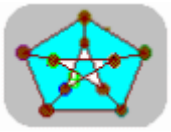
Lý Thuyết Đồ Thị
Đại Học Khoa Học Tự Nhiên Tp.HCM
227 Nguyễn Văn Cừ, Q5

GVHD: Lê Ngọc Thành
E-mail: lnthanh@fit.hcmuns.edu.vn
http://courses.cs.hcmuns.edu.vn



Tìm cây khung nhỏ nhất với Prim

1.Thuật toán Prim

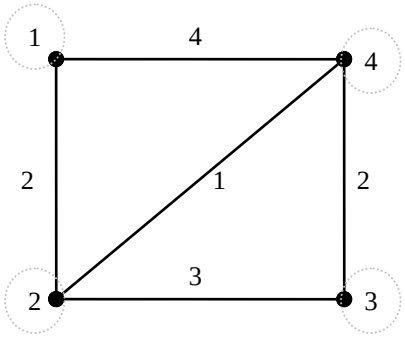
Cho G=(X,E) là một đồ thị liên thông có trọng gồm n đỉnh. Thuật toán Prim được dùng để tìm ra cây khung ngắn nhất của G.

Bước 1: Chọn tùy ý v $\in$ X và khởi tạo Y:= {v}; T := Ø. Trong đó X là tập các đỉnh của đồ thị, Y là tập các đỉnh được chọn vào cây khung ngắn nhất và T là tập các cạnh của cây này.
Bước 2: Trong số những cạnh e nối đỉnh w với đỉnh v trong Y với w $\in$ X\Y và v $\in$ Y ta chọn cạnh có trọng lượng 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.

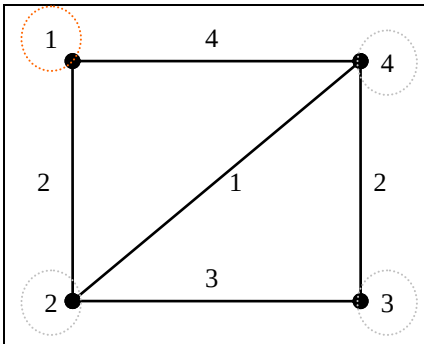
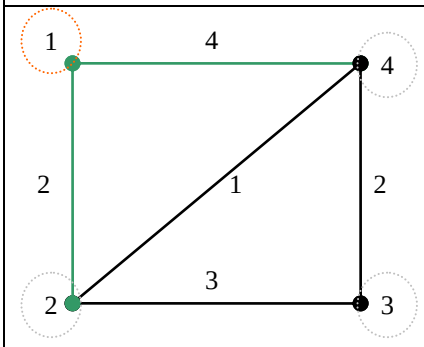
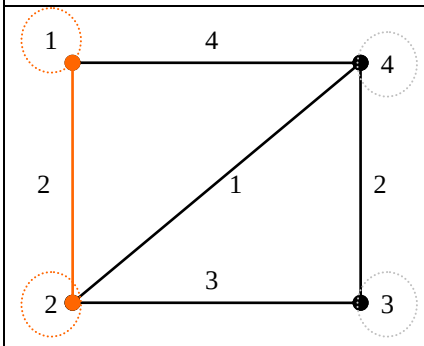
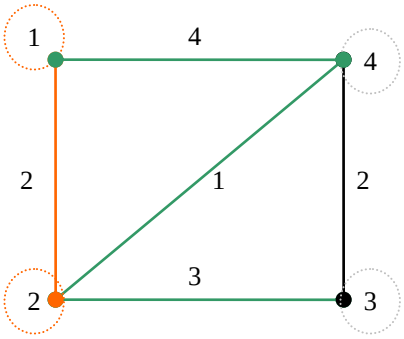
Chú ý: trong các thuật toán tìm khung ngắn nhất chúng ta có thể bỏ đi hướng các cạnh và các khuyên; đối với cạnh song song thì có thể bỏ đi và chỉ để lại một cạnh trọng lượng nhỏ nhất trong chúng.

