**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HCM**

**KHOA ĐÀO TẠO CHẤT LƯỢNG CAO**

**NGÀNH CÔNG NGHỆ THÔNG TIN**

**🕯✡🕮🕮✡🕯**



**BÁO CÁO ĐỒ ÁN CÔNG NGHỆ THÔNG TIN**

**TUẦN 9**

***ĐỀ TÀI:* GRAPH**

**GVHD**: ThS. Nguyễn Quang Ngọc

**Sinh viên thực hiện**:

|  |  |
| --- | --- |
| **HỌ VÀ TÊN** | **MÃ SỐ SINH VIÊN** |
| Nguyễn Minh Quang | 20143481 |
| Đỗ Minh Cường | 21110147 |

## Tiến độ nhiệm vụ

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Tiến trình hoàn thành môn Đồ án CNTT | | | | | | | | | | | | | | | | | | |
|  | Các phương thức | TKB | | | | | | | | | | | | | Cường | | Quang | |
| 1 | Tạo 1 đồ thị | o | o |  |  |  |  |  |  |  |  |  |  |  | |  | |  | | |
| 2 | Thêm một đỉnh vào đồ thị đã có |  |  | o |  |  |  |  |  |  |  |  |  |  | |  | |  | | |
| 3 | Thay đổi thông tin của một đỉnh |  |  | o |  |  |  |  |  |  |  |  |  |  | |  | |  | | |
| 4 | Thêm cạnh |  |  |  | o |  |  |  |  |  |  |  |  |  | |  | |  | | |
| 5 | Thay đổi trọng số của cạnh |  |  |  |  | o |  |  |  |  |  |  |  |  | |  | |  | | |
| 6 | Xuất các tên đỉnh, tên cạnh |  |  |  | o |  |  |  |  |  |  |  |  |  | |  | |  | | |
| 7 | Xuất thông tin 1 đỉnh, 1 cạnh | o | o |  |  | o |  |  |  |  |  |  |  |  | |  | |  | | |
| 8 | Xuất ma trận kề, ma trận liên kết |  |  |  |  |  |  |  |  |  |  |  |  |  | |  | |  | | |
| 9 | Duyệt đồ thị theo chiều rộng, chiều xâu |  |  |  |  |  |  | o |  |  |  |  |  |  | |  | |  | | |
| 10 | Đường đi ngắn nhất từ đỉnh v đến đỉnh w |  |  |  |  |  | o |  |  |  |  |  |  |  | |  | |  | | |
| 11 | Kiểm tra có chu trình Euler |  |  |  |  |  |  |  | o |  |  |  |  |  | |  | |  | | |
| 12 | Cây khung bé nhất |  |  |  |  |  |  |  |  | o |  |  |  |  | |  | |  | | |
| Day | | 03/09/2023 | 10/09/2023 | 17/09/2023 | 24/09/2023 | 24/09/2023 | 30/09/2023 | 7/10/2023 | 14/10/2023 | 21/10/2023 |  |  |  |  | |  | |  | | |
| Week | | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 |  |  |  |  | |  | |  | | |
| Note | |  |  |  |  |  |  |  |  |  | o – Begin  o – Complete 50%  o – Complete 100% | | | | | | | | |

