Chuong Le Hoang
Open University

Tháng Mười 3, 2014

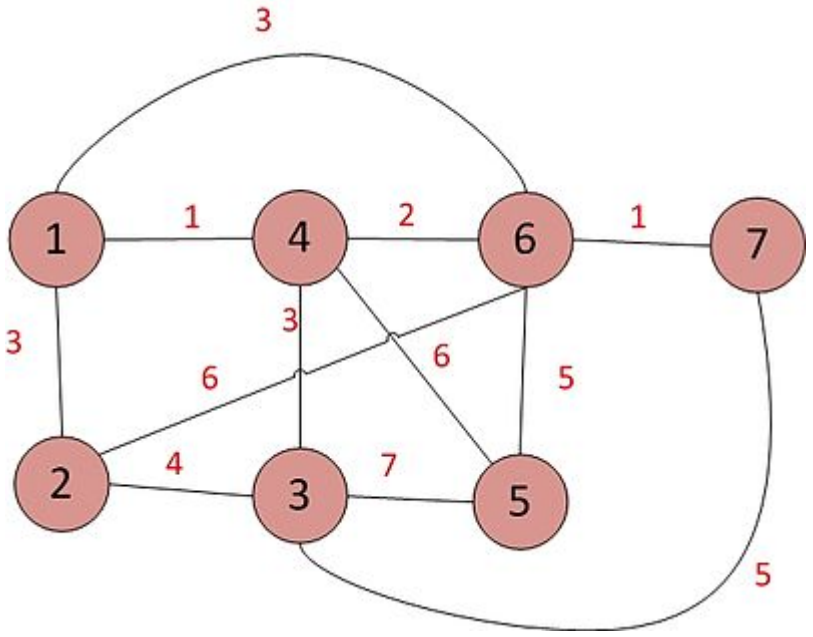
Thuật toán Kruskal – Tìm cây bao trùm nhỏ nhất

4 phản hồi

1. Mô tả:

– Giống như Prim, thuật toán Kruskal cũng tìm cây khung nhỏ nhất nhưng không phụ thuộc vào điểm bắt đầu.

– ***Hình 1***, đồ thị như sau:



(https://lhchuong.files.wordpress.com/2014/10/400px-c491e1bb93_the1bb8b_g_trong_lc3bd_thuye1babft_c491e1bb93_the1bb8b.jpg)

Hình 1

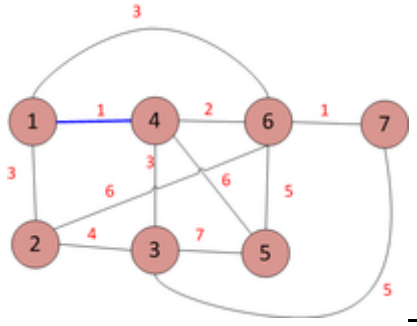
– **Bước 1**: lập bảng, liệt kê tăng dần theo trọng số của các cạnh.

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

(https://lhchuong.files.wordpress.com/2014/10/untitled4.png)

– **Bước 2**: dựa vào thứ tự bảng trên để đánh giá cạnh đó có thuộc cây khung tối tiểu hay không. Một cạnh thõa điều kiện khi nó không góp phần tạo thành một chu trình với tất cả các cạnh đã tìm được.

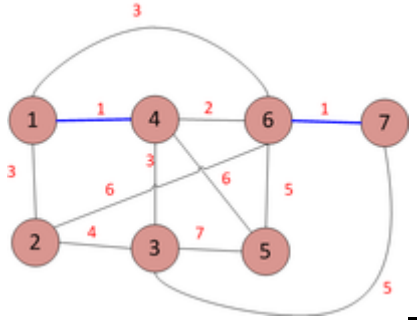
1-4-1*: Ta nhận thấy cạnh 1-4 không tạo ra một chu trình nào. Vì vậy, thêm 1-4 vào tập hợp*



(https://lhchuong.files.wordpress.com/2014/10/11.png)

Hình 2

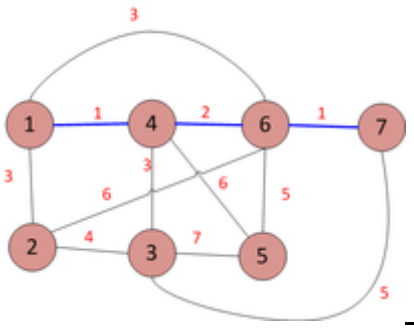
6-7-1*: Ta nhận thấy cạnh 6-7 không tạo ra một chu trình nào. Vì vậy, thêm 6-7 vào tập hợp*



(https://lhchuong.files.wordpress.com/2014/10/22.png)

Hình 3

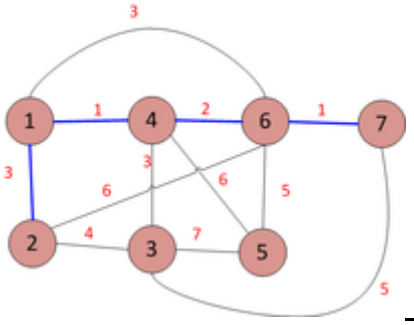
***4-6-2**: Ta nhận thấy cạnh 4-6 không tạo ra một chu trình nào. Vì vậy, thêm 4-6 vào tập hợp*



