CÁC GIẢI THUẬT ĐỆ QUY TRÊN ĐỒ THỊ CHẠY NHƯ THẾ NÀO ?
Phần 2 – Kiểm tra đồ thị có chu trình & Kiểm tra đồ thị phân đôi
Phạm Nguyên Khang

Trong phần 1, ta đã tìm hiểu các giải thuật đệ quy: *duyệt đồ thị theo chiều sâu (dfs)* và *giải thuật Tarjan*. Trong phần này, ta sẽ tiếp tục với giải thuật: *kiểm tra đồ thị có chứa chu trình không* và *đồ thị đang xét của phải là đồ thị phân đôi (bipartire graph) không*.

1. Giải thuật kiểm tra đồ thị có chu trình

Ý tưởng chỉnh của giải thuật là duyệt đồ thị theo chiều sâu kết hợp với tô màu đỉnh. Mỗi đỉnh sẽ được tô 1 trong 3 màu sau đây tuỳ thuộc vào trạng thái của nó trong quá trình duyệt:

- WHITE (trắng): chưa duyệt
- GRAY (xám): đang duyệt (đã đi đến nó nhưng chưa duyệt các đỉnh kề của nó)
- BLACK (đen): đã duyệt (đã duyệt nó và duyệt tất cả các đỉnh kề của nó)

Ta sẽ bắt đầu với đồ thị CÓ HƯỚNG. Đồ thị VÔ HƯỚNG sẽ được xét trong phần tiếp theo

```
#define WHITE 0
#define GRAY  1
#define BLACK 2

int color[MAX_N];
int cycle;

void visit(Graph* pG, int u) {
     color[u] = GRAY; //Đang duyệt u                (1)
     for (các đỉnh kề v của u) {                    (2)
          if (color[v] == GRAY) {
               cycle = 1;
               return; //đã phát hiện chu trình
          }
          if (color == WHITE)
               visit(pG, v);
     }
     color[u] = BLACK; //Duyệt xong u               (3)
}
```

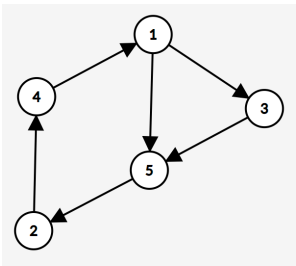
Hàm visit(pG, u) không tính toán và trả về gì cả mà chỉ kiểm tra để tô màu u và gọi đệ quy duyệt các đỉnh kề v của u.

Khi bắt đầu duyệt u, u được tô màu xám. Sau đó xét các đỉnh kề v của u.

- Nếu v có màu xám, có nghĩa là có đường đi u đến v và v lại là kề của v nên đồ thị chứa chu trình v …. -> u -> v. Lúc này ta không cần quan tâm tới các kề khác của u nữa mà return luôn.
- Nếu v có màu trắng, nghĩa là nó chưa duyệt, ta gọi đệ quy duyệt nó.

Khi xét xong các đỉnh kề của u, xem như u đã được duyệt xong, ta tô màu đen cho nó.

Giả sử ta có đồ thị có hướng như sau:



Khởi tạo:

- cycle = 1
- Tất cả các đỉnh đều có màu WHITE.

Hãy chạy từng bước giải thuật visit khi ta gọi visit(&G, 1)

(1) u = 1, tô màu GRAY cho đỉnh 1
(2) Xét các đỉnh kề của 1
v = 3, v có màu WHITE , gọi đệ quy duyệt 3

(1) u = 3, tô màu GRAY cho đỉnh 3
(2) Xét các đỉnh kề của 3
v = 5, có màu WHITE, gọi đệ quy duyệt 5

(1) u = 5, tô màu GRAY cho đỉnh 5
(2) Xét các đỉnh kề của 5
v = 2, có mày WHITE, gọi đệ quy duyệt 2

(1) u = 2, tô màu GRAY cho 2
(2) xét các đỉnh kề của 2
v = 4, có màu WHITE, gọi đệ quy duyệt 4

(1) u = 4, tô màu GRAY cho 4
(2) xét các đỉnh kề của u
v = 1, có màu GRAY
phát hiện chu trình, cycle = 1
return;

