Chuong Le Hoang
Open University

Tháng Mười 2, 2014

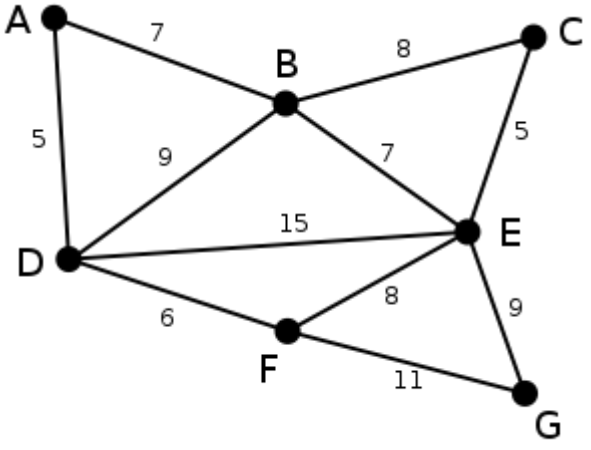
Thuật toán Prim – Tìm cây bao trùm nhỏ nhất

4 phản hồi

1. Mô tả:

Giống như Kruskal, Prim cũng tìm cây khung nhỏ nhất, tuy nhiên Prim phụ thuộc vào đỉnh xuất phát của quá trình tìm kiếm.

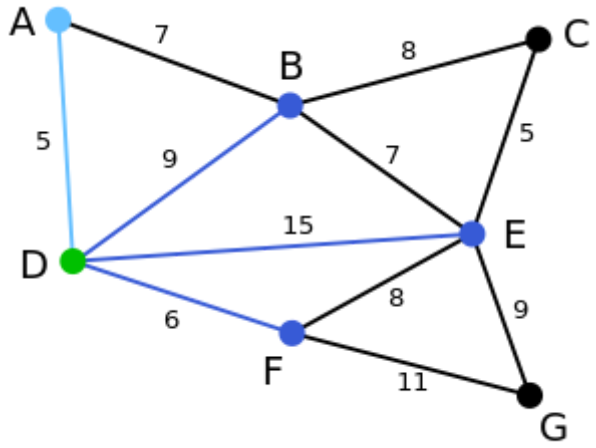
– ***Hình 1***: Đây là đồ thị có trọng số ban đầu. Các số là các trọng số của các cạnh.



(https://lhchuong.files.wordpress.com/2014/10/1.png)

Hình 1

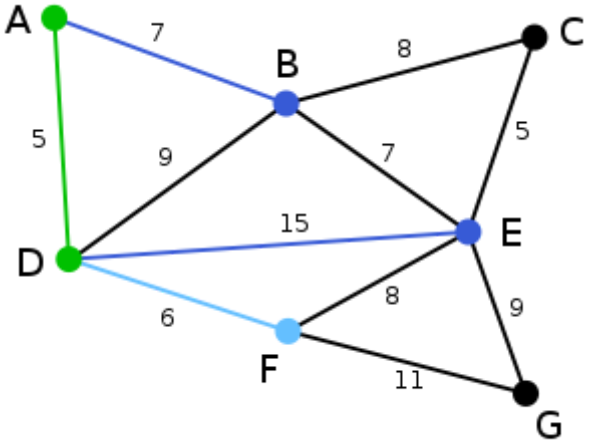
– ***Hình 2***: Chọn một cách tùy ý đỉnh **D** là đỉnh bắt đầu. Các đỉnh **A**, **B**, **E** và **F** đều được nối trực tiếp tới **D** bằng cạnh của đồ thị. **A** là đỉnh gần **D** nhất nên ta chọn **A** là đỉnh thứ hai của cây và thêm cạnh **AD** vào cây.



(https://lhchuong.files.wordpress.com/2014/10/21.png)

Hình 2

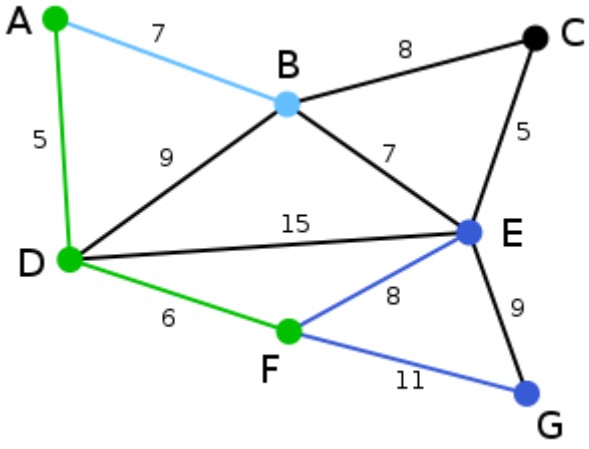
– ***Hình 3***: Đỉnh được chọn tiếp theo là đỉnh gần **D** hoặc **A** nhất. **B** có khoảng cách tới **D** bằng 9 và tới **A**bằng 7, **E** có khoảng cách tới cây hiện tại bằng 15, và **F** có khoảng cách bằng 6. **F** là đỉnh gần cây hiện tại nhất nên chọn đỉnh **F** và cạnh **DF**.



