**BÁO CÁO TUẦN THỨ 4 VỀ TIẾN ĐỘ CÔNG VIỆC**

MỤC LỤC

[**Nhận xét:** 1](#_Toc60654959)

[**Phần thuật toán** 2](#_Toc60654960)

[*Mã lệnh thực hiện chương trình:* 2](#_Toc60654961)

[*Độ phức tạp thuật toán* 5](#_Toc60654962)

[**Phần đồ họa** 5](#_Toc60654963)

# **Nhận xét:**

# **Phần thuật toán**

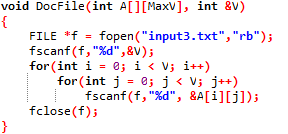
## *Mã lệnh thực hiện chương trình:*

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



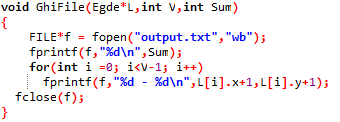
Chương trình được chia thành nhiều các chương trình nhỏ hơn để dễ quản lí và sửa chữa.

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.

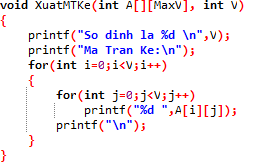


Nhưng để đọc file thì cần thêm thư viện stdlib.h để chương trình có thể thực hiện.

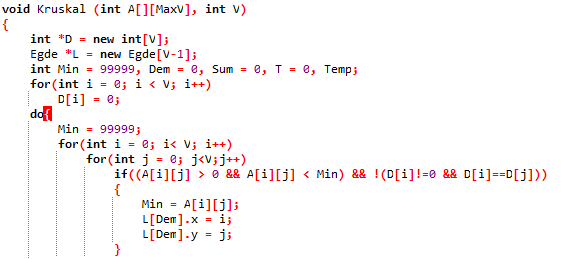
Đã lấy dữ liệu từ file thì cần xuất kết quả ra file.

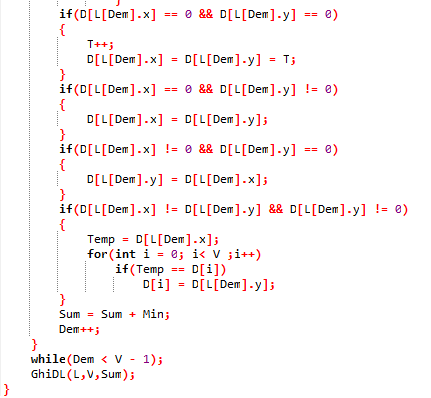


Cùng với việc xuất file thì cũng có chương trình xuất ra màn hình để xem được phần INPUT của chương trình.



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.





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



Phần đầu là khai báo các biến cần thiết của chương trình.

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

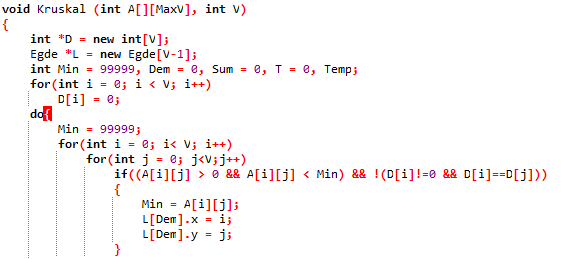
Ví dụ như đỉnh 1 thì có cha là 1 và đỉnh 2 có cha là 2.

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.

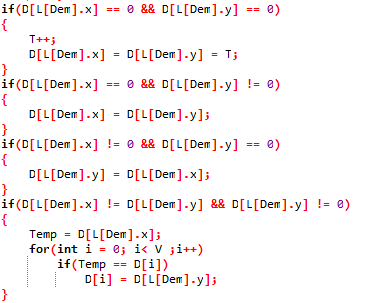
Biến T và biến Temp là 2 biến bổ trọ cho việc xét các cha của các đỉnh.

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.

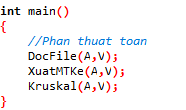


Đoạn mã này dùng để gán giá trị cha cho các đỉnh sau khi đã tìm được cạnh của cây khung.

