Chuong Le Hoang
Open University

Tháng Mười Một 8, 2013

Thuật toán BFS – Tìm kiếm theo chiều rộng

3 phản hồi

1. Mô tả

– Đây là thuật toán tìm các đỉnh bằng cách duyệt theo chiều rộng.

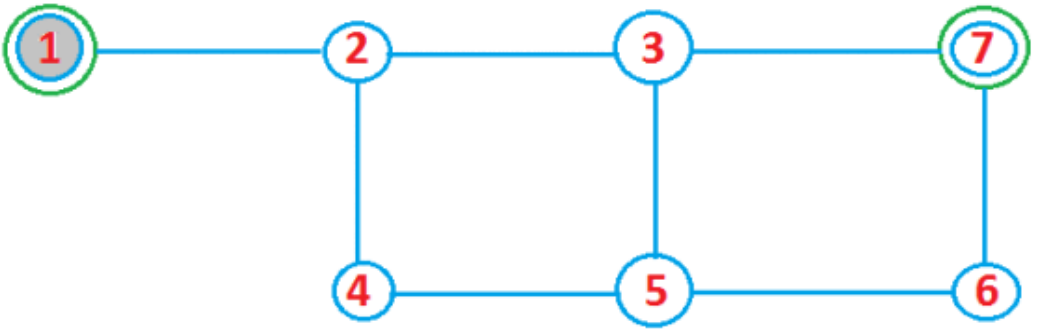
– Xuất phát từ 1 đỉnh và đi tới các đỉnh kề nó, tiếp tục cho đến khi không còn đỉnh nào có thể đi.

– Trong quá trình đi đến đỉnh kề, tiến hành lưu lại đỉnh cha của đỉnh kề để khi đi ngược lại từ đỉnh Kết thúc đến đỉnh Xuất phát, ta có được đường đi ngắn nhất.

– Sở dĩ thuật toán này tìm được đường đi ngắn nhất là nhờ vào cơ chế tô màu và lưu đỉnh cha. Quá trình tô màu khiến 1 đỉnh không thể xét 2 lần trở lên và có thể xem được đường đi từ đỉnh Kết Thúc đến đỉnh Xuất phát dựa vào việc lưu đỉnh cha.

– Sau đây là minh họa về thuật toán:

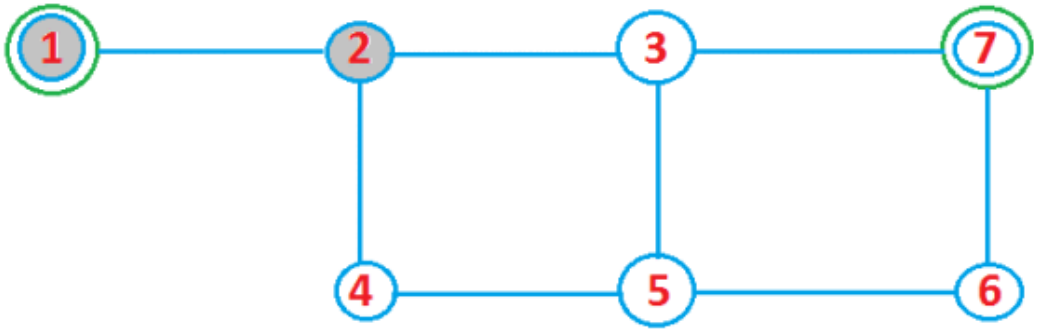
+ ***Hình 1*** : Xuất phát từ đỉnh 1



(https://lhchuong.files.wordpress.com/2013/11/untitled.png)

Hình 1

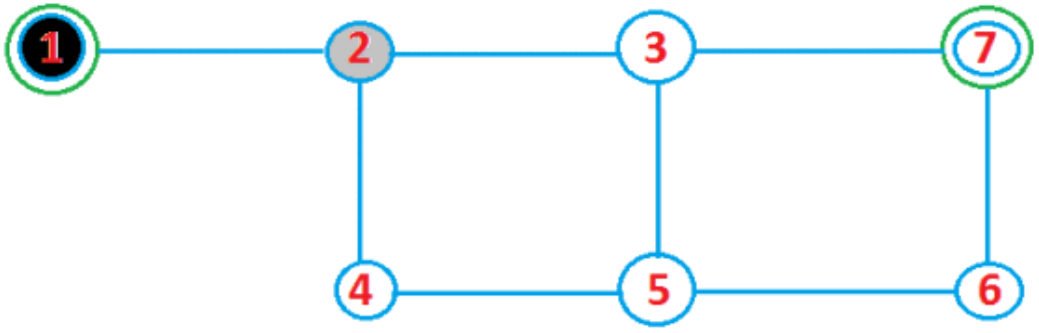
+ ***Hình 2*** : Đi đến đỉnh 2, như vậy nút 1 là nút cha của nút 2



