upload.wikimedia.org/wikipedia/commons/thumb/3/31/Do_thi_G_tung_buoc_%287%29.png/200px-Do_thi_G_tung_buoc_%287%29.png](https://vi.wikipedia.org/wiki/T%E1%BA%ADp_tin:Do_thi_G_tung_buoc_(7).png)  Đồ thị G | 3-7-5: Ta nhận thấy cạnh 3-7 tạo ra một chu trình. Không thêm vào tập hợp. |
| [https://upload.wikimedia.org/wikipedia/commons/thumb/5/5b/Do_thi_G_tung_buoc_%288%29.png/200px-Do_thi_G_tung_buoc_%288%29.png](https://vi.wikipedia.org/wiki/T%E1%BA%ADp_tin:Do_thi_G_tung_buoc_(8).png)  Đồ thị G | 5-6-5: Ta nhận thấy cạnh 5-6 không tạo ra một chu trình nào. Vì vậy, thêm 5-6 vào tập hợp |

* Đến đây, ta đã tìm được 6 cạnh. Vậy kết thúc thuật toán. (Thỏa (\*))
* Kết quả: Ta được đồ thị sau



*Hình 2. Kết quả cây khung*

**Với tổng chi phí là: Ta cộng tất cả các trọng số giữa các đỉnh lại với nhau**

* Vậy tổng chi phí: 3 + 1 + 3 + 2 + 5 + 1 = 15

# **CHƯƠNG 2: XÂY DỰNG CHƯƠN TRÌNH**

## **Phần thuật toán**

*\*độ thị vô hướng có trọng số*

***(MỨC 1)***

### *Khai báo*

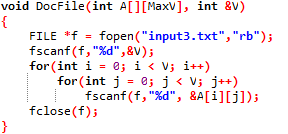
Khai báo các biến cần thiết để thực hiện chương trình.



### *Đầu vào*

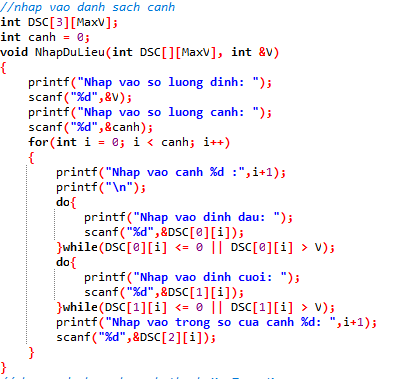
Chương trình được chia thành nhiều các chương trình nhỏ hơn để dễ quản lí và sửa chữa.

Phần lấy dữ liệu thì được chia làm 2 chương trình con: Lấy dữ liệu từ file .txt có cấu trúc file được quy định sẳn, và chương trình con nhập từ bản phím. Nhưng trong chương trình này sử dụng ma trận kề để lưu trữ đồ thị trọng số, nên sử dụng lấy dữ liệu từ file để tiện hơn.



Nhưng để đọc file thì cần thêm thư viện *stdlib.h* để chương trình có thể thực hiện.

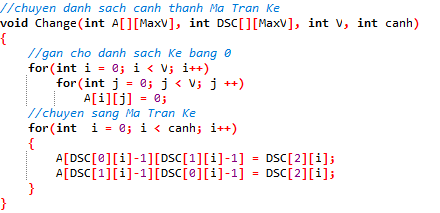
Mã lệnh nhập từ bàn phím dữ liệu (việc nhập vào ma trận kề như yêu cầu của đề thường sẽ phức tạp hơn mà rất dễ bị nhầm lẫn nên chương trình yêu cầu nhập vào danh sánh cạnh sau đó tự chuyển đổi thành ma trận kề để thực hiện chương trình).



Trước hết, khai báo mảng 3 x MaxV (MaxV là số đỉnh tối đa đã được định nghĩa ở đầu chương trình).

Xây dựng 1 chương trình để nhập, phần tử đầu là đỉnh đầu, phần tử thứ 2 là đỉnh cuối, còn phần tử cuối là trọng số của cạnh \*đầu và \*cuối.

Việc nhập vào được quy định để đảm bảo việc nhập không vượt quá giới hạn khai báo - ở đây là sô đỉnh. Ví dụ như khai báo 5 đỉnh thì không được nhập số lớn hơn 5 và nhỏ hơn 1 – vì các đỉnh được quy định theo thứ tự số tự nhiên từ 1 đến V.



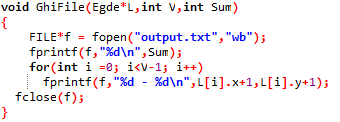
Khi đã quy định đúng số cho các đỉnh thì việc chuyển sang ma trận kề sẽ đơn giản hơn.

Trong hàm này đã sử dụng biến *canh* để chạy vòng lặp. Giải thích ở đây *canh* để nhập vào số lượng cạnh vào ma trận kề. Trước hết các công việc của chương trình này thì cần khai báo ma trận kề có V đỉnh tất cả bằng 0 – giá trị mặc định của ma trận kề.