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.

Trong chương trình chính chỉ cần gọi là các chương trình con đã nêu ở trên để chương trình cho thể thực hiện.

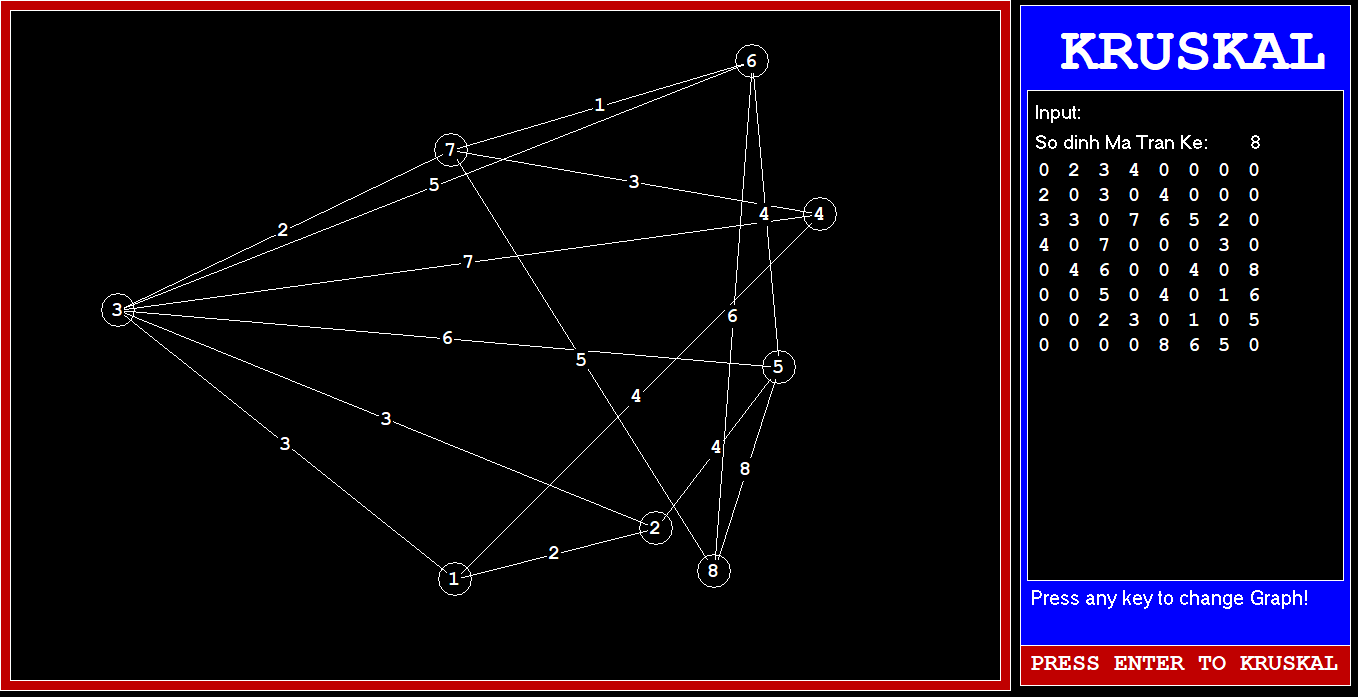


## *Độ phức tạp thuật toán*