(https://lhchuong.files.wordpress.com/2014/10/31.png)

Hình 3

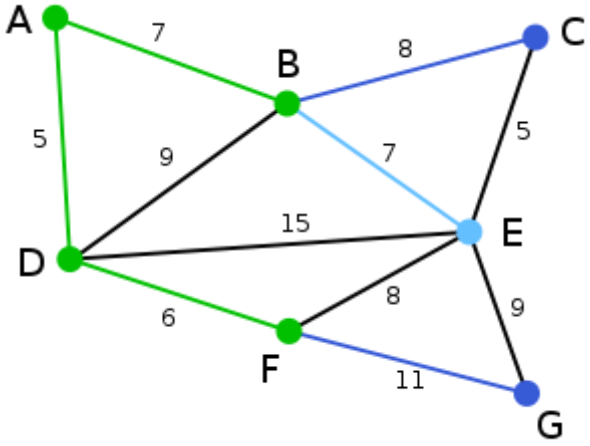
– ***Hình 4***: Thuật toán tiếp tục tương tự như bước trước. Chọn đỉnh **B** có khoảng cách tới **A** bằng 7.



(https://lhchuong.files.wordpress.com/2014/10/4.png)

Hình 4

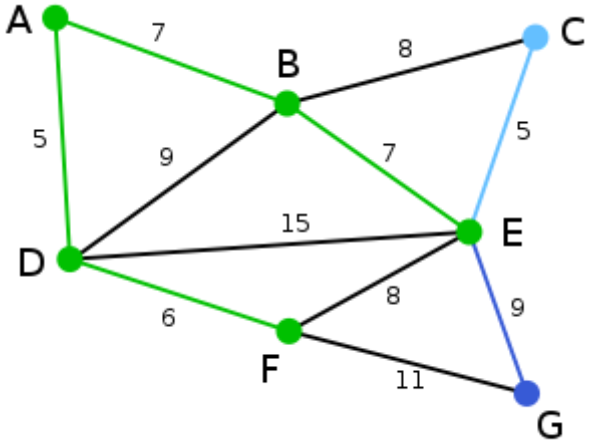
– ***Hình 5***: Ở bước này ta chọn giữa **C**, **E**, và **G**. **C** có khoảng cách tới **B** bằng 8, **E** có khoảng cách tới **B** bằng 7, và **G** có khoảng cách tới **F** bằng 11. **E** là đỉnh gần nhất, nên chọn đỉnh **E** và cạnh **BE**.



(https://lhchuong.files.wordpress.com/2014/10/5.png)

Hình 5

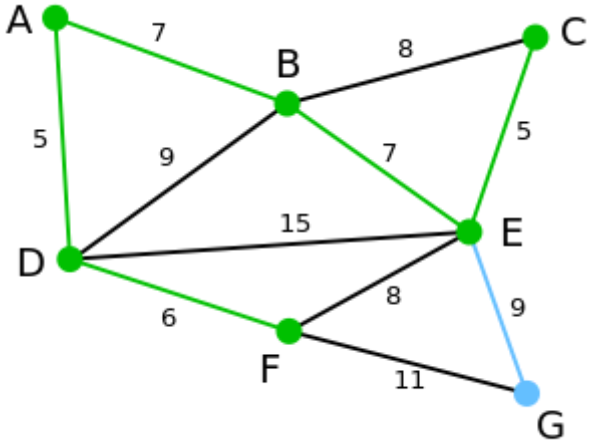
– ***Hình 6***: Ở bước này ta chọn giữa **C** và **G**. **C** có khoảng cách tới **E** bằng 5, và **G** có khoảng cách tới **E**bằng 9. Chọn **C** và cạnh **EC**.



(https://lhchuong.files.wordpress.com/2014/10/6.png)

Hình 6

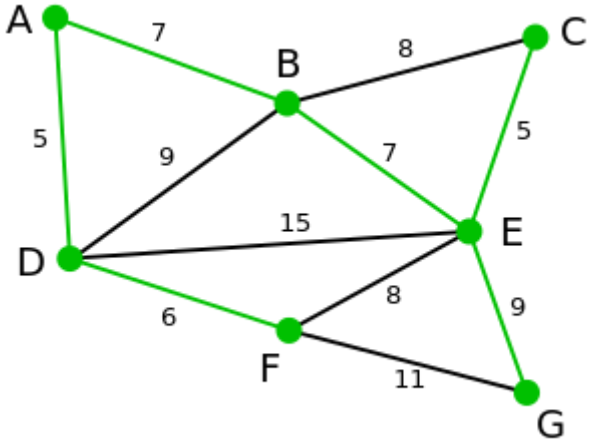
– ***Hình 7***: Quá trình tìm đỉnh kế tiếp không được tạo ra một chu trình so với ban đầu, để biết được đỉnh mới được chọn có trở thành chu trình hay không, ta cần kiểm tra đỉnh đó đã được đi qua hay chưa, nếu đã đi qua thì không thể đi tới, trong trường hợp này là đỉnh B và đỉnh F đều có trọng số mới là 8 (nhỏ nhất) nhưng ta không chọn. Như vậy, đỉnh **G** là đỉnh còn lại duy nhất. Nó có khoảng cách tới **F** bằng 11, và khoảng cách tới **E** bằng 9. **E** ở gần hơn nên chọn đỉnh **G** và cạnh **EG**.