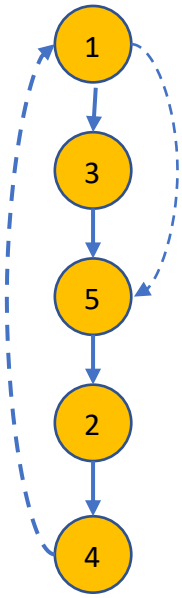
(3) Tô màu BLACK cho 2

(3) Tô màu BLACK cho 5

(3) Tô màu BLACK cho 3

v = 5, có màu BLACK, không làm gì cả
(3) Tô màu BLACK cho 1

Cây duyệt đồ thị:



2. Giải thuật kiểm tra đồ thị phân đôi

Tương tự như giải thuật kiểm tra chu trình, ý tưởng chính của giải thuật là duyệt đồ thị theo chiều sâu kết hợp với tô màu đỉnh. Mỗi đỉnh sẽ được tô 1 trong 3 màu:

- WHITE: chưa duyệt
- BLUE: đã duyệt và được tô màu xanh
- RED: đã duyệt và được tô màu đỏ

Giải thuật này chỉ làm việc với đồ thị VÔ HƯỚNG.

1

```
#define WHITE -1
#define BLUE  0
#define RED   1

int color[MAX_N];
int conflict;

void colorize(Graph* pG, int u, int c) {
  color[u] = c; //Tô màu c cho u                (1)
  for (các đỉnh kề v của u) {                   (2)
    if (color[v] == WHITE)                      (2a)
      colorize(pG, v, !c);
    else if (color[v] == c) {                   (2b)
      conflict = 1; //Đụng độ: 2 đỉnh kề tô cùng màu
      return;
    }
  }
}
```

Một bản cài đặt khác: kiểm tra màu trước khi duyệt u (Tài liệu thực hành sử dụng phương pháp này). Cả 2 phương pháp đều đúng !

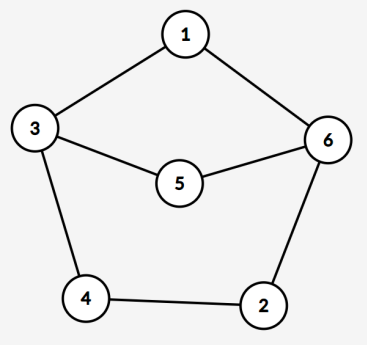
2

```
#define WHITE -1
#define BLUE  0
#define RED   1

int color[MAX_N];
int conflict;

void colorize(Graph* pG, int u, int c) {
  if (color[u] == WHITE) {
    color[u] = c; //Tô màu c cho u              (1)
    for (các đỉnh kề v của u) {                 (2)
      colorize(pG, v, !c);
  } else if (color[u] != c) {
      conflict = 1; //Đụng độ: u bị tô 2 màu khác nhau
      return;
    }
  }
}
```

Cho đồ thị vô hướng G như bên dưới.



Khởi tạo, tất cả các đỉnh có màu trắng và conflict = 0. Sử dụng bản cài đặt 1, hãy chạy từng bước lệnh colorize(&G, 1, BLUE);

(1) u = 1, c = **BLUE**, tô màu BLUE cho đỉnh 1
(2) Xét các đỉnh kề của 1
v = 3, v có màu WHITE , gọi đệ quy duyệt 3 với màu !BLUE = RED

(1) u = 3, c = **RED**, tô màu RED cho đỉnh 3
(2) Xét các đỉnh kề của 3
v = 1, có màu BLUE khác với c, không làm gì cả
v = 4, có màu WHITE, gọi đệ quy duyệt 5 với màu !RED = BLUE

(1) u = 4, c = **BLUE**, tô màu BLUE cho đỉnh 4
(2) Xét các đỉnh kề của 4
v = 3, có màu RED khác với c, không làm gì cả
v = 2, có màu WHITE, gọi đệ quy duyệt 2 với màu !BLUE = RED

(1) u = 2, c = **RED**, tô màu RED cho 2
(2) xét các đỉnh kề của 2
v = 4, có màu BLUE khác với c, không làm gì cả
v = 6, có màu WHITE, gọi đệ quy duyệt 6 với màu !RED = BLUE

