BÁO CÁO: THUẬT TOÁN FLOOD FILL

Trần Công Hiệp

Ngày 20 tháng 1 năm 2025

Tóm tắt nội dung

Báo cáo này trình bày thuật toán Flood Fill, một phương pháp thường dùng trong xử lý đồ họa để lấp màu khu vực liên tục. Nội dung bào cáo bao gồm: mô tả vấn đề, phân tích thuật toán, mã giả minh họa, và ví dụ chi tiết. Kết quả cho thấy đây là một giải pháp đơn giản, hiệu quả nhờ vào cơ chế duyệt theo chiều rộng.

1. Giới thiệu

Thuật toán Flood Fill là một phương pháp quán trọng trong lĩnh vực xử lý đồ họa và đồ thị. Nó được sử dụng để tìm và lấp màu cho một vùng liên tục trong ma trận (hoặc hình ảnh). Mô tả vấn đề như sau:

- Cho một ma trận hai chiều biểu diễn hình ảnh, trong đó mỗi phần tử đại diện cho một pixel.
- Cho tọa độ (x, y) của pixel cần bắt đầu.
- Mục tiêu: Thay đổi màu tất cả các pixel liên kết cùng màu với pixel ban đầu thành một màu mới.

2. Phân tích thuật toán

2.1. Ý tưởng thuật toán

Thuật toán Flood Fill dựa trên việc duyệt theo chiều rộng (BFS) sử dụng hàng đợi:

1. Thêm pixel ban đầu vào hàng đợi.
2. Trong khi hàng đợi chưa rỗng:
    a) Lấy pixel đầu tiên ra khỏi hàng đợi.
    b) Kiểm tra các pixel lân cận (trên, dưới, trái, phải).
    c) Nếu pixel lân cận có cùng màu với pixel ban đầu, thêm nó vào hàng đợi.

2.2 Độ phức tạp

Với ma trận kích thước m×n, thuật toán có độ phức tạp thời gian là O(m×n) vì mỗi pixel được duyệt một lần duy nhất.

3. Mã giả chi tiết

3.1 Mã giả

Dưới đây là mã giả (pseudocode) cho thuật toán Flood Fill dùng hàng đợi:

Hàm floodFill(matrix, x, y, newColor):

1. oldColor = matrix[x][y]
2. Nếu oldColor == newColor: • Trả về (không làm gì cả).
3. Tạo một hàng đợi q, thêm (x, y) vào q.
4. Trong khi q không rỗng: a. Lấy (cx, cy) ra khỏi q. b. matrix[cx][cy] = newColor. c. Kiểm tra các pixel lân cận (cx ±1, cy ±0) và (cx ±0, cy ±1):
    - Nếu pixel có cùng oldColor và nằm trong giới hạn ma trận, thêm vào q.

3.2 Cài đặt mã giả

```
      void floodFill(int[][] matrix, int x, int y, int newColor) {
int oldColor = matrix[x][y];
if (oldColor == newColor) return;
Queue<int[]> q = new LinkedList<>();
q.add(new int[]{x, y});

while (!q.isEmpty()) {
    int[] pixel = q.poll();
    int cx = pixel[0], cy = pixel[1];
    matrix[cx][cy] = newColor;

    for (int[] d : new int[][]{{1, 0}, {-1, 0}, {0, 1}, {0, -1}}) {
        int nx = cx + d[0], ny = cy + d[1];
        if (nx >= 0 && ny >= 0 && nx < matrix.length && ny < matrix[0].length
&& matrix[nx][ny] == oldColor) {
            q.add(new int[]{nx, ny});
        }
    }
}
}
```

4 Ví dụ

Giả sử dụng ma trận:

$$\begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Pixel bắt đầu: (0, 0), newColor = 2.

Kết quả sau khi lặp màu:

$$\begin{bmatrix} 2 & 2 & 0 \\ 2 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

5 Kết luận

Thuật toán Flood Fill dùng hàng đợi cho thấy:

- Hiệu quả trong việc xử lý khu vực liên kết trong ma trận.
- Độ phức tạp thời gian tỷ tuyến O(m×n).
- Là cơ chế quan trọng trong xử lý đồ họa và các bài toán tìm kiếm liên quan.

Tài liệu tham khảo:

[1] GeeksforGeeks. "Flood Fill Algorithm – How to Implement fill() in Paint?"

https://www.geeksforgeeks.org/flood-fill-algorithm/

[2] Techie Delight. "Flood Fill Algorithm."

https://www.techiedelight.com/flood-fill-algorithm/