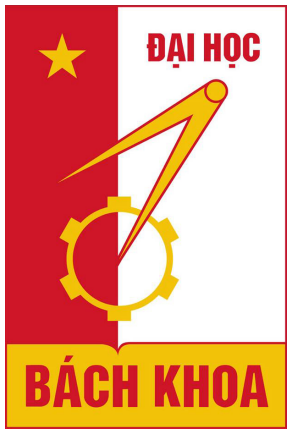
Thuật toán ứng dụng
Bài thực hành số 5.2: Các thuật toán trên đồ thị

TS. Đinh Viết Sang, TA. Đặng Xuân Vương



Trường Đại học Bách khoa Hà Nội
Viện Công nghệ thông tin và Truyền thông

Ngày 20 tháng 12 năm 2020

Mục lục

Mục lục

MỞ ĐẦU

- Đồ thị là một cấu trúc gồm các đỉnh và cạnh, có thể mô hình hoá nhiều cấu trúc thực tế: hệ thống giao thông, mạng máy tính, mạng xã hội,...

MỞ ĐẦU

- Đồ thị là một cấu trúc gồm các đỉnh và cạnh, có thể mô hình hoá nhiều cấu trúc thực tế: hệ thống giao thông, mạng máy tính, mạng xã hội,...
- Vì vậy, nhiều bài toán thực tế được đưa về các bài toán trên đồ thị: tìm đường đi ngắn nhất, tô màu đồ thị,...

MỞ ĐẦU

- Đồ thị là một cấu trúc gồm các đỉnh và cạnh, có thể mô hình hoá nhiều cấu trúc thực tế: hệ thống giao thông, mạng máy tính, mạng xã hội,...
- Vì vậy, nhiều bài toán thực tế được đưa về các bài toán trên đồ thị: tìm đường đi ngắn nhất, tô màu đồ thị,...
- Các thuật toán trên đồ thị: DFS, BFS, Dijkstra, Kruskal,...

Mục lục

06. BUGLIFE

- Cho một đồ thị vô hướng.

06. BUGLIFE

- Cho một đồ thị vô hướng.
- Kiểm tra xem nó có phải là đồ thị hai phía hay không.

Thuật toán

- Dùng hai màu đen và đỏ để tô màu cho đồ thị.

Thuật toán

- Dùng hai màu đen và đỏ để tô màu cho đồ thị.
- Với mỗi thành phần liên thông của đồ thị, chọn một đỉnh bất kỳ và tô màu đỏ.

Thuật toán

- Dùng hai màu đen và đỏ để tô màu cho đồ thị.
- Với mỗi thành phần liên thông của đồ thị, chọn một đỉnh bất kỳ và tô màu đỏ.
- Với mỗi đỉnh đã được tô màu, xét các đỉnh kề với nó:

Thuật toán

- Dùng hai màu đen và đỏ để tô màu cho đồ thị.
- Với mỗi thành phần liên thông của đồ thị, chọn một đỉnh bất kỳ và tô màu đỏ.
- Với mỗi đỉnh đã được tô màu, xét các đỉnh kề với nó:
  - Nếu đỉnh kề chưa được tô màu, ta tô màu ngược lại với đỉnh đang xét.

Thuật toán

- Dùng hai màu đen và đỏ để tô màu cho đồ thị.
- Với mỗi thành phần liên thông của đồ thị, chọn một đỉnh bất kỳ và tô màu đỏ.
- Với mỗi đỉnh đã được tô màu, xét các đỉnh kề với nó:
  - Nếu đỉnh kề chưa được tô màu, ta tô màu ngược lại với đỉnh đang xét.
- Nếu có 2 đỉnh kề bất kỳ được tô cùng màu, trả về thông báo phát hiện bất thường.

Code

```
vector<int> a[N];
int color[N];

void dfs(int u) {
    for (int v : a[u]) {
        if (color[v] == -1) {
            color[v] = !color[u];
            dfs(v);
        }
    }
}
```

Code

```
for (int i = 1; i <= n; ++i) color[i] = -1;
for (int i = 1; i <= n; ++i) {
    if (color[i] == -1) {
        color[i] = 0;
        dfs(i);
    }
}
bool bipartite = true;
for (int u = 1; u <= n; ++u) {
    for (int v : a[u]) {
        bipartite &= color[u] != color[v];
    }
}
```

Mục lục

ICBUS

- Cho n thị trấn được đánh số từ 1 tới n.