(https://lhchuong.files.wordpress.com/2013/11/untitled1.png)

Hình 2

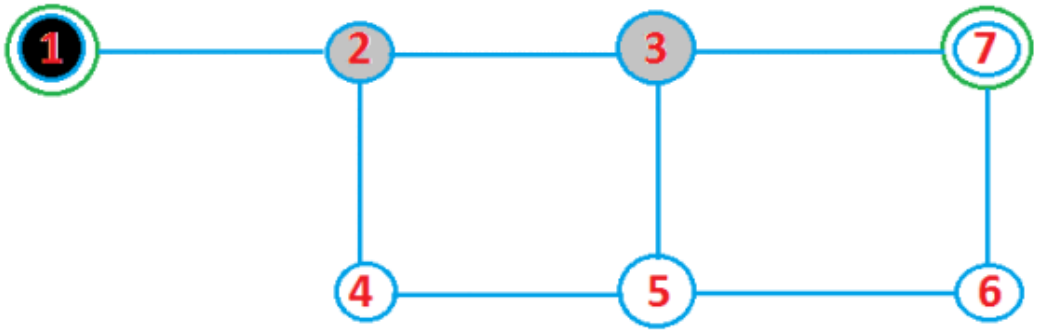
+ ***Hình 3*** : Đã đi hết tất cả các đỉnh kề của đỉnh 1, tiến hành bôi đen đỉnh 1



(https://lhchuong.files.wordpress.com/2013/11/untitled2.png)

Hình 3

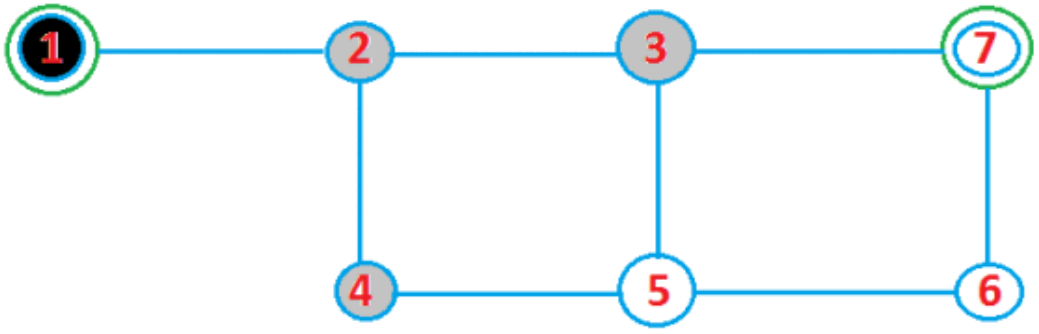
+ ***Hình 4***: Xuất phát từ đỉnh 2, chọn đỉnh 3, nút cha của đỉnh 3 là đỉnh 2



(https://lhchuong.files.wordpress.com/2013/11/untitled3.png)

Hình 4

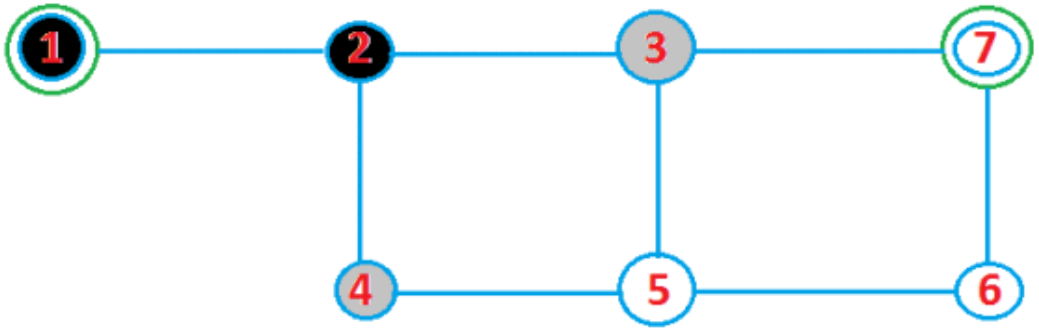
+ ***Hình 5*** : Xuất phát từ đỉnh 2, bôi đen đỉnh 4, nút cha của đỉnh 4 là đỉnh 2



(https://lhchuong.files.wordpress.com/2013/11/untitled4.png)