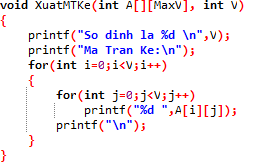
Vì chương trình đang hướng đến là ma trận kề của đồ thị trọng số vô hướng. Tính chất của ma trận kề của đồ thị vô hướng là đối xứng qua đường chéo chính. Nên chúng ta cần phải nhập [đỉnh đầu][đỉnh cuối] và [đỉnh cuối][đỉnh đầu] và cả 2 biến đều có cùng giá trị trọng số.

### *Đầu ra*

Đã lấy dữ liệu từ file thì cần xuất kết quả ra file. Nên chương tình được xây dựng thêm hàm xuất kết quả ra một file riêng để tiện cho việc lưu giữ kết quả của thuật toán sau khi thực hiện.

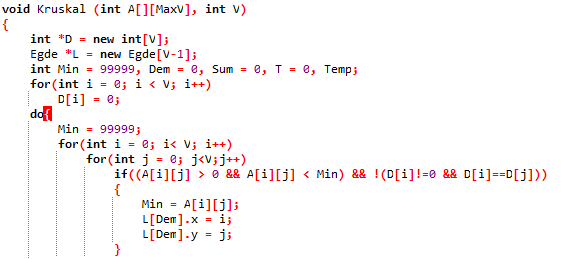


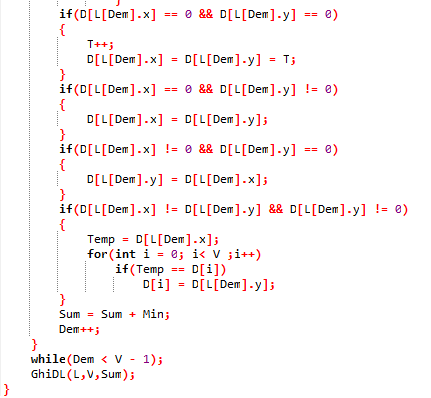
Cùng với việc xuất file thì cũng có chương trình xuất ra màn hình để xem được phần INPUT của chương trình. Việc xuất ra màn hình kết quả tiện cho việc xem kết quả trực tiếp ngay sau khi chương trình chạy xong.



### *Chương trình xử lý*

Trong các chương trình con thì chương trình con Krusal là chương trình quan trọng nhất vì nó là chương trình thực hiện chính cho thuật toán này – có thể ví chương trình này như bộ não của chương trình. “Nó” quyết định tính năng của toàn bộ chương trình. Mình có thể có 1 giao diện đẹp, một đầu vào hay ho nhưng nếu không có chương trình xử lý thì bài toán cũng chẳng khác gì một thùng rỗng.





Trong phần chường trình con này nó thể hiện rõ như trong thuật toán Kruskal thông qua chuỗi câu lệnh. Hệ thống câu lệnh thể hiện tương tư các cách xử lý của thuật toán



Phần đầu là khai báo các biến cần thiết của chương trình. Các biến được sử dụng xuyên suốt trong chương trình xử lý.

Trong đó \*D là biến con trỏ được khai báo để gán giá trị cha cho các đỉnh đồ thị

Ví dụ: như đỉnh 1 thì có cha là 1 và đỉnh 2 có cha là 2.

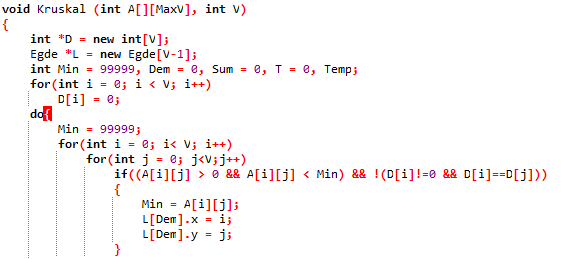
Các biến này lưu giữ giá trị cha của đỉnh nhằm cho việc các đỉnh có tạo thành chu trình hay không khi chạy thuật toán tìm cây khung nhỏ nhất Kruskal.

Biến *Egde \*L* là kiểu cấu trúc do người dùng định nghĩa gồm 2 biến x và y thể hiện cho 1 cạnh có đỉnh đầu và đỉnh cuối và được khai báo V – 1 (V là số đỉnh) tức là nếu có V cạnh thì sẽ có V - 1 cạnh của cây khung.

Biến *Sum* là biến lưu trữ giá trị của tổng chi phí của cây khung.

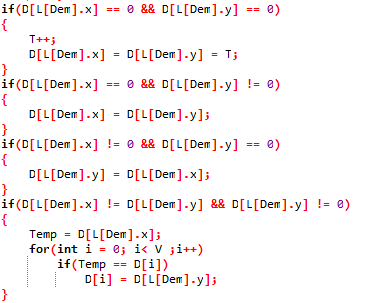
Biến *T* và biến *Temp* là 2 biến bổ trợ cho việc xét các cha của các đỉnh.

Trong đó có biến *Min = 99999;* là biến dùng để so sánh với các giá trị trọng số của đồ thị.



