

HƯỚNG DẪN THỰC HÀNH TÌM CÂY KHUNG NHỎ NHẤT - PRIM

1 Thuật toán

Cho $G=(X,E)$ là một đồ thị liên thông có trọng số gồm n đỉnh. Thuật toán Prim được dùng để tìm ra cây khung nhỏ nhất của G .

Bước 1: Chọn tùy ý đỉnh $v \in X$ và khởi tạo $Y := \{v\}$; $T := \emptyset$. Trong đó,

- X là tập các đỉnh của đồ thị.
- Y là tập các đỉnh được chọn vào cây khung nhỏ nhất.
- T là tập các cạnh của cây này.

Bước 2: Trong số những cạnh e nối đỉnh w với đỉnh v , với $w \in X \setminus Y$ và $v \in Y$ ta chọn cạnh có trọng số nhỏ nhất.

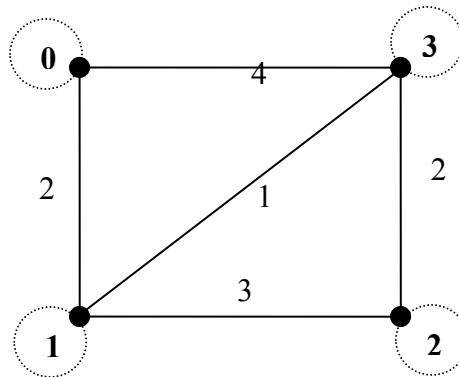
Bước 3: Gán $Y := Y \cup \{w\}$ và $T := T \cup \{e\}$

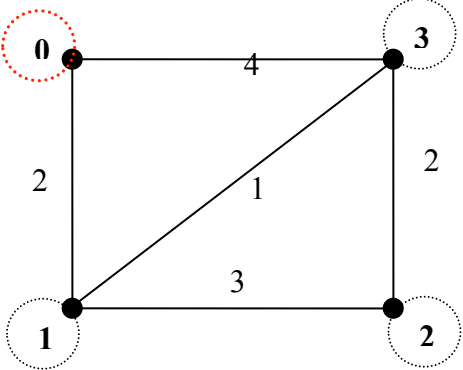
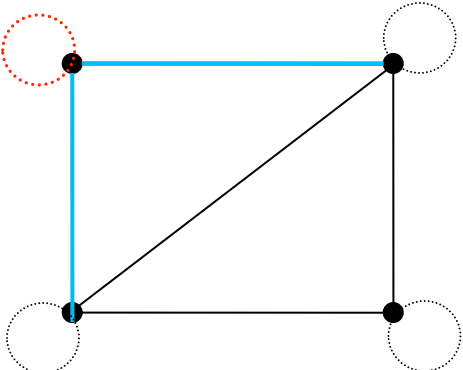
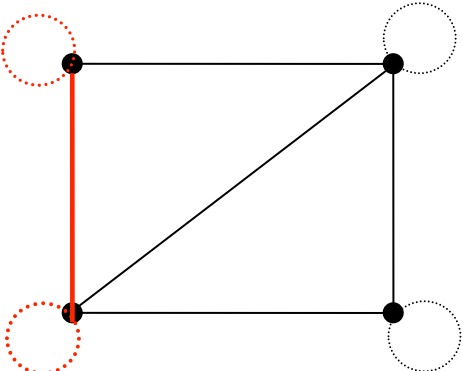
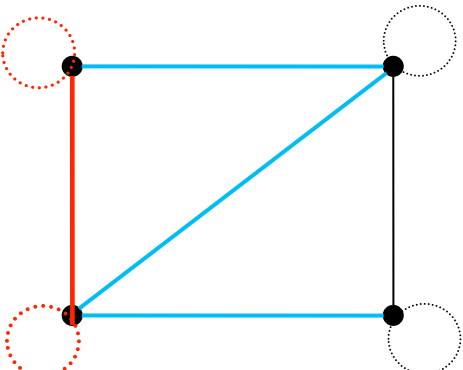
Bước 4: Nếu T đủ $n - 1$ phần tử thì dừng, ngược lại làm tiếp tục bước 2.

Chú ý: trong các thuật toán tìm khung nhỏ nhất chúng ta có thể bỏ đi hướng các cạnh và các khuyên; đối với cạnh song song thì có thể bỏ đi và chỉ để lại một cạnh trọng lượng nhỏ nhất trong chúng.