(1) u = 6, c = **BLUE**, tô màu BLUE cho 6
(2) xét các đỉnh kề của 6
v = 1, có màu BLUE trùng với màu của c
Đụng độ, conflict = 1
return; **//bỏ qua các kề còn lại của 6: 2 & 5**

v = 5, có màu WHITE, gọi đệ quy duyệt 5 với màu !RED = BLUE

(1) u = 5, c = **BLUE**, tô màu BLUE cho 5
(2) xét các đỉnh kề của 5
v = 3, có màu RED khác với c, không làm gì cả
v = 6, có màu BLUE trùng với c
Đụng độ, conflict = 1
return;

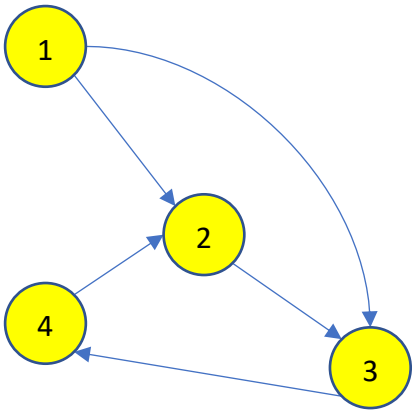
v = 6, có màu BLUE trùng với c
Đụng độ, conflict = 1;

Trong quá trình chạy có đến 3 lần bị đụng độ (conflict = 1), vì thế đồ thị này không thể phân đôi. Thật ra chỉ cần 1 lần đụng độ là đồ thị không thể phân đôi được.

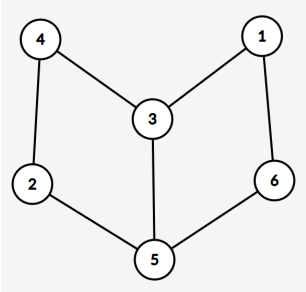
Trong bản cài đặt này (kể cả 1 & 2) thì sau khi bị đụng độ giải thuật chỉ *bỏ qua các đỉnh kề của đỉnh u hiện tại* và vẫn tiếp tục quá trình duyệt các đỉnh khác. Trong ví dụ trên khi gặp đụng độ lần đầu tiên ở đỉnh u = 6 và v = 1, ta bỏ qua kề của 6 là 2 & 5 để quay về 4, sau đó về 3 lại tiếp tục xét đỉnh kề còn lại của 3 là 5 !

3. Bài tập

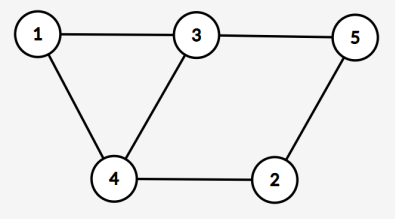
1) Cho đồ thị có hướng như hình vẽ. Hãy chạy thủ công giải thuật kiểm tra chu trình bằng cách duyệt theo sâu bắt đầu từ đỉnh 1 (trình bày kết quả giống ví dụ bên trên). Vẽ cây duyệt đồ thị theo chiều sâu thu được.



2) Cho đồ thị vô hướng như hình vẽ. Hãy chạy thủ công giải thuật kiểm tra tính phân đôi của đồ thị bằng cách tô màu từ đỉnh 1. Vẽ cây duyệt đồ thị theo chiều sâu thu được và màu của các đỉnh tương ứng. Cho biết đồ thị có phân đôi được không ?



3) Cho đồ thị vô hướng như hình vẽ. Hãy chạy thủ công giải thuật kiểm tra tính phân đôi của đồ thị bằng cách tô màu từ đỉnh 1. Vẽ cây duyệt đồ thị theo chiều sâu thu được và màu của các đỉnh tương ứng. Cho biết đồ thị có phân đôi được không ?



4) Hãy điều chỉnh giải thuật visit() sao cho khi phát hiện chu trình ta không cần phải duyệt thêm bất cứ đỉnh nào nữa.
5) Hãy điều chỉnh giải thuật colorize() sao cho khi bị đụng độ ta không cần phải duyệt thêm bất cứ đỉnh nào nữa.