Đoạn code này nhằm mục đính là tìm cạnh có trọng số nhỏ nhất.

Giải tích về câu điều kiện trong **if**: việc đầu tiên là tìm giá trị trọng số nhỏ nhât của đồ thị (A[i][j] phải là số khác 0 – 0 thể hiện là không nối giữa 2 đỉnh, và A[i][j] phải nhỏ hơn biến Min, cùng với đó là 2 đỉnh đó chưa cùng cha. Tức là mỗi đỉnh sẽ có 1 cha và sau khi cạnh đó được chọn là cạnh của cây khung thì nó sẽ được cho cùng cha.



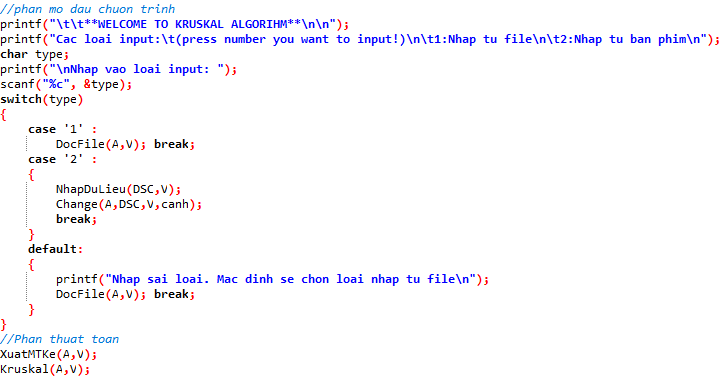
Đoạn mã này dùng để gán giá trị cha cho các đỉnh sau khi đã tìm được cạnh của cây khung.

Sau đó ta gán giá trị tổng chi phí cho Sum và tăng giá trị Dem lên 1 đơn vị - biến Dem để đếm số lượng cạnh của cây khung.

Vì chưa biết được số lượng vòng lặp nên chương trình phải đặt trong vòng lặp do {…} while(); hoặc trong vòng lặp while (){…}; nhưng trong chương trình này sử dụng do {…} while(); sẽ cho độ tối ưu tốt hơn.

### *Hàm main*

Trong chương trình chính chỉ cần gọi là các chương trình con đã nêu ở trên để chương trình cho thể thực hiện. Nhưng chương trình hiện có 2 loại input, nên trong hàm *main* sử dụng câu lệnh *switch… case …*  để thuận tiện cho việc lựa chọn loại input mong muốn.



## **Độ phức tạp thuật toán**

Nếu E là số cạnh và V là số đỉnh của đồ thị thì thuật toán Kruskal chạy trong thời gian O(E log V).

Có thể đạt được thời gian này bằng phương pháp sau: sắp xếp tất cả các cạnh theo trọng số trong thời gian O(E log E). Điều này cho phép thực hiện bước "xóa cạnh nhỏ nhất trong S" trong thời gian hằng số. Sau đó sử dụng cấu trúc dữ liệu cho các tập hợp không giao nhau để lưu trữ thông tin đỉnh nào nằm ở cây nào trong F. Ta cần thực hiện O(E) thao tác, hai thao tác 'tìm' và không quá một thao tác 'hợp' cho mỗi cạnh. Ngay cả những thuật toán đơn giản cho bài toán này, chẳng hạn hợp bằng trọng số cũng có thể thực hiện O(E) thao tác trong thời gian O(E log V).