2 Ví dụ

Tìm cây khung nhỏ nhất của đồ thị sau:



	<p><u>Bước 1:</u> Chọn tùy ý $v \in X$ và khởi tạo $Y := \{v\}; T := \emptyset$</p> <p>Tập X là tập các đỉnh của đồ thị:</p> $X = \{0,1,2,3,4\}$ <p>Ta chọn đỉnh 0 làm đỉnh xét đến đầu tiên:</p> $Y = \{0\}; T = \emptyset \rightarrow X \setminus Y = \{1,2,3,4\}$
	<p><u>Bước 2:</u> Trong số những cạnh e nối đỉnh w với đỉnh v trong Y với $w \in X \setminus Y$ và $v \in Y$ ta chọn cạnh có trọng lượng nhỏ nhất.</p> <p>Cạnh (3,0) và (1,0) nối đỉnh 3 và 1 đến đỉnh 0 trong Y, ta chọn cạnh (1,0) vì nó có trọng lượng nhỏ hơn.</p>
	<p><u>Bước 3:</u> Gán $Y := Y \cup \{w\}$ và $T := T \cup \{e\}$</p> $Y = \{0, 1\}$ $T = \{(1,0)\}$ <p><u>Bước 4:</u> Nếu T đủ $n-1$ phần tử thì dừng, ngược lại làm tiếp tục bước 2.</p> <p>T chỉ có 1 phần tử $< n - 1 = 4 - 1 = 3$ nên thuật toán chưa dừng.</p>
	<p>Bước 2 (lần 2):</p> <p>Cạnh (3,0); (3,1); (2,1) là các cạnh nối đỉnh 3; 2 (tập những đỉnh chưa có trong cây) đến đỉnh 0 và 1 (tập các đỉnh đã có trong cây). Tương tự trên, ta chọn cạnh có trọng lượng nhỏ nhất: (3, 1).</p>

	<p><i>Bước 3 (lần 2):</i></p> <p>$Y = \{0, 1, 3\}$</p> <p>$T = \{(1, 0), (3, 1)\}$</p> <p>T chỉ có đủ phần tử nên thuật toán tiếp tục chạy.</p>
	<p>Lúc này T đã chứa đủ 3 cạnh vì vậy thuật toán dừng.</p> <p>Tập đỉnh: $Y = 0, 1, 3, 2$</p> <p>Tập cạnh: $T = \{(0, 1); (3, 1); (2, 3)\}$</p>

3 Tổ chức lớp

Lớp **EDGE** thể hiện một cạnh của cây khung.

```
class EDGE
{
private:
    int v; // Đỉnh thứ nhất
    int w; // Đỉnh thứ hai
}
```

Lớp **SpanningTree** thể hiện một cây khung.

```
class SpanningTree
{
private:
    vector<EDGE> Tree;
}
```

...
 Hàm tìm cây khung là một phương thức của lớp Graph.

```
SpanningTree PrimAlg()
{
    int lblVertex[MAX]; //nhãn đánh dấu đỉnh nào đã xét (1) và đỉnh nào chưa xét (0).
                        // Đóng vai trò như tập X và Y trong thuật toán nói trên.
    int nLbl;           //số phần tử của mảng lblVertex

    // Khởi tạo cây khung rỗng
    // khởi tạo nhãn của các đỉnh là chưa xét (0)
    ...
    // gán nhãn đã xét (1) cho đỉnh 0
    ...
    while (số cạnh của cây khung chưa đủ)
    {
        EDGE edgeMin;
        int nMinWeight = -1; // -1 nghĩa là chưa có cạnh min
        // duyệt các đỉnh w thỏa điều kiện chưa xét
        for (w...)
            if (...)
            {
                // tìm một đỉnh v bất kỳ đã xét và có
                // cung nối (trực tiếp) giữa v & w
                for (v...)
                    if (... && ...)
                    {
                        // kiểm tra nếu chưa có cạnh min (nMinWeight <
                        // hoặc trọng của (v, w) nhỏ hơn nMinWeight
                        // thì cập nhật lại edgeMin = (v, w)
                        if (... || ...)
                        {
                            edgeMin.v = v; edgeMin.w = w;
                            nMinWeight = ...;
                        }
                    }
            }
        // thêm cạnh edgeMin vào cây khung
        // gán nhãn đã xét cho đỉnh w
        ...
    }
}
```