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).**

# **Phần đồ họa**



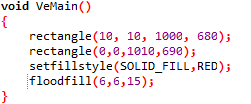
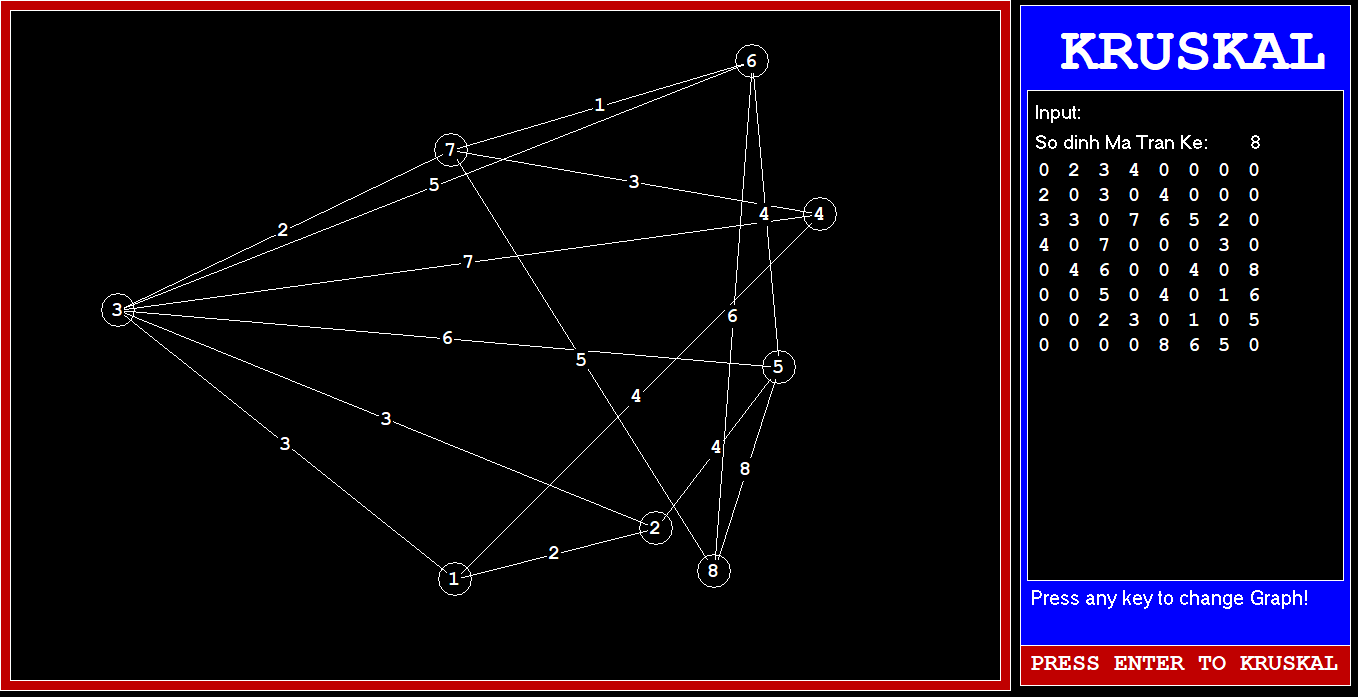
Trong phần đồ họa thì có 2 phần chính

Phần thứ 1 là bảng bên trái là vùng thể hiện đồ thị có trọng số.