**Vì vậy tổng thời gian là O(E log E) = O(E log V).**

# **CHƯƠNG 3: XÂY DỰNG ĐỒ HỌA**

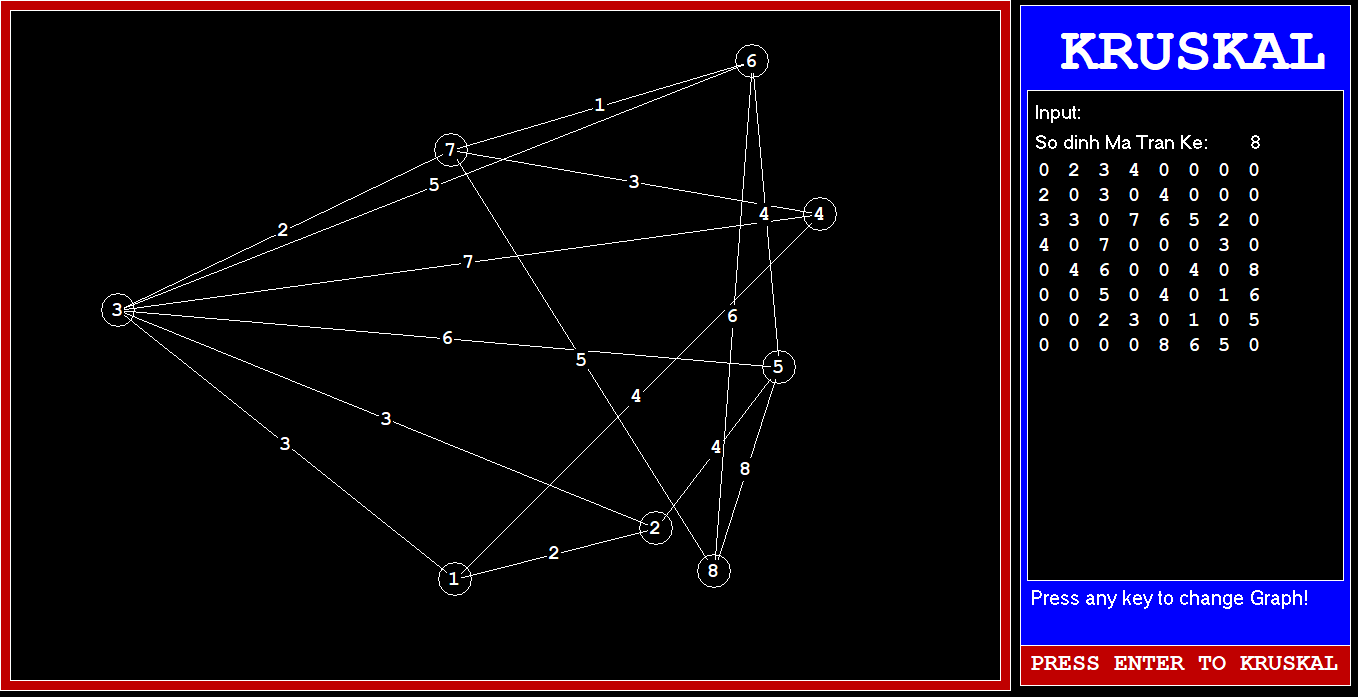
***(MỨC 2)***

## **Lời dẫn**

Thuật toán chạy tốt, kết quả hoàn hảo nhưng để hoản hoàn hảo của hoàn hảo hơn thì cần có giao diện để thể cách thức chạy của chương trình. Và việc giao tiếp với người dùng sẽ dễ dàng hơn. Mục đích của chương trình là giúp người dùng dễ dàng trong việc tìm kiếm cây khung nhỏ nhất nến chắc chắn việc thể hiện giao diện cách tìm thì chương trình sẽ đạt hiệu quả tối đa.

## **Phần code đồ họa**

*Demo giao diện*



Hình 3. Demo giao diện chương trình

Trong phần đồ họa thì có 2 phần chính

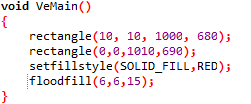
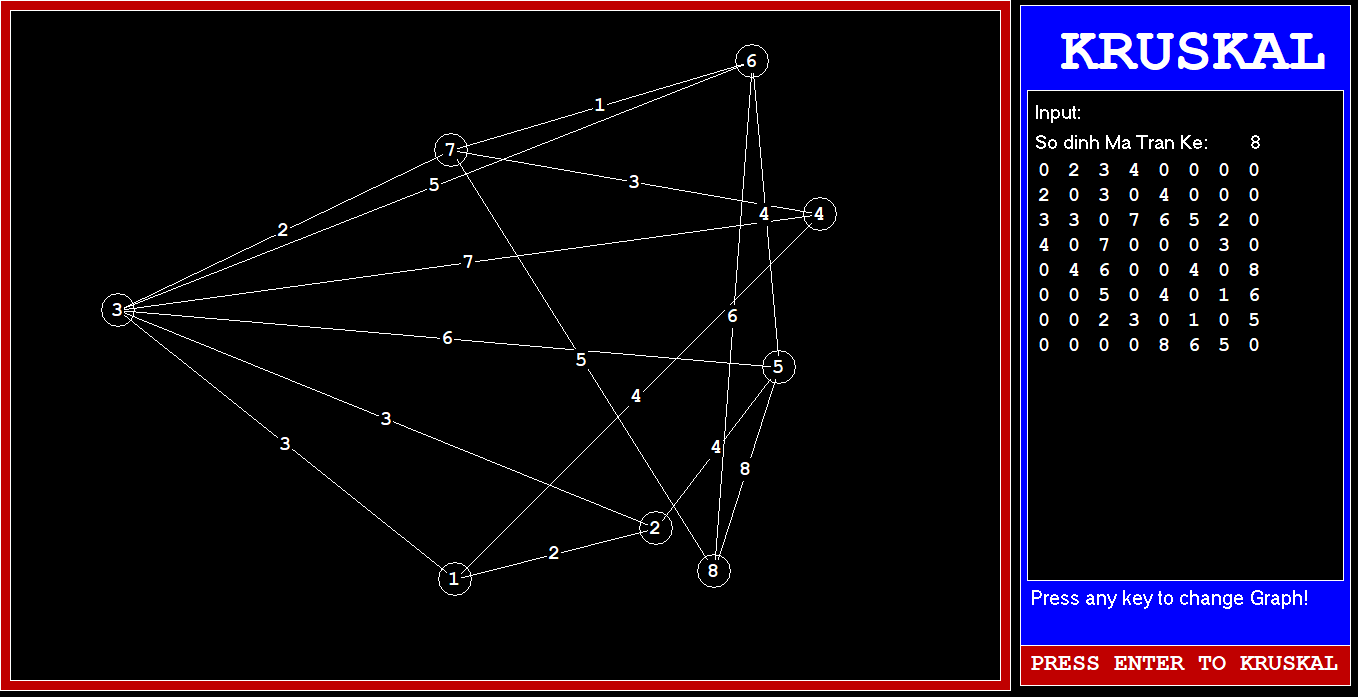
Phần thứ 1 là bảng bên trái là vùng thể hiện đồ thị có trọng số. Biểu diễn đồ thị một cách trực quan và tổng quan đối với người nhìn