(https://lhchuong.files.wordpress.com/2014/10/7.png)

Hình 7

– ***Hình 8***: Hiện giờ tất cả các đỉnh đã nằm trong cây và cây bao trùm nhỏ nhất được tô màu xanh lá cây. Tổng trọng số của cây là 39.



(https://lhchuong.files.wordpress.com/2014/10/8.png)

Hình 8

Nguồn từ: http://vi.wikipedia.org/wiki/Thu%E1%BA%ADt_to%C3%A1n_Prim

2. Cài đặt:

– **Bước 1**: Khởi tạo các biến:

```
#define max 100
int V,//số đỉnh
    Mat[max][max],//lưu đồ thị bằng ma trận
    MST[max][max];//đồ thị chứa cây bào trùm nhỏ nhất
//khởi tạo các biến cần thiết cho thuật toán Prim
int parent[max],key[max];
bool Free[max];
```

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

– **Bước 2**: Cài đặt các bước thực hiện trong thuật toán Prim:

```
//ta sẽ bắt đầu tại đỉnh D, ở vị trí thứ 4
//nhưng chỉ số trong mảng bắt đầu từ 0 nên sẽ giảm 1 đơn vị
// => start = 3
void Prim(int start)
{
    //khởi tạo các biến
    //parent: nút cha của nút hiện tại
    //free: nút đó đã được thăm hay chưa
    //key: giá trị
    for(int i = 0; i < V; i++)
    {
        parent[i] = -1;
        Free[i] = true;
        key[i] = INT_MAX;
    }

    //bắt đầu quá trình tìm kiếm
    key[start] = 0;

    for(int i = 0; i < V - 1; i++)
    {
        //tìm đỉnh tiếp theo từ tất cả các đỉnh đã tìm được
        //sao cho đỉnh mới tìm được phải là cạnh nhỏ nhất
        int u = Extract_Min();
        //thông báo đã tìm được
        Free[u] = false;
        //duyệt hết tất cả các nút
        for(int v = 0; v < V; v++)
            //nếu nút đó chưa được tìm thấy
            if(Free[v] && Mat[u][v] != 0 && Mat[u][v] < key[v])
            {
                //lưu lại nút cha và trọng số mới
                key[v] = Mat[u][v];
                parent[v] = u;
            }
    }
    //MST là cây vừa tìm được
    //dựa vào nút parent để truy vết và tìm ra cây bao trùm nhỏ nhất
    for(int i = 0; i < V; i++)
        MST[parent[i]][i] = MST[i][parent[i]] = Mat[parent[i]][i];
}
```

(https://lhchuong.files.wordpress.com/2014/10/untitled1.png)**Hàm tìm nút kế tiếp:**

```
//hàm tìm nút kế tiếp
//nút này phải nối với 1 trong các đỉnh tìm được
//và tạo thành 1 cạnh nhỏ nhất kế tiếp
int Extract_Min()
{
    int min = INT_MAX,u;
    for(int i = 0; i < V; i++)
        if(Free[i] && key[i] < min)
        {
            min = key[i];
            u = i;
        }
    return u;
}
```

(https://lhchuong.files.wordpress.com/2014/10/untitled3.png)

Như vậy là tôi vừa hướng dẫn các bạn cách cài đặt thuật toán Prim – tìm cây bao trùm nhỏ nhất, chúc các bạn thành công ^_^

Posted by Chuong Le Hoang in Data structures & Algorithms

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

1. **Nguyễn Văn Thái** *nói:*
   Tháng Mười 19, 2015 lúc 1:59 chiều
   Bài rất chi tiết. Cám ơn bạn rất nhiều

   Phản hồi
   - **Ặc** *nói:*
     Tháng Tám 23, 2016 lúc 7:35 sáng
     kcj =))

     Phản hồi
2. **Trường Giang** *nói:*
   Tháng Chín 23, 2016 lúc 3:41 chiều
   Bài viết rất tuyệt! Minh họa chi tiết, dễ hiểu, thanks!

   Phản hồi
3. **Dinh Son** *nói:*
   Tháng Một 6, 2017 lúc 8:13 sáng
   bài viết có minh hoạ ví dụ rất dễ hiểu, cảm ơn tác giả.

   Phản hồi