## Cây khung bé nhất:

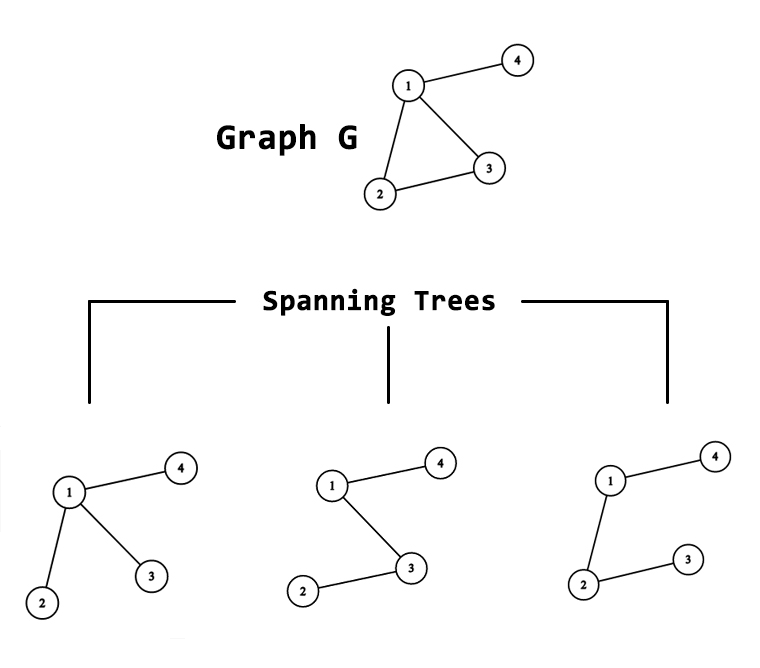
1. **Định nghĩa:**

Theo lý thuyết đồ thị, chúng ta đều biết rằng 1 đồ thị được biểu diễn bằng công thức *[Math Processing Error]*, trong đó đồ thị *[Math Processing Error]* của chúng ta bao gồm tập các đỉnh *[Math Processing Error]* và tập các cạnh *[Math Processing Error]*.

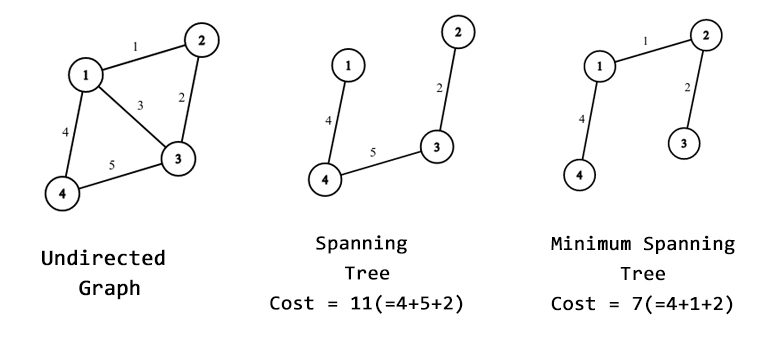
* ****Cây khung (**spanning tree**)**** của đồ thị là một tập hợp các cạnh của đồ thị thỏa mãn tập cạnh này **không chứa chu trình** và **liên thông** (từ một đỉnh bất kì có thể đi tới bất kỳ đỉnh nào khác theo mà chỉ dùng các cạnh trên ****cây khung****)

Trong đồ thị ****có trọng số****, ****cây khung nhỏ nhất (**minimum spanning tree**)**** là ****cây khung**** có tổng trọng số các cạnh trong cây ****nhỏ nhất****.

Một ví dụ về cây khung trong đồ thị vô hướng không trọng số:



Một ví dụ về cây khung ****nhỏ nhất**** trong đồ thị vô hướng có trọng số:



### **Tính chất**

Một vài tính chất của cây khung nhỏ nhất trong đồ thị vô hướng có trọng số:

****1. Tính chất chu trình****: Trong một chu trình *[Math Processing Error]* bất kỳ, nếu *[Math Processing Error]* là cạnh có trọng số lớn nhất ****tuyệt đối**** (không có cạnh nào có trọng số bằng *[Math Processing Error]*) thì *[Math Processing Error]* không thể nằm trên bất kỳ cây khung nhỏ nhất nào.