Hình 5

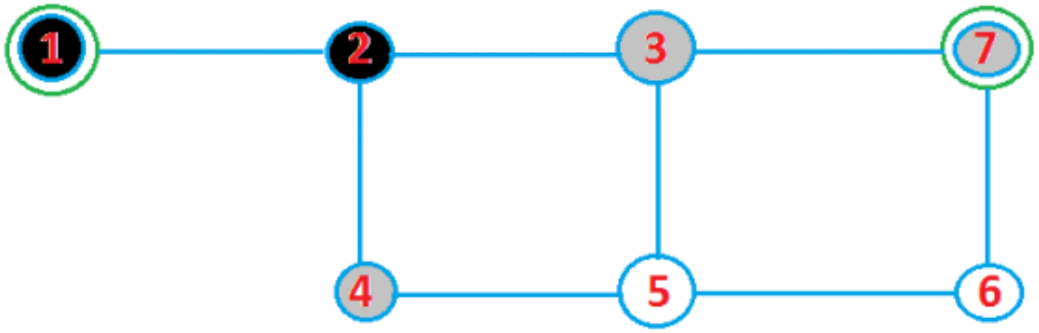
+ ***Hình 6*** : Đã đi hết tất cả các đỉnh kề của đỉnh 2, tiến hành bôi đen đỉnh 2



(https://lhchuong.files.wordpress.com/2013/11/untitled5.png)

Hình 6

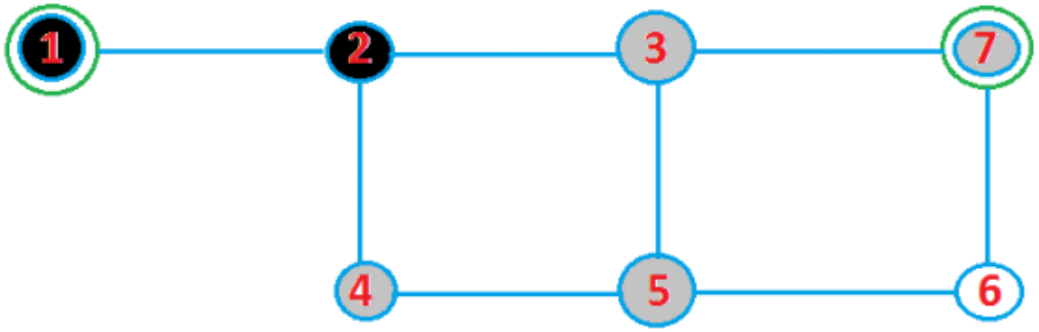
+ ***Hình 7*** : Xuất phát tử đỉnh 3, đi đến đỉnh 7, như vậy đỉnh 3 là đỉnh cha của đỉnh 7



(https://lhchuong.files.wordpress.com/2013/11/untitled7.png)

Hình 7

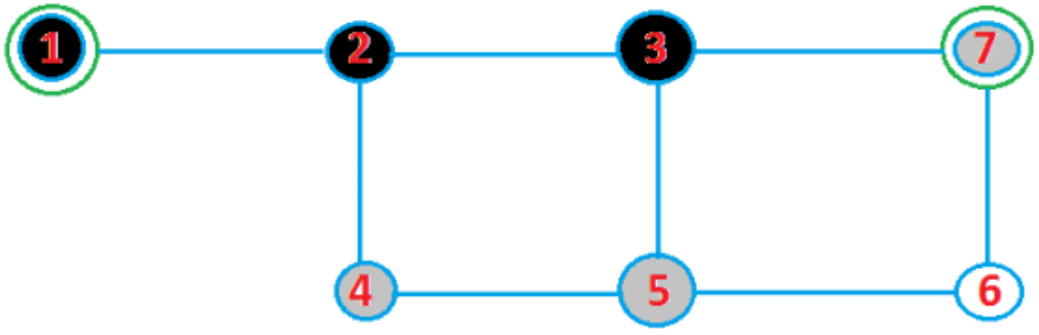
+ ***Hình 8*** : Xuất phát từ đỉnh 3, đi đến đỉnh 5, như vậy đỉnh 3 là đỉnh cha của đỉnh 5



(https://lhchuong.files.wordpress.com/2013/11/untitled8.png)

