Bài 11.3: Thuật toán BFS

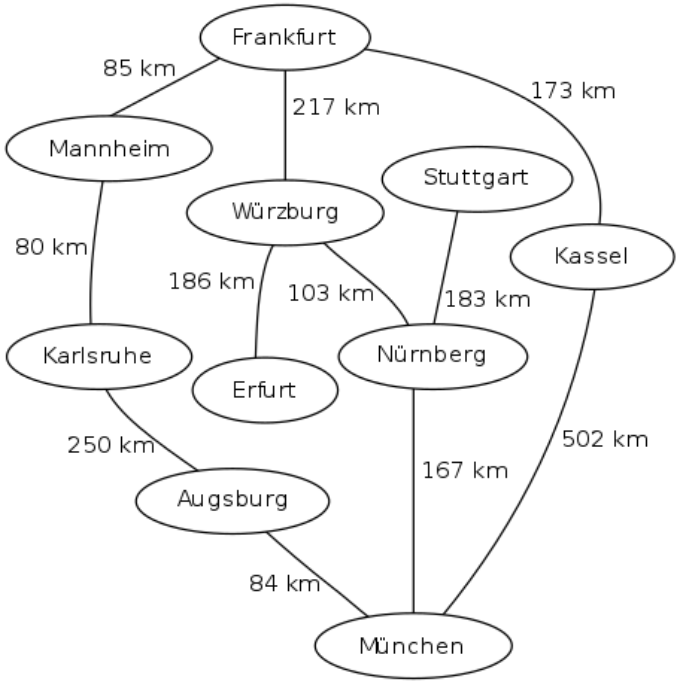
- ✓ Khái niệm và đặc điểm
- ✓ Mục đích sử dụng
- ✓ Mã giả và triển khai
- ✓ Ví dụ minh họa
- ✓ Bài tập thực hành

Khái niệm và đặc điểm

- BFS viết tắt của Breath first search. Là thuật toán tìm kiếm theo chiều rộng.
- Dùng để tìm kiếm một node thỏa mãn tiêu chí tìm kiếm nào đó trên cấu trúc dữ liệu cây hoặc đồ thị.
- Thuật toán bắt đầu từ node gốc sau đó mở rộng tất cả các node cùng độ sâu trước khi chuyển tới các node ở độ sâu kế tiếp.
- Thuật toán BFS sử dụng hàng đợi để lưu vết các node con đã được tìm thấy nhưng chưa được duyệt(thăm).
- Độ phức tạp của thuật toán là O(|V| + |E|) trong đó V và E lần lượt là số đỉnh và cạnh của đồ thị.

Mục đích sử dụng

- Ứng dụng vào giải quyết nhiều vấn đề của lý thuyết đồ thị:
- Tìm tất cả các đỉnh trong một thành phần liên thông.
- Tìm đường đi ngắn nhất giữa hai đỉnh u, v. của đồ thị không trọng số.



Mục đích sử dụng

- Tìm cây bao trùm nhỏ nhất của đồ thị không trọng số.
- Tìm các node lân cận trong mạng ngang hàng như BitTorrent.
- Trong các trang mạng xã hội, sử dụng BFS để tìm các người dùng ở khoảng cách k so với một người được chỉ định.
- Thuật toán Cheny cho trình dọn rác.
- Trong hệ thống GPS, BFS sử dụng để tìm các điểm lân cận.
- Kiểm tra có tồn tại đường đi giữa hai đỉnh cho trước không.
- Kiểm tra xem một đồ thị có phải đồ thị hai phía không.
- Thuật toán Ford-Fulkerson để tìm luồng cực đại trong mạng.

Mã giả

- Sau đây là mã giả thuật toán BFS:

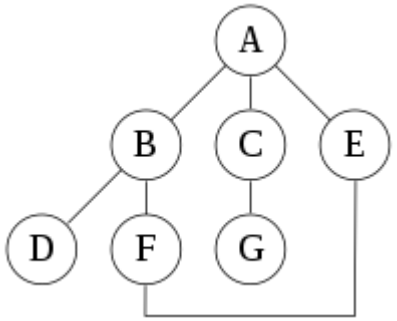
```
// thuật toán duyệt theo chiều rộng dùng vòng lặp
// vertices: tập đỉnh của đồ thị
// adjMatrix: ma trận kề của đồ thị
// root: đỉnh gốc của đồ thị
// output: đường đi ngắn nhất giữa đỉnh đích và đỉnh root
function bfs(vertices[], adjMatrix[], size, root):
    tạo queue q
    đẩy đỉnh root vào queue q
    đánh dấu đỉnh root đã được duyệt
    while(q chưa rỗng):
        v = lấy phần tử đầu hàng đợi
        if(đỉnh v là đỉnh đích):
            return v
        for(tất cả các đỉnh u kề với v):
            if(u chưa được duyệt):
                đánh dấu u đã được duyệt
                thêm u vào hàng đợi
```

Triển khai

➢ Triển khai thuật toán BFS:

```
// thuật toán bfs
void bfs(Vertex* vertices, bool** adjMatrix, int size, int index) {
    queue<int> mQueue;
    vertices[index].visited = true;
    mQueue.push(index);
    while (!mQueue.empty()) {
        index = mQueue.front();
        mQueue.pop();
        displayVertex(vertices[index]);
        for (int i = 0; i < size; i++)
        {
            if (adjMatrix[index][i] && !vertices[i].visited) {
                vertices[i].visited = true;
                mQueue.push(i);
            }
        }
    }
}
```

Ví dụ



Queue	Kết quả hiển thị ra màn hình
A	Rỗng
B C E	A
C E D F	A B
E D F G	A B C
D F G	A B C E
F G	A B C E D
G	A B C E D F
Rỗng	A B C E D F G

- Thứ tự duyệt BFS: A -> B -> C -> E -> D -> F -> G.
- Tìm đường đi ngắn nhất đến đỉnh G: A -> C -> G.

Nội dung tiếp theo

Thuật toán Dijkstra tìm đường đi ngắn nhất