****2. Đường đi hẹp nhất****: Xét 2 đỉnh *[Math Processing Error]*, *[Math Processing Error]* bất kỳ trong đồ thị. Nếu *[Math Processing Error]* là trọng số của cạnh lớn nhất trên đường đi từ *[Math Processing Error]* đến *[Math Processing Error]* trên cây khung nhỏ nhất của đồ thị thì ta không thể tìm được đường đi nào từ *[Math Processing Error]* đến *[Math Processing Error]* trên đồ thị ban đầu chỉ đi qua những cạnh có trọng số nhỏ hơn *[Math Processing Error]*.

****3. Tính duy nhất****: Nếu tất cả các cạnh đều có trọng số khác nhau thì chỉ có duy một cây khung nhỏ nhất. Ngược lại, nếu một vài cạnh có trọng số giống nhau thì có thể có nhiều hơn một cây khung nhỏ nhất.

****4. Tính chất cạnh nhỏ nhất****: Nếu *[Math Processing Error]* là cạnh có trọng số nhỏ nhất của đồ thị, và không có cạnh nào có trọng số bằng *[Math Processing Error]* thì *[Math Processing Error]* nằm trong mọi cây khung nhỏ nhất của đồ thị.

## **Các thuật toán tìm cây khung nhỏ nhất**

### **1. Thuật toán Kruskal**

****Ý tưởng thuật toán****: Ban đầu mỗi đỉnh là một cây riêng biệt, ta tìm cây khung nhỏ nhất bằng cách duyệt các cạnh theo trọng số từ nhỏ đến lớn, rồi hợp nhất các cây lại với nhau.

Cụ thể hơn, giả sử cạnh đang xét nối 2 đỉnh *[Math Processing Error]* và *[Math Processing Error]*, nếu 2 đỉnh này nằm ở 2 cây khác nhau thì ta thêm cạnh này vào cây khung, đồng thời hợp nhất 2 cây chứa *[Math Processing Error]* và *[Math Processing Error]*.

Giả sử ta cần tìm cây khung nhỏ nhất của đồ thị *[Math Processing Error]*. Thuật toán bao gồm các bước sau:

* Khởi tạo rừng *[Math Processing Error]* (tập hợp các cây), trong đó mỗi đỉnh của G tạo thành một cây riêng biệt.
* Khởi tạo tập *[Math Processing Error]* chứa tất cả các cạnh của *[Math Processing Error]*.
* Chừng nào *[Math Processing Error]* còn ****khác rỗng**** và *[Math Processing Error]* gồm ****hơn một cây****
* Xóa cạnh nhỏ nhất trong *[Math Processing Error]*
* Nếu cạnh đó nối hai cây khác nhau trong *[Math Processing Error]*, thì thêm nó vào *[Math Processing Error]* 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 khung nhỏ nhất của đồ thị *[Math Processing Error]*

### **2. Thuật toán Prim:**

****Ý tưởng thuật toán****: Ý tưởng của thuật toán Prim rất giống với ý tưởng của thuật toán Dijkstra (tìm đường đi ngắn nhất trên đồ thị). Nếu như thuật toán ****Kruskal**** xây dựng cây khung nhỏ nhất bằng cách kết nạp từng ****cạnh**** vào đồ thị thì thuật toán ****Prim**** lại kết nạp từng ****đỉnh**** vào đồ thị theo tiêu chí: đỉnh được nạp vào tiếp theo phải ****chưa được nạp**** và ****gần nhất**** với các đỉnh đã được nạp vào đồ thị.

Thuật toán bao gồm các bước sau:

* Khởi tạo tập *[Math Processing Error]* là cây khung hiện tại, ban đầu ****S**** chưa có đỉnh nào.
* Khởi tạo mảng *[Math Processing Error]* trong đó *[Math Processing Error]* là khoảng cách ngắn nhất từ đỉnh *[Math Processing Error]* đến 1 đỉnh đã được kết nạp vào tập *[Math Processing Error]*, ban đầu *[Math Processing Error]*= *[Math Processing Error]+∞*
* Lặp lại các thao tác sau *[Math Processing Error]* lần(*[Math Processing Error]* là số cạnh của đồ thị)
* Tìm đỉnh *[Math Processing Error]* không thuộc *[Math Processing Error]* có *[Math Processing Error]* nhỏ nhất, thêm *[Math Processing Error]* vào tập *[Math Processing Error]*.
* Xét tất cả các đỉnh *[Math Processing Error]* kề *[Math Processing Error]*, cập nhật *[Math Processing Error]* với *[Math Processing Error]* là trọng số cạnh *[Math Processing Error]*. Nếu *[Math Processing Error]* được cập nhật theo *[Math Processing Error]* thì đánh dấu *[Math Processing Error]*.
* Thêm cạnh *[Math Processing Error]* vào tập cạnh thuộc cây khung nhỏ nhất.

Khi hoàn thành xong *[Math Processing Error]* bước trên, ta thu được cây khung nhỏ nhất của đồ thị gồm *[Math Processing Error]* đỉnh và *[Math Processing Error]* cạnh.

## Thiết kế

1. **Cây khung bé nhất:**

Tạo hàm primMST() được viết dựa trên thuật toán Prim.

void primMST(int a[][100], int n) {

int parent[100];

int key[100];

bool mstSet[100];

for (int i = 0; i < n; i++) {

key[i] = MAX;

mstSet[i] = false;

}

key[0] = 0;

parent[0] = -1;

for (int count = 0; count < n - 1; count++) {

int u = minKey(key, mstSet, n);

mstSet[u] = true;

for (int v = 0; v < n; v++) {

if (a[u][v] && !mstSet[v] && a[u][v] < key[v]) {

parent[v] = u;

key[v] = a[u][v];

}

}

}

printMST(a, parent, n);

}

Viết hàm minKey() tìm ra đỉnh có trọng số nhỏ nhất trong danh sách các đỉnh chưa được thêm vào cây khung bé nhât.

int minKey(int key[], bool mstSet[], int n) {

int min = MAX, minIndex;

for (int v = 0; v < n; v++) {

if (!mstSet[v] && key[v] < min) {

min = key[v];

minIndex = v;

}

}

return minIndex;

}

Tạo hàm printMST() in ra cây khung bé nhất.

void printMST(int a[][100], int parent[], int n) {

cout << "Cay khung be nhat:\n";

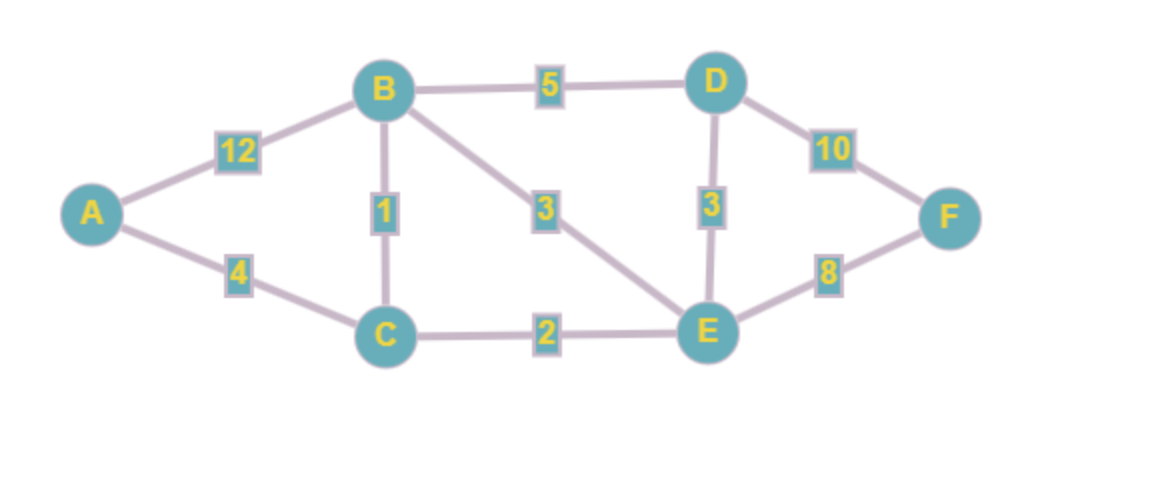
for (int i = 1; i < n; i++) {

cout << "Canh: " << vertex[parent[i]] << " - " << vertex[i] << " co trong so: " << a[i][parent[i]] << endl;