(https://lhchuong.files.wordpress.com/2014/10/32.png)

Hình 4

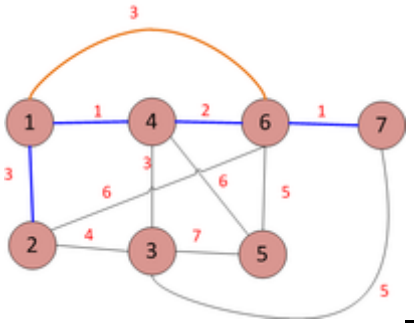
***1-2-3**: Ta nhận thấy cạnh 1-2 không tạo ra một chu trình nào. Vì vậy, thêm 1-2 vào tập hợp*



(https://lhchuong.files.wordpress.com/2014/10/41.png)

Hình 5

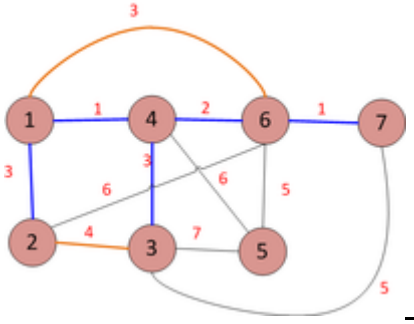
***1-6-3**: Ta nhận thấy cạnh 1-6 tạo ra một chu trình. Không thêm vào tập hợp.*



(https://lhchuong.files.wordpress.com/2014/10/51.png)

Hình 6

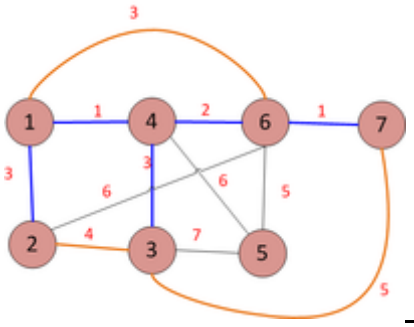
***2-3-4**: Ta nhận thấy cạnh 2-3 tạo ra một chu trình. Không thêm vào tập hợp.*



(https://lhchuong.files.wordpress.com/2014/10/61.png)

Hình 7

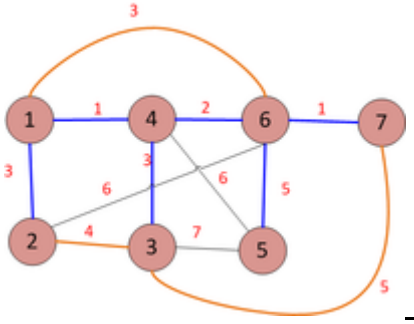
***3-7-5**: Ta nhận thấy cạnh 3-7 tạo ra một chu trình. Không thêm vào tập hợp.*



(https://lhchuong.files.wordpress.com/2014/10/71.png)

Hình 8

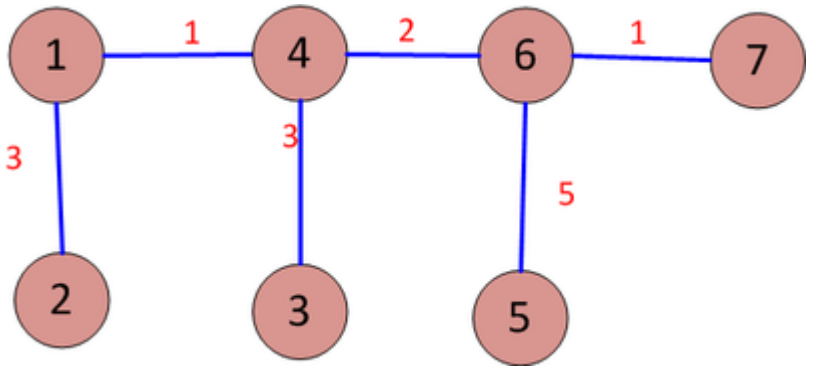
***5-6-5:** Ta nhận thấy cạnh 5-6 không tạo ra một chu trình nào. Vì vậy, thêm 5-6 vào tập hợp*



(https://lhchuong.files.wordpress.com/2014/10/81.png)

Hình 9

CÂY BAO TRÙM THU ĐƯỢC, *Hình 10*:



(https://lhchuong.files.wordpress.com/2014/10/9.png)

Hình 10

Nguồn từ: http://vi.wikipedia.org/wiki/Thu%E1%BA%ADt_to%C3%A1n_Kruskal

2. Cài đặt:

– ***Bước 1***, khai báo các biến, ta sẽ không sử dụng ma trận kề để lưu đồ thị, mà sẽ sử dụng danh sách các cạnh đã khai báo để lưu từng cạnh, vì vậy khi nhập dữ liệu vào, ta cần gán dữ liệu vào danh sách các cạnh:

```
//khai báo các biến cần thiết cho thuật toán Kruskal
#define max 100
int V,nEdge,Mat[max][max],MST[max][max],parent[max];
//cấu trúc 1 cạnh
struct Edge
{
    int u,v,weight;
};
//khai báo mảng các cạnh
Edge edge[max];
```