Phần thứ 2 là bảng bên phải thể hiện phần Input và Output của thuật toán. Tức là đầu vào và đầu ra của thuật toán. Không chỉ biểu diễn ở màn hình giống như màn hình “Comman line” đơn sơ và đơn giản.

Việc vẽ các đối tượng thì cần sử dụng thư viện *#include<graphics.h>*. Thư viện đồ họa của ngôn ngữ C/C++. Và chương trìn sử dụng toàn bộ trong thư viện *graphis.h* để biểu diễn đồ họa.

### *Bảng biểu diễn đồ thị*

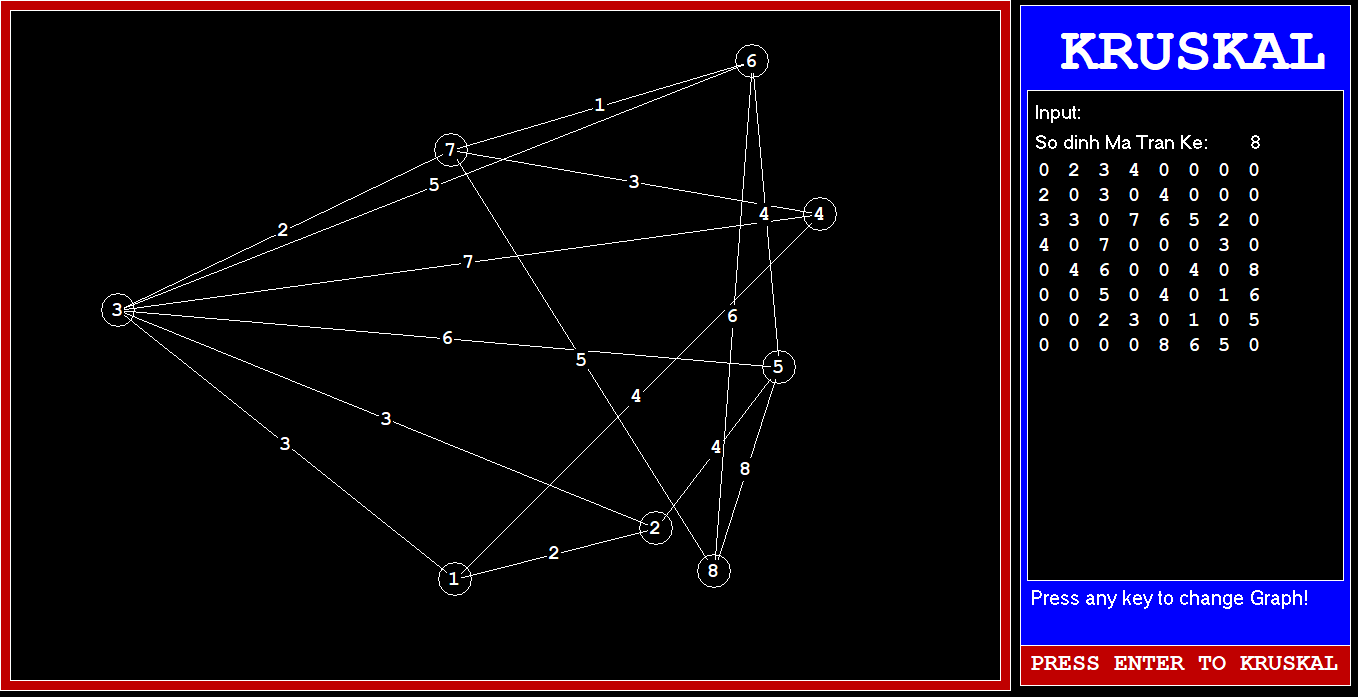
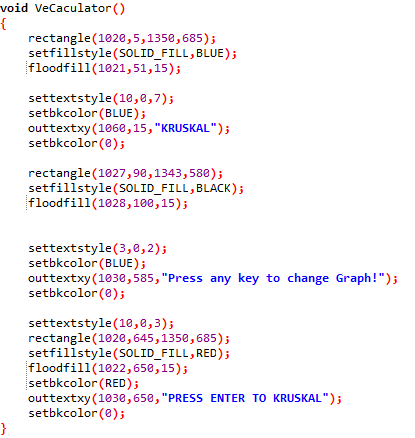
Đoạn mã vẽ phần bên tay trái.