Hình 8

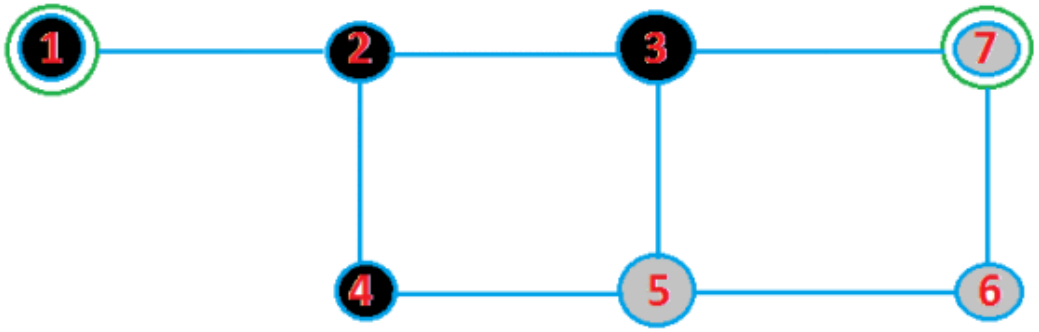
+ ***Hình 3*** : Đã đi hết tất cả các đỉnh kề của đỉnh 3, tiến hành bôi đen đỉnh 3



(https://lhchuong.files.wordpress.com/2013/11/untitled9.png)

Hình 9

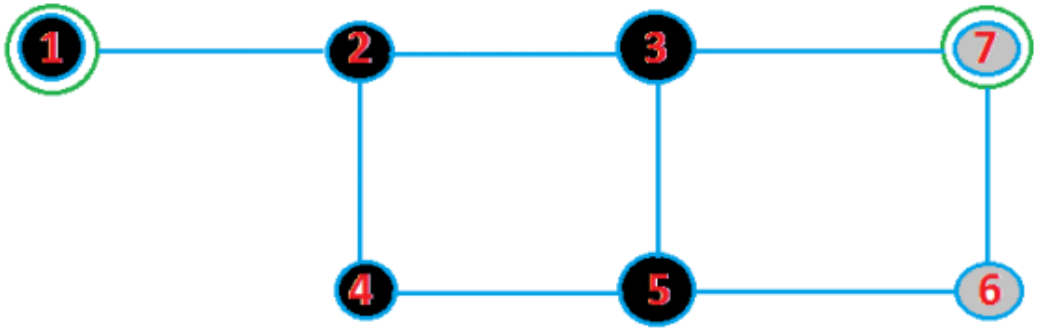
+ ***Hình 10*** : Xuất phát từ đỉnh 5, đi đến đỉnh 6, như vậy đỉnh 5 là đỉnh cha của đỉnh 6



(https://lhchuong.files.wordpress.com/2013/11/untitled11.png)

Hình 10

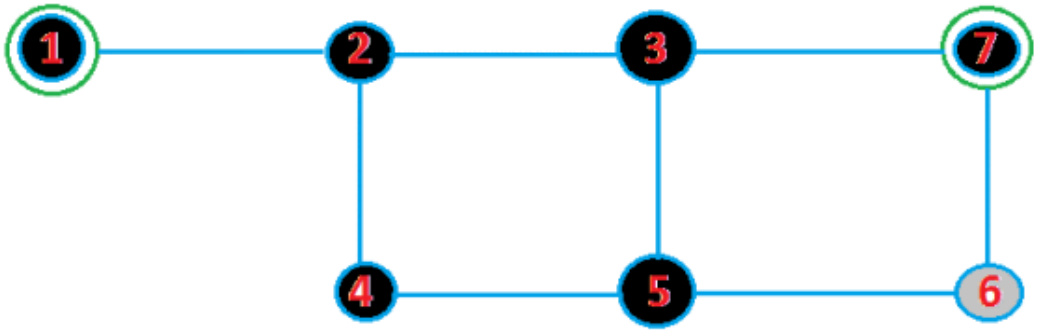
+ ***Hình 11***: Đã đi hết tất cả các đỉnh kề của đỉnh 5, tiến hành bôi đen đỉnh 5



(https://lhchuong.files.wordpress.com/2013/11/untitled12.png)

Hình 11

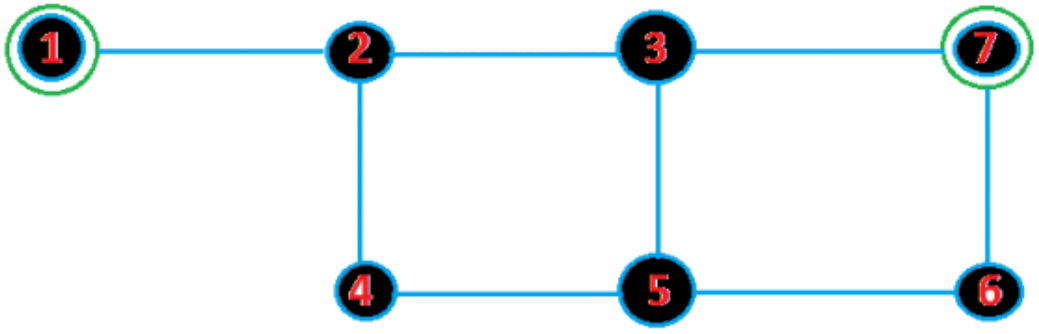
+ ***Hình 12***: Đã đi hết tất cả các đỉnh kề của đỉnh 7, tiến hành bôi đen đỉnh 7