(https://lhchuong.files.wordpress.com/2014/10/untitled5.png)

– ***Bước 2***, cài đặt thuật toán Kruskal:

```
void Kruskal()
{
    //khai báo các nút cha
    for(int i = 0; i < V; i++)
        parent[i] = -1;

    //sắp xếp các cạnh theo chiều tăng dần của trọng số
    Sort();

    //duyệt tất cả các cạnh
    for(int i = 0; i < nEdge; i++)
    {
        //u và v là 2 đỉnh tạo nên cạnh[i]

        //tìm nút gốc của đỉnh U
        int u = Find_Set(edge[i].u);
        //tìm nút gốc của đỉnh V
        int v = Find_Set(edge[i].v);

        //nếu u giống v thì u và v có cùng đỉnh gốc
        //vì vậy cần kiểm tra u và v không cùng 1 gốc
        if (u != v)
        {
            int x = edge[i].u;
            int y = edge[i].v;
            //lưu lại cây bao trùm
            MST[x][y] = MST[y][x] = edge[i].weight;
            //kết nối đỉnh u và đỉnh v
            Union(u,v);
        }
    }
}
```

(https://lhchuong.files.wordpress.com/2014/10/untitled6.png)

– Hàm thực hiện sắp xếp các cạnh tăng dần theo trọng số:

```
//sắp xếp các cạnh tăng dần theo trọng số
void Sort()
{
    for(int i = 0; i < nEdge; i++)
        for(int j = nEdge - 1; j > i; j--)
            if(edge[j - 1].weight > edge[j].weight)
            {
                Edge hold = edge[j - 1];
                edge[j - 1] = edge[j];
                edge[j] = hold;
            }
}
```

(https://lhchuong.files.wordpress.com/2014/10/untitled7.png)

– Hàm tìm nút cha của một đỉnh:

```
//hàm tìm nút cha
int Find_Set(int x)
{
    while(parent[x] != -1)
        x = parent[x];
    return x;
}
```

(https://lhchuong.files.wordpress.com/2014/10/untitled9.png) – Hàm thực hiện việc kết nối 2 nút lại với nhau:

```
//hàm kết nối 2 nút lại với nhau
void Union(int u,int v)
{
    if(parent[u] > parent[v])
    {
        parent[v] += parent[u];
        parent[u] = v;
    }
    else
    {
        parent[u] += parent[v];
        parent[v] = u;
    }
}
```

(https://lhchuong.files.wordpress.com/2014/10/untitled10.png)

Như vậy là mình vừa hướng dẫn các bạn thuật toán Kruskal – tìm cây khung nhỏ nhất, chúc các bạn cài đặt thành công ^_^

Posted by Chuong Le Hoang in Data structures & Algorithms

4 thoughts on “Thuật toán Kruskal – Tìm cây bao trùm nhỏ nhất”

1. **Châu** *nói:*
   Tháng Tư 1, 2016 lúc 9:32 sáng
   Cám ơn bạn! Viết rất dễ hiểu

   Phản hồi
2. **Châu** *nói:*
   Tháng Tư 4, 2016 lúc 10:24 chiều

Góp ý: ở phần Find_set theo mình là bạn phải viết
while (parent[x]>-1)
x=parent[x]
mới đúng. vì viết như bạn chương trình sẽ lặp vô hạn. Vì có những nút gốc nó sau vài lần nối thì gốc có thể giảm xuống -2, hoặc -3…

Phản hồi

3. **Trường Giang** *nói:*
Tháng Chín 23, 2016 lúc 3:39 chiều
Ví dụ minh họa rất chi tiết và dễ hiểu. Bài viết rất tuyệt!

Phản hồi

4. **Nguyễn Ngọc Anh** *nói:*
Tháng Tám 7, 2017 lúc 12:54 sáng

```
void KRUSKAL(int a[][MAX], int n){
int b[100];
int c[100];
int d[100];
int m = 0, s = 0;
for (int i = 1; i <= n; i++){
for (int j = i; j <= n; j++){
if (a[i][j] == a[j][i] && a[i][j] != MAX){
b[m] = i;
c[m] = j;
d[m] = a[i][j];
m++;
}
}
}
HeapSort(b, c, d, 0, m);
int father[MAX];
for (int i = 1; i <= m; i++)
father[i] = -1;
int i = 1, dem = 0;
while (i <= m && dem -1)
u = father[u];
int v = c[i];
while (father[v] > -1)
v = father[v];
if (u != v){
cout << b[i] << " " << c[i] < father[v]) {
father[v] += father[u];
father[u] = v;
}
else {
father[u] += father[v];
father[v] = u;
}
}
```

```
i++;
}
cout << "Do dai ngan nhat:" << s << endl;
}
```

Phản hồi