Hình 4. Khung chính

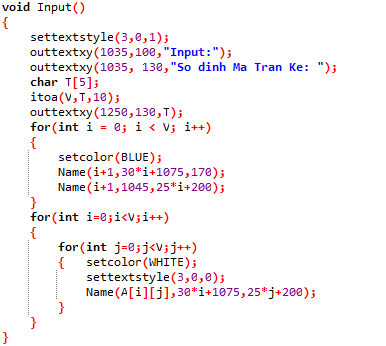
Sử dụng 2 hình chữ nhật để tạo nên 1 cái khung bao quanh hoặc có thể dử dụng 1 chiếc khung và tăng mức độ đậm của đường line lên. Ở chương trình này thì sử dụng 2 hình chữ nhật sau đó tô màu khoảng trống giữa 2 hình chữ nhật thì hình thành một khung chữ nhật màu đỏ, khoanh vùng biểu diễ của đồ thị.

### *Khung biểu diễn đầu vào và đầu ra*

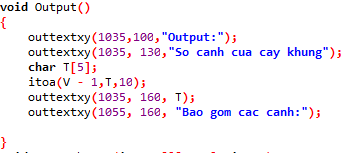


Hình 5. Khung phụ

Đây là đoạn mã vẽ phần bên phải. khu vực thể hiện Input và Output của chương trình. Trong phần này thì có thêm phần bảng đen thể hiện đầu vào và đầu ra của chương trình tương tự như 1 chiếc bảng đen để viết lên. Thì có 2 đoạn mã để làm chức năng in ra phần đầu vào và đầu ra ra màn hình window.

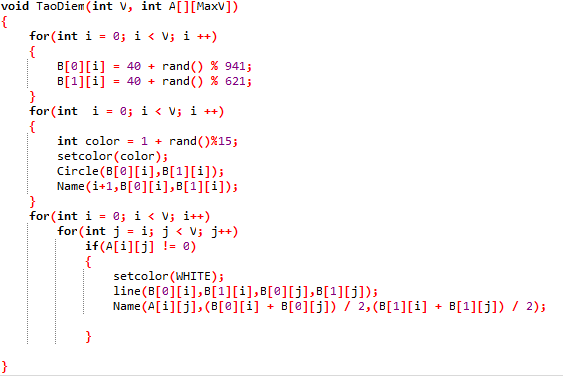


Sử dụng câu lệnh *outtextxy()* để xuất ra thông tin text và sử dụng 2 vòng lặp *for* để xuất ra các số của đồ thị thông qua ma trận kề bằng việc xác định toạn độ và sử dụng câu lệnh tăng dần tọa đồ và biển diễn rõ ràng trên đồ họa.



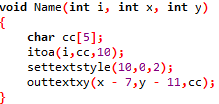
Tương tự như phần bên phải thì phần bên trai cũng có bảng đen để thể hiện đồ thì thì cũng có đoạn mã để thể hiện đồ thị

### *Xây dựng các thực thể*



Vì số lượng điểm của các đồ thì là khác nhau nên khi biểu diễn ta không quy định được điểm cụ thể cho các đỉnh. Nên sử dụng hàm rand() để gán cho các đỉnh có giá trị ngẫu nhiên. Trong đó sử dụng mang 2 chiều B[2][MaxV] để quy đinh cho các tọa độ x và y của các đỉnh.

Trong chương trình sử dụng 2 vòng lặp for để việc quét ma trận kề của đồ thị để biểu diễn trọng số lên đồ thị. Các trọng số được biểu diễn ở trung tâm các đoạn thẳng của các đỉnh. Thông qua hàm Name(i,x,y);



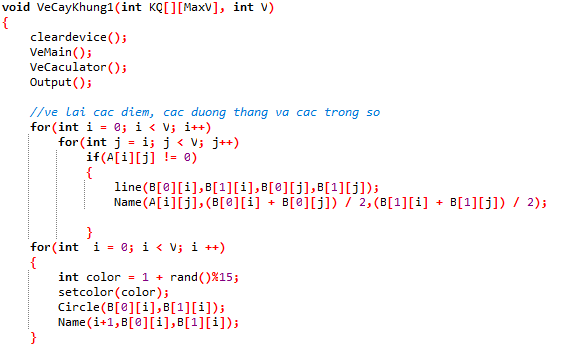
Vì do biểu diễn đồ họa không xuất ra được kiểu interger nên sử dụng hàm itoa để chuyển nó về kiểu char để xuất ra màn hình.

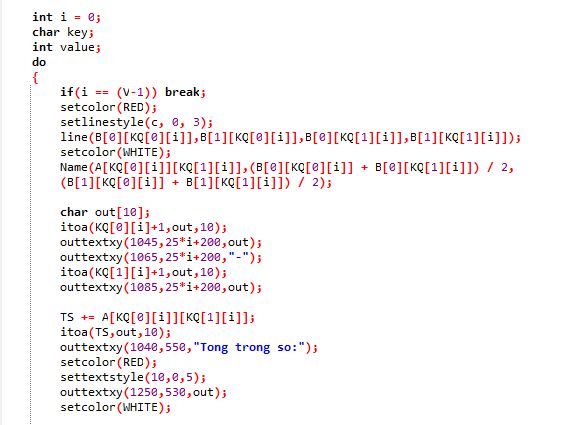


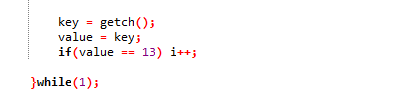
Sử dụng 1 hàm đơn giản để vẻ vòng tròn thể hiện các đỉnh của đồ thị.