ICBUS

- Cho n thị trấn được đánh số từ 1 tới n.
- Có k con đường hai chiều nối giữa các thị trấn.

ICBUS

- Cho n thị trấn được đánh số từ 1 tới n.
- Có k con đường hai chiều nối giữa các thị trấn.
- Ở thị trấn thứ i có thể mua vé với giá là c_i và đi qua tối đa d_i cạnh bất kỳ, xuất phát từ i.

ICBUS

- Cho n thị trấn được đánh số từ 1 tới n.
- Có k con đường hai chiều nối giữa các thị trấn.
- Ở thị trấn thứ i có thể mua vé với giá là c_i và đi qua tối đa d_i cạnh bất kỳ, xuất phát từ i.
- Tìm chi phí tối thiểu để đi từ thị trấn 1 tới thị trấn n.

Thuật toán

- **Bước 1:** Tính khoảng cách di chuyển ngắn nhất của tất cả các cặp đỉnh u, v bằng thuật toán BFS. Lưu vào mảng $dist[u][v]$.

Thuật toán

- **Bước 1:** Tính khoảng cách di chuyển ngắn nhất của tất cả các cặp đỉnh u, v bằng thuật toán BFS. Lưu vào mảng $dist[u][v]$.
- **Bước 2:** Tạo một đồ thị có hướng trong đó tồn tại cung (u, v) khi $dist[u][v] <= d[u]$ và cung này có trọng số là $c[u]$.

Thuật toán

- **Bước 1:** Tính khoảng cách di chuyển ngắn nhất của tất cả các cặp đỉnh u, v bằng thuật toán BFS. Lưu vào mảng $dist[u][v]$.
- **Bước 2:** Tạo một đồ thị có hướng trong đó tồn tại cung (u, v) khi $dist[u][v] <= d[u]$ và cung này có trọng số là $c[u]$.
- **Bước 3:** Tìm đường đi ngắn nhất từ 1 tới n trên đồ thị mới được tạo ra bằng thuật toán Dijkstra.

Thuật toán

- **Bước 1:** Tính khoảng cách di chuyển ngắn nhất của tất cả các cặp đỉnh u, v bằng thuật toán BFS. Lưu vào mảng $dist[u][v]$.
- **Bước 2:** Tạo một đồ thị có hướng trong đó tồn tại cung (u, v) khi $dist[u][v] <= d[u]$ và cung này có trọng số là $c[u]$.
- **Bước 3:** Tìm đường đi ngắn nhất từ 1 tới n trên đồ thị mới được tạo ra bằng thuật toán Dijkstra.
- Độ phức tạp thuật toán $O(n^2)$

Code

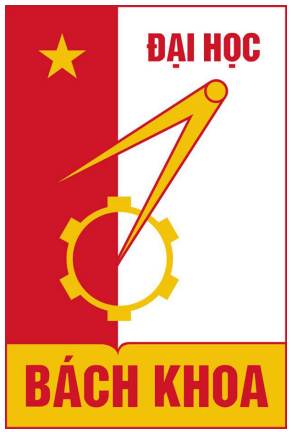
```
void calculate_dist() {
    ** Calculate dist[u][v] using BFS algorithm **
}
void find_shortest_path() {
    for (int i = 0; i <= n; i++) {
        ans[i] = MAX;
        visit[i] = 0;
    }
    ans[1] = 0;
    int step = n;
    while (step--) {
        int min_vertex = 0;
        for (int i = 1; i <= n; i++) {
            if (visit[i] == 0 &&
            ans[min_vertex] > ans[i]) {
                min_vertex = i;
            }
        }
```

Code

```
            visit[min_vertex] = 1;
            for (int i = 1; i <= n; i++) {
                if (dist[min_vertex][i]
                <= d[min_vertex]) {
                    ans[i] = min(ans[i],
                    ans[min_vertex] + c[min_vertex]);
                }
            }
        }
        cout << ans[n] << endl;
}
```

Thuật toán ứng dụng
Bài thực hành số 5.2: Các thuật toán trên đồ thị

TS. Đinh Viết Sang, TA. Đặng Xuân Vương



Trường Đại học Bách khoa Hà Nội
Viện Công nghệ thông tin và Truyền thông

Ngày 20 tháng 12 năm 2020