Giới thiệu về BFS trên mảng hai chiều

BFS (Breadth-First Search) là kỹ thuật duyệt theo từng lớp, hoạt động theo cơ chế "gặp ai trước duyệt người đó trước". Khi áp dụng lên ma trận 2 chiều, BFS trở thành một công cụ rất mạnh mẽ để xử lý các bài toán:

- **Tìm đường đi ngắn nhất** trên lưới 2 chiều.
- **Mô phỏng hiện tượng lan truyền**: dịch bệnh, lửa cháy, lũ tràn, mực nước...
- **Tìm vùng lan rộng đồng đều về thời gian**.
- **Tính số bước để biến đổi từ trạng thái này sang trạng thái khác.**

Mô hình BFS trong ma trận

Giả sử bạn có một lưới kích thước $N \times M$, mỗi ô có thể là:

- Trạng thái đã được duyệt (visited).
- Trạng thái đang chờ được mở rộng (queue).
- Trạng thái cản trở hoặc không hợp lệ.

Giả mã (Pseudocode)

```
BFS(start):
    Khởi tạo hàng đợi Q với ô bắt đầu
    Đánh dấu ô bắt đầu là đã thăm
    trong khi Q không rỗng:
        lấy ô (r, c) đầu tiên ra khỏi Q
        với mỗi hướng (dx, dy):
            ô mới (nr, nc) = (r + dx, c + dy)
            nếu (nr, nc) nằm trong lưới và chưa thăm và thỏa điều kiện:
                đánh dấu (nr, nc) là đã thăm
                thêm (nr, nc) vào Q
```

Ứng dụng chính của BFS 2D

Bài toán	Mô tả
Lan truyền dịch bệnh / nước lũ	Mỗi bước BFS tương ứng với 1 đơn vị thời gian lan ra các ô lân cận.
Tìm đường đi ngắn nhất	BFS sẽ tìm đúng đường ít bước nhất từ A đến B (trên grid không trọng số).
Đếm số bước để bao phủ toàn bộ lưới	Dùng khi nhiều điểm xuất phát lan ra đồng thời.
Chuyển trạng thái lưới theo thời gian	Tính toán số bước tối thiểu để chuyển toàn bộ grid về trạng thái mục tiêu.

Bài toán mẫu: Rotting Oranges (LeetCode 994)

Đề bài

Cho một ma trận 2D gồm:

- 0: ô trống.
- 1: quả cam tươi.
- 2: quả cam thối.

Mỗi phút, cam thối sẽ làm các quả cam tươi **lân cận (4 hướng)** bị thối.
Tính **số phút tối thiểu** để tất cả cam đều bị thối. Nếu không thể, trả về -1.

Mô phỏng giải bằng BFS

- Bắt đầu với tất cả vị trí cam thối, cho vào queue.
- Với mỗi phút, xử lý tất cả cam trong queue hiện tại.
- Với mỗi cam, duyệt 4 hướng, làm thối các cam tươi xung quanh và đưa chúng vào queue mới.
- Đếm số phút cho đến khi không còn cam tươi.

Code mẫu C++

```cpp
#include <vector>
#include <queue>
using namespace std;

int orangesRotting(vector<vector<int>>& grid) {
    int rows = grid.size(), cols = grid[0].size();
    queue<pair<int, int>> q;
    int fresh = 0;

    for (int r = 0; r < rows; ++r) {
        for (int c = 0; c < cols; ++c) {
            if (grid[r][c] == 2) q.push({r, c});
            else if (grid[r][c] == 1) fresh++;
        }
    }

    int minutes = 0;
    vector<pair<int, int>> dirs = {{-1,0}, {1,0}, {0,-1}, {0,1}};

    while (!q.empty() && fresh > 0) {
        int sz = q.size();
        for (int i = 0; i < sz; ++i) {
            auto [r, c] = q.front(); q.pop();
            for (auto [dr, dc] : dirs) {
                int nr = r + dr, nc = c + dc;
                if (nr >= 0 && nr < rows && nc >= 0 && nc < cols && grid[nr][nc] == 1) {
                    grid[nr][nc] = 2;
                    fresh--;
```

```
                    q.push({nr, nc});
                }
            }
        }
        minutes++;
    }

    return (fresh == 0) ? minutes : -1;
}
```

Các bài tương tự để luyện tập:

- Shortest Path in Binary Matrix (LeetCode 1091)
- Walls and Gates (LeetCode 286)
- 01 Matrix (LeetCode 542)
- Zombie in Matrix (LintCode 598)
- Cut Off Trees for Golf Event (LeetCode 675)
- Open the Lock (LeetCode 752)
- Knight Shortest Path (LintCode 611)
- Escape from the Grid (Codeforces 1705D)
- Swim in Rising Water (LeetCode 778)

Kết luận

Kỹ thuật BFS 2D cực kỳ mạnh mẽ và thường xuất hiện trong các bài toán:

- Mô phỏng lan truyền.
- Tìm đường đi tối ưu.
- Đếm số bước hoặc tính thời gian.

So với DFS, BFS giúp **tìm lời giải nhanh nhất về mặt bước đi (độ sâu nhỏ nhất)** và có tính ứng dụng rất cao trong thực tế.