Bên trong đó còn có các hàm dùng để biến đồ thị thành cây khung nhỏ nhất

### *Phần xử lý đồ họa*



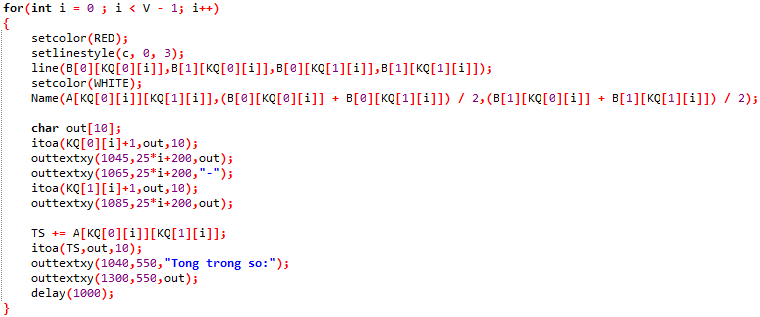




Sử dụng nhiều hàm định nghĩa sẳn để thể hiện việc các cây khung được hình thành như thế nào trong thực tế. Đa phần trong chuỗi câu lệnh trên là các câu lệnh thực hiện việc biểu diễn các đường các điểm các text ra màn hình với việc xác định bị trí rõ ràng.

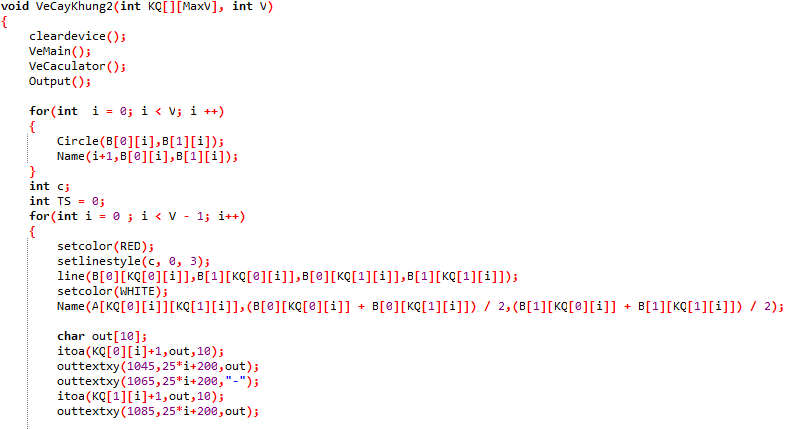
Ở chuỗi câu lệnh ở trên thể hiện cho việc thể hiện việc nhấn vào nút **ENTER** để thực hiện việc tìm cây khung theo từng cạnh 1 và thực thiện tuần tự theo thao tác của người sử dụng.

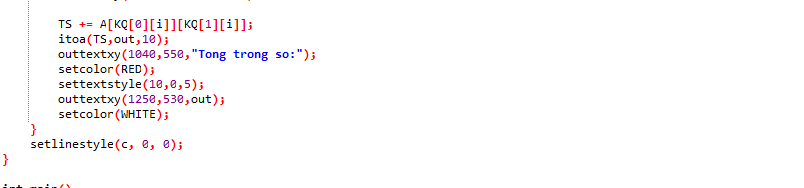
Bên cạnh đó chương trình còn mở rộng hơn về việc chỉ cần nhấn 1 lần là chuownh trình sẽ chạy tất cả với khoảng thời gian delay là 1 giây như đoạn code phía dưới đây.



Hàm phía trên là hàm dùng để thể hiện quá trình chọn lựa cây khung nhỏ nhất theo thứ tự tương tự như thuật toán Kruskal. Đoạn mã này sử dụng hết toàn bộ các phần giao diện để thực hiện chẳng hạn như vẽ cây đồ thị, vẽ bên trai bên phải, và in ra phần Output của chương trình trong phần đồ họa này.

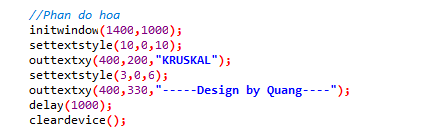
Cùng với việc thể hiện quá trình thể hiện quá trình thì tôi còn phát sinh thêm 1 chương trình con về việc thể hiện kết quả cuối cùng. Chương trình con này kế thừa hoàn toàn chương trình phía trên chỉ lược bỏ đi nhưng phần không cần thiết của phần thể hiện kết quả. Kế thừa chương trình con trên về phần chạy tự động và lược bộ đi phần chờ 1 giây cho mỗi cạnh thế là chương trình sẽ chạy thẳng đến đích để hiện ra kết quả cuối cùng.