(https://lhchuong.files.wordpress.com/2013/11/untitled13.png)

Hình 12

+ ***Hình 13*** : Đã đi hết tất cả các đỉnh kề của đỉnh6, tiến hành bôi đen đỉnh 6



(https://lhchuong.files.wordpress.com/2013/11/untitled14.png)

Hình 13

– Như vậy ta vừa đi hết tất cả các đỉnh trong đồ thị, và mỗi lần đi đến đỉnh mới, ta đều lưu lại nút cha của đỉnh mới, dựa vào những đỉnh cha này, ta có liệt kê đường đi ngắn nhất bằng cách đi ngược từ đỉnh Kết thúc, đến đỉnh cha của đỉnh Kết thúc … rồi đến đỉnh cha của đỉnh tiếp theo … đến đỉnh Bắt đầu.

2. Cài đặt

– **Hình 14**: Cài đặt thuật toán BFS

```
void BFS()
{
    //nhập đỉnh bắt đầu và kết thúc
    int start, finish;
    cin >> start >> finish;

    //khai báo các mảng cần thiết
    int color[MAX], back[MAX];
    queue<int> qList;

    //khởi tạo giá trị cho các phần tử
    for(int i = 0; i < V; i++)
    {
        //color = 0 <=> chưa đi qua lần nào
        //color = 1 <=> đã đi qua 1 lần
        //color = 2 <=> tất cả đỉnh kề đã được đánh dấu
        color[i] = 0;
        back[i] = -1;//mảng lưu các đỉnh cha của đỉnh i
    }

    //bắt đầu tại đỉnh start
    color[start] = 1;
    //thêm vào hàng đợi
    qList.push(start);
    while(!qList.empty())//nếu hàng đợi k rỗng
    {
        //lấy giá trị phần tử đầu tiên trong hàng đợi
        int u = qList.front();
        //loại phần tử đầu tiên ra khỏi hàng đợi
        qList.pop();
        //tìm đỉnh kề chưa đi qua lần nào
        for(int v = 0; v < V; v++)
            if(A[u][v] != 0 && color[v] == 0)
            {
                color[u] = 1;
                //lưu lại nút cha của đỉnh v
                back[v] = u;
                //tiếp tục thêm đỉnh v vào hàng đợi
                qList.push(v);
            }
        //đã duyệt hết đỉnh kề của đỉnh u
        color[u] = 2;
    }

    //in đường đi ngắn nhất từ đỉnh start đến đỉnh finish
    Print_Path(start, finish, back);
}
```

(https://lhchuong.files.wordpress.com/2013/11/untitled15.png)

Hình 14

– **Hình 15**: In đường đi từ đỉnh Xuất phát đến đỉnh Kết thúc

```
void Print_Path(int u, int v,int back[])
{
    if(u == v)
        cout << v << " ";
    else
        if(back[v] == -1)
            cout << "Khong co duong di\n";
        else
        {
            //dựa vào nút cha để tìm đường đi ngắn nhất
            Print_Path(u, back[v], back);
            cout << v << " ";
        }
}
```

(https://lhchuong.files.wordpress.com/2013/11/untitled16.png)

Hướng phát triển: trong bài này tôi đã không đề cập đến việc tính tổng số đỉnh phải đi từ đỉnh Xuất phát đến đỉnh Kết thúc, vì nó không quan trọng, các bạn có thể tìm hiểu thêm và tự cài đặt. Chúc các bạn thành công!

Posted by Chuong Le Hoang in Data structures & Algorithms

3 thoughts on “Thuật toán BFS – Tìm kiếm theo chiều rộng”

1. Pingback: *Thuật toán DFS – Tìm kiếm theo chiều sâu | lhchuong*

2. **Phong** *nói:*
   Tháng Hai 17, 2014 lúc 9:32 sáng
   Thanks Chương ^_^

   Phản hồi
3. **Ruan Yu Ling** *nói:*
   Tháng Mười Hai 1, 2017 lúc 10:10 sáng
   cám ơn b rất nhiều

   Phản hồi