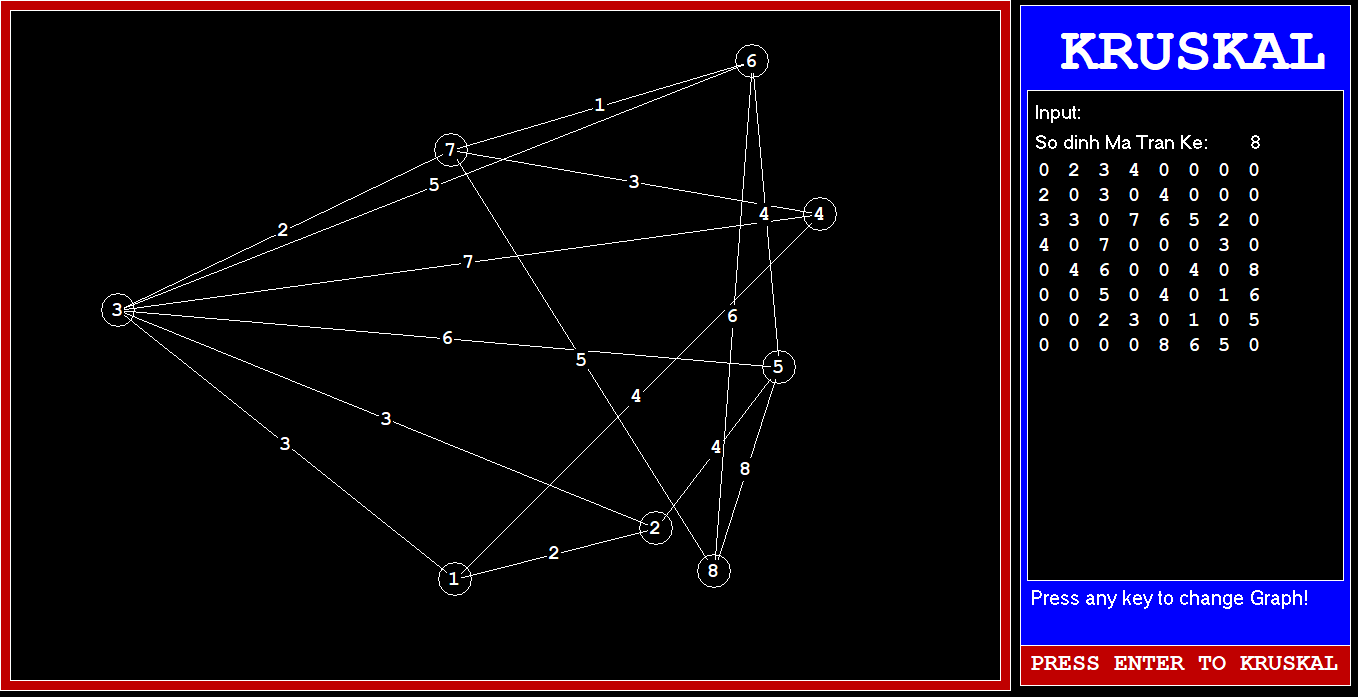
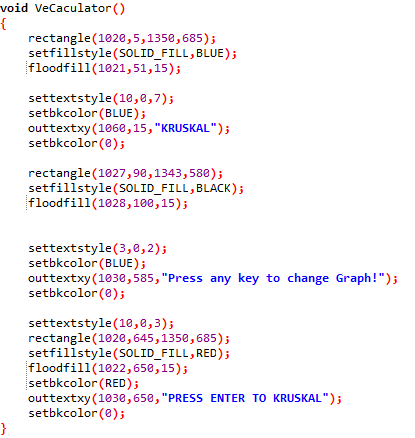
Phần thứ 2 là bảng bên phải thể hiện phần Input và Output của thuật toán.

Việc vẽ các đối tượng thì cần sử dụng thư viện .

Đoạn mã vẽ phần bên tay trái

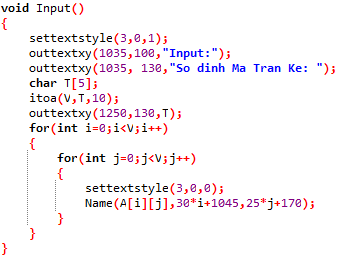
 

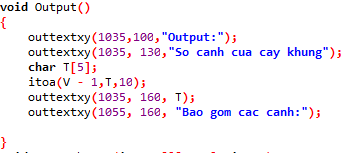
Sử dụng 2 hình chữ nhật để tạo nên 1 cái khung bao quanh hoắc có thể dử dụng 1 chiếc khung và tăng mức độ đậm của đường line lên.



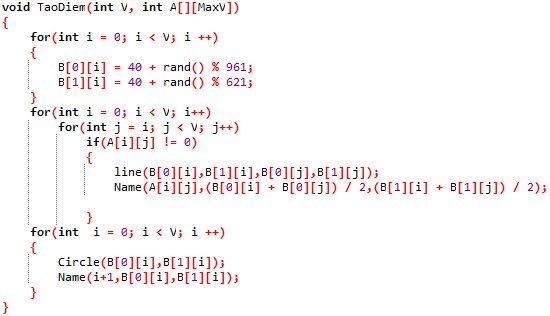
Đây là đoạn mã vẽ phần bên phải. khu vực thể hiện Input và Output của chương trình.