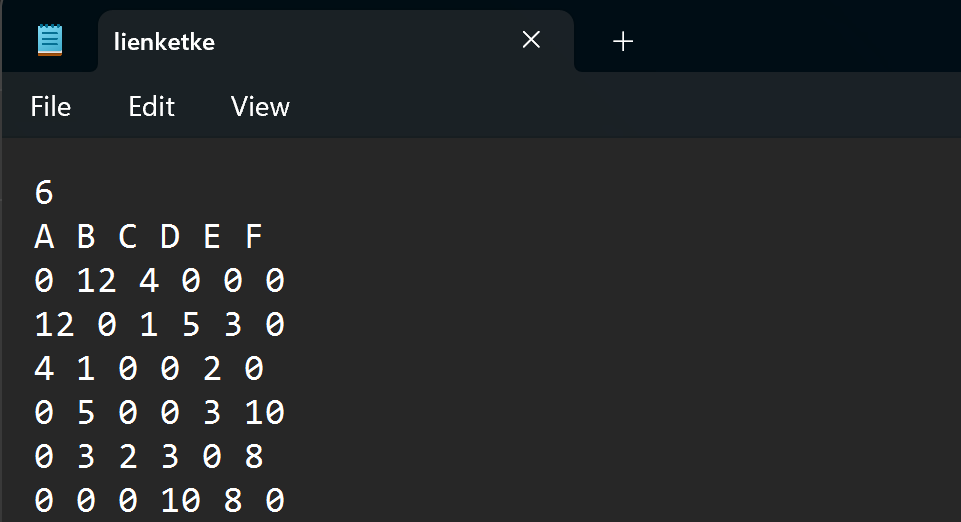
}

}

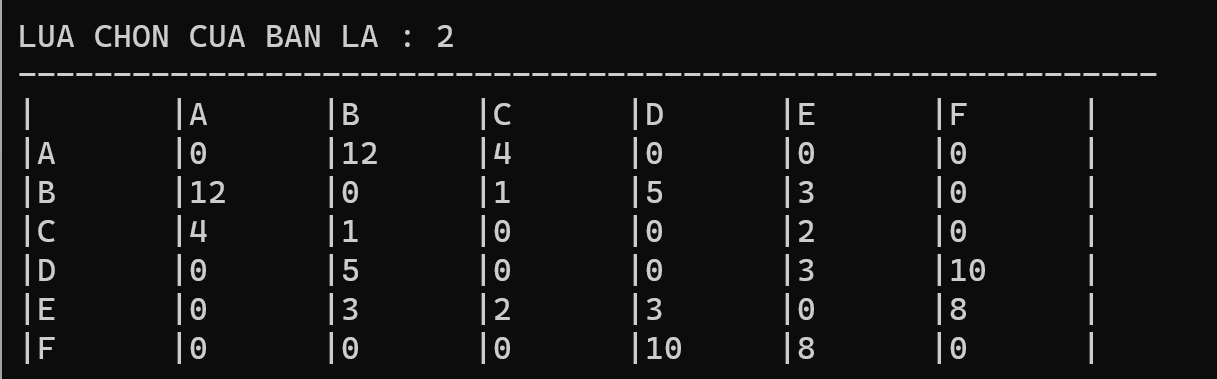
Tạo 1 đồ thị như hình vẽ để thử nghiệm:



