Bài 11.2: Thuật toán DFS

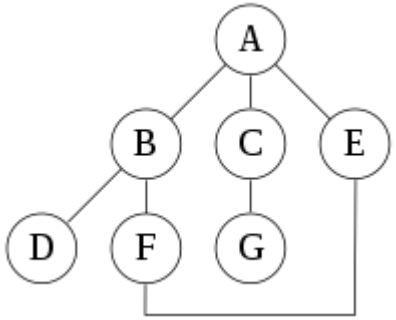
- ✓ Khái niệm và đặc điểm
- ✓ Mục đích sử dụng
- ✓ Mã giả và triển khai
- ✓ Ví dụ minh họa
- ✓ Bài tập thực hành

Khái niệm và đặc điểm

- Thuật toán tìm kiếm(duyệt) theo chiều sâu tên tiếng Anh là Depth first search(DFS).
- Là một thuật toán tìm kiếm hoặc duyệt một cây hoặc đồ thị.
- Thuật toán bắt đầu từ node gốc hoặc node được chọn và tiến sâu nhất có thể theo mỗi nhánh.
- Đây là một thuật toán tìm kiếm không đầy đủ. Quá trình tìm kiếm phát triển đến đỉnh con đầu tiên của node đang tìm kiếm cho tới khi gặp node cần tìm. Hoặc tới khi gặp một node không có node con.
- Khi đó giải thuật quay lui về node vừa tìm kiếm ở bước trước.
- Trong dạng không đệ quy ta sử dụng ngăn xếp hỗ trợ duyệt các đỉnh.

Khái niệm và đặc điểm

- Độ phức tạp không gian của DFS nhỏ hơn BFS(duyệt theo chiều rộng).
- Độ phức tạp thời gian của hai thuật toán là tương đương nhau bằng O(|V| + |E|).



- Kết quả duyệt: A->B->D->F->E->C->G.

Kết quả của thuật toán

- Tạo ra một cây bao phủ lên tất cả các đỉnh được duyệt của đồ thị.
- Có thể duyệt theo 3 cách để tạo ra một danh sách tuyến tính các đỉnh của đồ thị:
  - Duyệt tiền thứ tự: tạo ra danh sách mà các đỉnh xuất hiện theo đúng trật tự nó được duyệt(thăm).
  - Duyệt hậu thứ tự: tạo ra danh sách mà các đỉnh xuất hiện theo thứ tự của lần duyệt đến sau cùng.
  - Duyệt đảo hậu thứ tự: đảo ngược kết quả trong duyệt hậu thứ tự.

Mục đích sử dụng

- Xác định các thành phần liên thông của đồ thị.
- Xác định các thành phần liên thông mạnh của đồ thị có hướng.
- Kiểm tra một đồ thị xem nó có phải đồ thị phẳng hay không.
- Tìm các cạnh cầu của đồ thị.
- Giải các bài toán như tìm đường đi trong mê cung.
- Tạo mê cung với thuật toán duyệt theo chiều sâu ngẫu nhiên.

Mã giả và triển khai DFS

➢ Phương pháp đệ quy:

```
// thuật toán duyệt theo chiều sâu đệ quy
// vertices:  danh sách đỉnh của đồ thị
// adjMatrix: ma trận kề của đồ thị
// n:         số đỉnh của đồ thị
// index:     vị trí đỉnh cần duyệt
function dfs(vertices[], adjMatrix, n, index):
    đánh dấu đỉnh tại vị trí index đã được duyệt
    hiển thị đỉnh tại vị trí index
    for (tất cả các đỉnh i kề đỉnh index trong ma trận kề):
        if (đỉnh i chưa được duyệt):
            gọi đệ quy duyệt DFS tại đỉnh i
```

```
void dfsRecursion(Vertex* vertices, bool** adjMatrix, int size, int index) {
    vertices[index].visited = true;
    displayVertex(vertices[index]);
    for (int i = 0; i < size; i++)
    {
        if (adjMatrix[index][i] && !vertices[i].visited) {
            dfsRecursion(vertices, adjMatrix, size, i);
        }
    }
}
```

Mã giả và triển khai DFS

➢ Phương pháp sử dụng vòng lặp:

```
// thuật toán duyệt theo chiều sâu dùng vòng lặp
// vertices:  danh sách đỉnh của đồ thị
// adjMatrix: ma trận kề của đồ thị
// n:         số đỉnh của đồ thị
// index:     vị trí đỉnh cần duyệt
function dfs(vertices[], adjMatrix, n, index):
    tạo stack s
    đẩy đỉnh index vào stack s
    while(s chưa rỗng):
        i = pop stack
        if(đỉnh i chưa được duyệt):
            đánh dấu đỉnh i đã duyệt
            hiển thị đỉnh i
            for(tất cả các đỉnh v kề với i):
                đẩy v vào stack
```

Mã giả và triển khai DFS

➢ Phương pháp sử dụng vòng lặp:

```
// thuật toán DFS sử dụng vòng lặp
void dfsIteration(Vertex* vertices, bool** adjMatrix, int size, int index) {
    stack<int> stack;
    stack.push(index);
    while (!stack.empty())
    {
        index = stack.top();
        stack.pop();
        if (!vertices[index].visited) {
            vertices[index].visited = true;
            displayVertex(vertices[index]);
            for (int i = 0; i < size; i++)
            {
                if (adjMatrix[index][i]) {
                    stack.push(i);
                }
            }
        }
    }
}
```

Nội dung tiếp theo

Thuật toán tìm kiếm theo chiều rộng (BFS)