BÁO CÁO THUẬT TOÁN TÔ MÀU ĐỒ THỊ*

Lê Quang Khải†

Ngày 7 tháng 6 năm 2023

Tóm tắt nội dung

Bài viết này trình bày thuật toán tô màu đồ thị.

Mục lục

*Lê Quang Khải
†email: Khai.LQ225638@sis.hust.edu.vn

1 Thuật toán

Bài toán 1. Cho một đồ thị có n đỉnh, m cạnh, tìm một cách tô màu đồ thị.

- Dữ liệu vào: Từ file dothi.txt
  - Dòng đầu nhập vào hai số nguyên n, m. Trong đó n là số đỉnh và m là số cạnh của đồ thị.
  - m dòng tiếp theo, mỗi dòng ghi hai số nguyên ứng với một cạnh của đồ thị.
- Dữ liệu ra: file dothitomau.txt gồm n + 1 dòng, dòng đầu là số màu dùng để tô đồ thị. Với n dòng tiếp theo, dòng thứ i + 1 tương ứng với màu dùng để tô đỉnh i.

Diễn giải thuật toán.
Đầu tiên, ta khai báo thư viện `bits/stdc++.h` và sử dụng `ll` thay thế cho kiểu `unsigned long long` để code được ngắn gọn.

```
#include <bits/stdc++.h>
using namespace std;
using ll = unsigned long long;
```

Đoạn code sau thực hiện chức năng nhập dữ liệu cho đồ thị.

```
freopen("dothi.txt", "r", stdin);
freopen("dothitomau.txt", "w", stdout);
ll n, m;
cin >> n >> m;
vector<vector<ll>> adj(n + 1);
for (ll i = 0; i < m; i++)
{
    ll u, v;
    cin >> u >> v;
    adj[u].push_back(v);
    adj[v].push_back(u);
}
```

Ta khai báo biến `k` là bậc lớn nhất của một đỉnh trong đồ thị

```
ll k = 0;
for (ll i = 1; i <= n; i++)
{
    if (adj[i].size() > k)
    {
        k = adj[i].size();
    }
}
```

Ta sử dụng thuật toán BFS để kiểm tra đồ thị có liên thông hay không

```
bool isConnected = true;
vector<bool> reach(n + 1, false);
queue<ll> que;
reach[1] = true;
que.push(1);
while(!que.empty())
{
    ll u = que.front();
    que.pop();
    for (auto v: adj[u])
    {
        if (!reach[v])
        {
            reach[v] = true;
            que.push(v);
        }
    }
}

for (ll i = 1; i <= n; i++)
{
    if (!reach[i])
    {
        isConnected = false;
        break;
    }
}
```

Sau đó, kiểm tra xem đồ thị có phải đồ thị chính quy (regular graph) hay không. Nếu không phải, `cur` là đỉnh đầu tiên có bậc khác k và đánh dấu `check[cur] = true`

```
vector<bool> check(n + 1, false);
ll cur = 0;
bool isRegular = true;
for (ll i = 1; i <= n; i++)
{
    if (adj[i].size() < k)
    {
        cur = i;
        check[cur] = true;
        isRegular = false;
        break;
    }
}
```

Ta sử dụng thuật toán tham lam để tô màu đồ thị. Thuật toán này cho ta số màu để tô không vượt quá $k+1$ và không vượt quá k nếu đồ thị liên thông, không chính quy. Khai báo mảng `vector<ll> v` ứng với dãy đỉnh mà ta sẽ thực hiện thuật toán. Nếu đồ thị là chính quy hoặc liên thông thì xét dãy đỉnh là dãy từ 1 đến n. Nếu không thì xét một đỉnh bậc bé hơn `k`, ở đây là đỉnh `cur` và điền tất cả các đỉnh kề với `cur` vào dãy. Tiếp tục thực hiện cho đến khi thu được dãy n đỉnh. Ý tưởng ở đây là sử dụng cách chứng minh cho định lý:
Nếu G là một đồ thị liên thông, không chính quy và mọi đỉnh đều có bậc không vượt quá k thì $\chi(G) \leq k$.
Mảng `v` được cài đặt như sau:

```
vector<ll> v;

if (isRegular == true || isConnected == false)
{
    for (ll i = 0; i < n; i++)
    {
        v.push_back(i + 1);
    }
}
else
{
    ll j = 0;
    v.push_back(cur);
    while (v.size() < n)
    {

        ll u = v[j];
        for (ll i = 0; i < adj[u].size(); i++)
        {
            ll id = adj[u][i];
            if (check[id] == false)
            {
                v.push_back(id);
                check[id] = true;
            }
        }
        j++;
    }
}
```

Cuối cùng, ta cài đặt thuật toán tham lam và in kết quả ra màn hình. Ta duyệt lần lượt các đỉnh trong dãy đã cài đặt ở trên, nếu đỉnh `v[i]` chưa được tô thì khai báo mảng `flag` để kiểm tra các màu đã được tô cho các đỉnh kề với `v[i]`. Chọn màu bé nhất chưa được tô để tô cho `v[i]`. Ta lưu số màu cần dùng vào biến `Max` với lưu ý rằng số màu cần dùng chính bằng màu được đánh số lớn nhất để tô đồ thị.

```
vector<ll> color(n + 1); // color[i] la mau cua dinh i
color[v[0]] = 1;
ll Max = 1;
for (ll i = 0; i < n; i++)
{
    if (color[v[i]] == 0)
    {
        vector<bool> flag(n + 1, false);
        for (ll j = 0; j < adj[v[i]].size(); j++)
        {
            ll id = adj[v[i]][j];
            if (color[id] != 0)
            {
                flag[color[id]] = true;
            }
        }
        for (ll j = 1; j <= n; j++)
        {
            if (flag[j] == false)
            {
                color[v[i]] = j;
                if (j > Max)
                {
                    Max = j;
                }
                break;
            }
        }
    }
}
cout << Max << "\n";
for (ll i = 1; i <= n; i++)
{
    cout << color[i] << "\n";
}
```

Đánh giá thuật toán. Thuật toán BFS chạy trong $O(n + m)$. Thuật toán tham lam chạy trong $O(n^2)$. Do đó chương trình chạy với độ phức tạp thời gian là $O(n^2 + m)$.

2 Nguồn tham khảo

[1] Tô màu đỉnh của đồ thị
https://drive.google.com/drive/folders/1IBRntbxtsIpoSn_I3jj7zgzQxgq52L3v
[2] Breadth First Search
https://cp-algorithms.com/graph/breadth-first-search.html#implementation