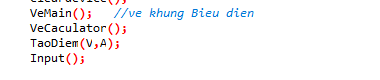
### *Chương trình chính*

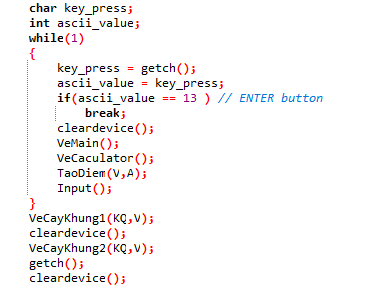
Đó là toàn bộ các chương trình con để xây dựng đồ họa. Việc chia nhỏ làm chương trình tối giản lại vì có nhiều thực thể tuy khác nhau nhưng vẫn sử dụng chung loại code chỉ khác ở phần dữ liệu vào. Còn trong chương trình chính phía sau phần code cấu trúc để thực hiện chương trình tìm cây khung là code xử lý các chương trình con đã xây dựng thành một thể thống nhất – chương trình hoàn chỉnh.





Hình 6. Giao diện mở đầu





Trước hết ta khai báo cửa sổ đồ họa có kích thước 1400,1000.

Đã là chương trình thì cần có phần intro tức là phần giới thiệu ngắn khoảng 1 giây. Bằng việc thể hiện tên thuật toán và tên người thiết kế sau đó xóa màn hình để nhường chỗ cho phần đồ họa thể hiện.

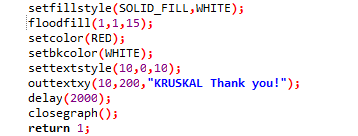
**B1**. Vẽ lại các đối tượng cần thiết để thể hiện ra màn hình. Gồm các chương trình: VeMain(), VeCaculator(), TaoDiem(V,A); và Input();

**B2**. Tạo vòng lặp để kiểm tra việc người dùng nhấn phím nào.

Trong chương trình này được quy định là nhấn bất kì phím nào để thay đổi đồ thì cho đến khi bạn nhấn phim “ENTER” thì chương trình sẽ thực hiện việc tìm cây khung.

**B3**. Sau khi hiển thị kết quả thì nhấn bất kì phím nào để đóng chương trình.

Đó là toàn bộ chương trình thuật toán tìm cây khung nhỏ nhất Kruskal theo ma trận kề. Cuối cùng sau khi chương trình hoàn thiện thì cần có một lời cảm ơn người dùng khi sử dụng chương trình.



Tô màu cho màn hình màu trắng sau đó cài đặt màu cho chữ và xuất ra màn hình. Chờ 2 giây sau đó thì đóng giao diện.



Hình 7. Giao diện kết thúc.

Và màn hình này xuất hiện thể hiện cho việc kết thúc chương trình. Cảm ơn người dùng, cảm ơn người thử, cảm ơn người xem, cảm ơn tất cả mọi người ….

# **KẾT LUẬN**

Thuật toán Kruskal là một thuật toán trong lý thuyết đồ thị để tìm cây bao trùm nhỏ nhất của một đồ thị liên thông có trọng số. Nói cách khác, nó tìm một tập hợp các cạnh tạo thành một cây chứa tất cả các đỉnh của đồ thị và có tổng trọng số các cạnh là nhỏ nhất.Thuật toán Kruskal là một ví dụ của thuật toán tham lam. Việc xây dựng thuật toán là điều cần thiết trong việc học và nghiên cứu lý thuyết đồ thị. Chương trình sẽ là công cụ đắc lực trong việc thao tác trên dồ thị. Hỗ trợ việc các bước tìm kiếm cây khung nhỏ nhất khi thao tác tay, tiết kiệm thời gian và khắc phục được lỗi thiếu chính xác của việc tìm bằng tay.

Chương trình đã giải quyết được vấn đề các đồ thị phức tạp trong việc tìm kiếm cây khung nhỏ nhất. Cùng với đó là việc mô phỏng việc tìm kiếm cây khung nhỏ nhất Krusal bằng ma trận kề. Việc mô phỏng đã hoàn thiện và thể hiện tốt việc tìm kiếm cây khung một cách rành mạch.

**PHƯƠNG HƯỚNG PHÁT TRIỂN:**

* Cải thiện đồ họa chi tiết
* Tăng thêm dữ liệu đầu vào: Đọc dữ liệu bản đồ, đọc dữ liệu hình ảnh,…
* Thao tác trực tiếp trên đồ họa: nhấn tạo điệm, tạo cạnh và tạo trọng số trực tiếp trên giao diện dồ họa.

# **TÀI LIỆU THAM KHẢO**

* Lý thuyết đồ thị - Lê minh hoàng
* Bài tập lý thuyết đồ thị 2018
* TOPICA bài 6 – các thuật toán tìm kiếm trên đồ thị và ứng dụng
* Lập trình C
* Kỹ thuật đồ họa – Đoàn Vũ thịnh