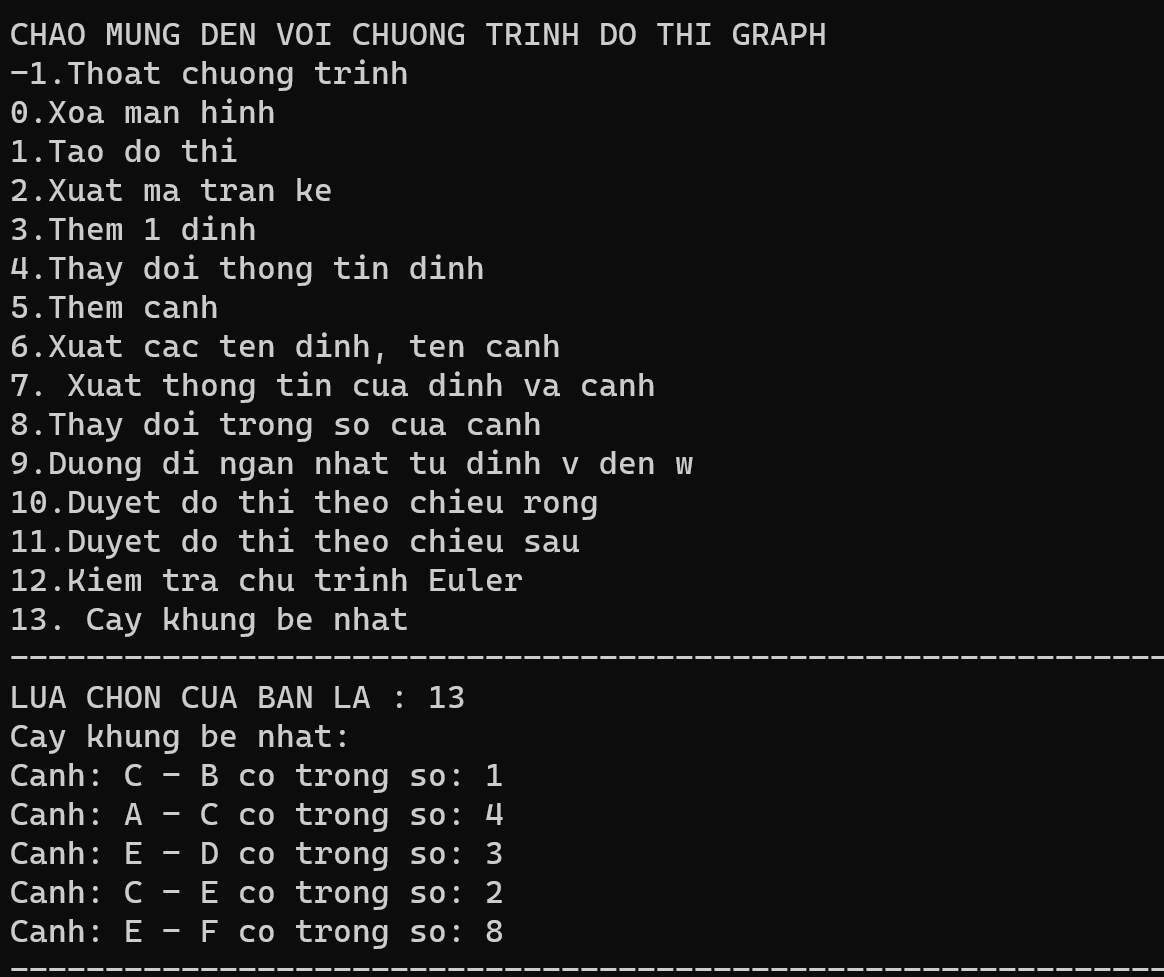
Lưu vào file lienketke:



Khi chạy lưu file và xem đồ thị trong chương trình:



Chọn số 13, tìm cây khung bé nhất:



Kết quả ra cây khung bé nhất và tổng trọng số là 18. Kết quả chính xác theo trang web <https://graphonline.ru/en/>