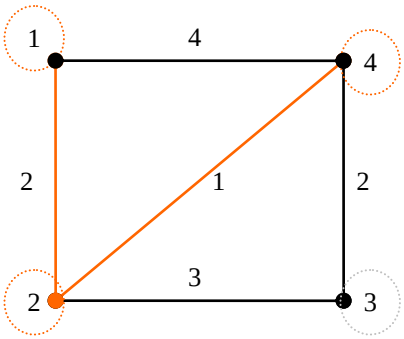
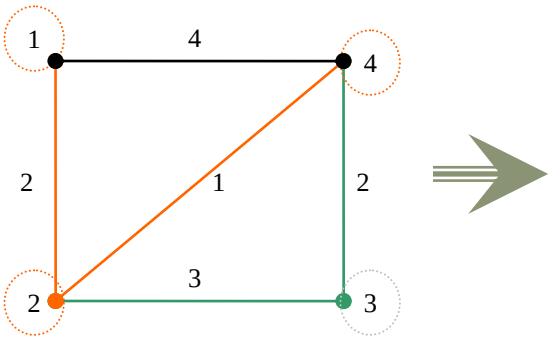
2. Ví dụ Prim

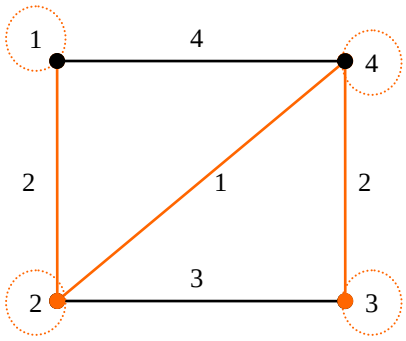
Cho đồ thị sau:



Tìm cây khung ngắn nhất của đồ thị.

	Bước 1: *Chọn tùy ý v ∈X và khởi tạo Y:= {v}; T := Ø* Tập X là tập các đỉnh của đồ thị nên X = {1,2,3,4,5} Ta chọn đỉnh **1** làm đỉnh xét đến đầu tiên: Y = {1}; T = Ø. Như vậy X\Y = {2,3,4,5}
	Bước 2: *Trong số những cạnh e nối đỉnh w với đỉnh v trong Y với w ∈ X\Y và v ∈ Y ta chọn cạnh có trọng lượng nhỏ nhất.* Cạnh (4,1) và (2,1) nối đỉnh 2 và 4 đến đỉnh 1 trong Y, ta chọn cạnh (2,1) vì nó có trọng nhỏ nhất trong hai cạnh (giá trị là 2).
	Bước 3: *Gán Y:= Y∪ {w} và T:= T∪ {e}* Y = Y ∪{2} T = T ∪{(2,1)} Bước 4: *Nếu T đủ n-1 phần tử thì dừng, ngược lại làm tiếp tục bước 2.* T chỉ có 1 phần tử < n – 1 = 4 – 1 = 3 nên thuật toán chưa dừng.
	Bước 2 (lần 2): Cạnh (4,1); (4,2); (3,2) là các cạnh nối đỉnh 4; 3 (tập những đỉnh chưa có trong cây) đến 1;2 (tập các đỉnh đã có trong cây). Tương tự, ta chọn cạnh có trọng lượng nhỏ nhất: (4,2).

<table>
<tr>
<td> 
</td>
<td>Bước 3 (lần 2):
Y = Y ∪ {4}
T = T ∪ {(4,2)}
T chỉ có đủ phần tử nên thuật toán tiếp tục chạy.</td>
</tr>
<tr>
<td> 

Tương tự cho các bước lặp kế tiếp.</td>
<td> 

Lúc này T đã chứa đủ 3 cạnh vì vậy thuật toán dừng.
Tập đỉnh: Y = 1,2,4,3
Tập cạnh: T = {(1,2); (4,2); (3,4)}</td>
</tr>
</table>

3. Hướng dẫn cài đặt Prim

```
#define MAX 100

typedef struct {
        int n;
        int L[MAX][MAX]; // ma trận trọng số
}GRAPH;

typedef struct {
        int v;          //đỉnh thứ nhất
        int w;          //đỉnh thứ hai
}EDGE;

EDGE T[MAX];            //cạnh của cây khung ngắn nhất
int nT;                 //số cạnh của cây T
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

…
void PrimAlg(GRAPH g)
{
        // gán số cạnh của cây khung nT ban đầu là 0
        …
        // khởi tạo nhãn của các đỉnh là chưa xét (0)
```

```
	…
	// gán nhãn đã xét (1) cho đỉnh 0
	…
	while (nT < …)  // số cạnh tối đa của cây khung
	{
		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 < 0)
				// 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
		T[nT] = edgeMin;
		nT++;
		// gán nhãn đã xét cho đỉnh w
		…
	}
}
…
```

Chú ý: ta cũng cần xác định đồ thị đưa vào có liên thông hay không trước khi gọi hàm PrimAlg đồng thời chú ý đồ thị có hướng (xét cả hai chiều và lấy cạnh có trọng nhỏ nhất!!!*quan trọng*!!!).