Trong phần này thì có thêm phần bảng đen thể hiện. thì có 2 đoạn mã để làm chức năng in ra phần đầu vào và đầu ra ra màn hình window.



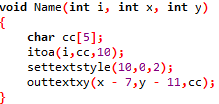


Tương tự như phần bên phải thì phần bên trai cũng có bảng đen để thể hiện đồ thì thì cũng có đoạn mã để thể hiện đồ thị



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);

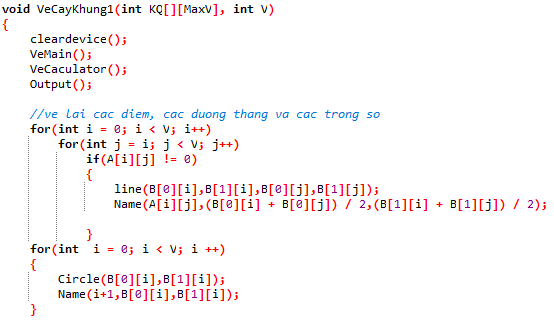


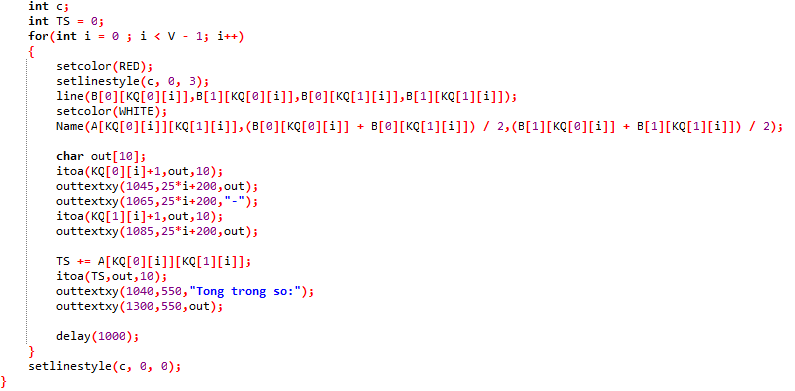
Vì do biểu diễn đồ họa không xuất ra được kiểu interger nên sử dụng hàm itoa để chuyển nó về kiểu char để xuất ra màn hình.



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

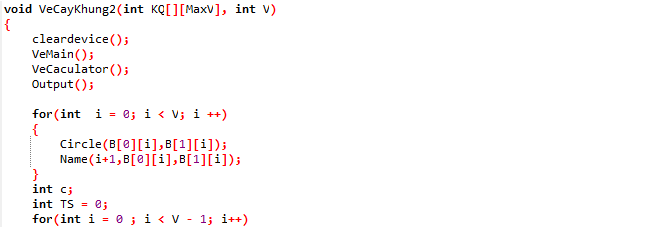


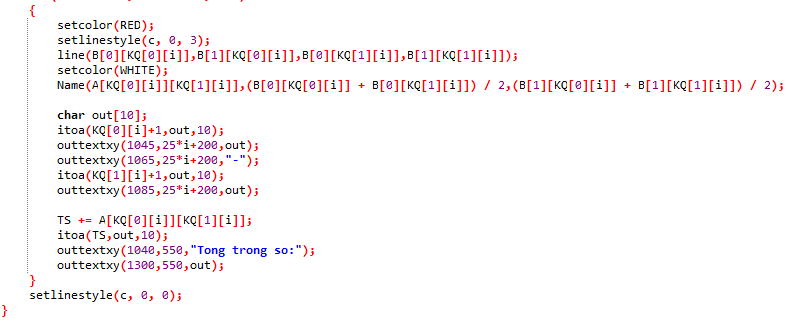


Sử dụng nhiều hàm định nghĩa sẳn để thể hiện việc các cây khung được hình thành như thế nào trong thực tế.

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.





Đó là toàn bộ các chương trình con để xây dựng đồ họa. Còn trong chương trình chính.



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

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.

Cảm ơn cô và các bạn đã xem.