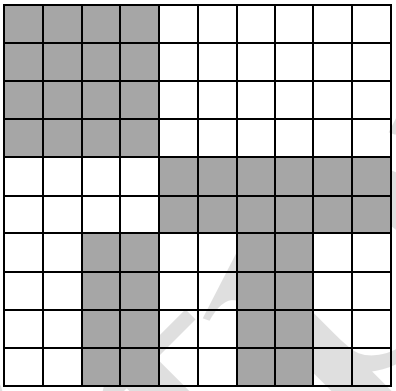
KỸ THUẬT TÌM KIẾM THEO CHIỀU RỘNG, TÌM KIẾM THEO CHIỀU SÂU

1. Bài toán mở đầu

Cho mảng 2 chiều kích thước $m \times n$, mỗi phần tử trong mảng chỉ mang một trong hai giá trị 0 hoặc 1. Đếm số vùng chỉ gồm số 0 hoặc số 1 trên mảng. Mỗi vùng là một nhóm lớn nhất gồm các ô chung cạnh và có cùng giá trị. Cho biết vùng có số lượng ô là lớn nhất.



Mảng 2 chiều trong bài toán có thể minh họa như hình trên, quy ước ô mang giá trị 0 tương ứng với ô có màu trắng, ngược lại ô mang giá trị 1 tương ứng với ô có màu đen. Số vùng màu đen là 3 và số vùng có màu trắng là 4.

Mỗi vùng như thế được gọi là một miền liên thông. Bài toán trên thuộc dạng bài toán đếm số miền liên thông. Bài toán duyệt qua tất cả vùng để đếm số miền liên thông được giải quyết bằng kỹ thuật duyệt theo chiều sâu và chiều rộng.

2. Duyệt theo chiều sâu (DFS)

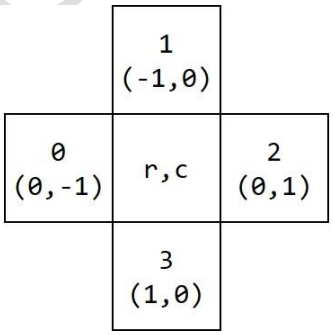
Ý tưởng: khi có đường đi, ưu tiên chọn hướng đi đầu tiên tìm thấy. Nếu không còn đường để đi tiếp, quay ngược lại con đường vừa đi để chọn hướng ưu tiên thứ hai.

Tư tưởng: nếu có nhiều đường đi, chọn đi đường đầu tiên nhìn thấy.

Mô hình duyệt theo chiều sâu từ ô (r, c)

```
DFS(r, c)
{
    mark[r][c] = true;
    for (x, y) ∈ {tập các ô kề với ô (r, c)}
        if (mark[x][y] = false)
            DFS(x, y);
}
```

Cài đặt hàm duyệt theo chiều sâu. Sử dụng 2 mảng hằng `dr`, và `dc` theo hình minh họa. Hàm `DFS(r,c)` trả về số lượng ô cùng vùng với ô `(r,c)`.




```
int dr[4] = {0, -1, 0, 1};
int dc[4] = {-1, 0, 1, 0};

bool mark[maxN+1][maxN+1] = {false};
int a[maxN+1][maxN+1];

bool ValidCell(int r, int c)
{
    return (r >= 1 && r <= m && c >= 1 && c <= n);
}

int DFS(int r, int c)
{
    int res = 1; //hiện tại duyệt đến ô (r, c) và số ô đã duyệt là 1
    mark[r][c] = true; //đánh dấu lại ô đã đi qua
    for (int i = 0; i < 4; ++i)
    {
        int x = r + dr[i], y = c + dc[i];
        int v = a[r][c];
        if (ValidCell(x, y) && !mark[x][y] && a[x][y] == v)
            res = res + DFS(x, y);
    }
    return res;
}
```

Thuật toán duyệt theo chiều sâu được ứng dụng:

- Tìm đường đi có thứ tự từ điển nhỏ nhất trên đồ thị.

- Tìm cha chung gần nhất trên cây.
- Tìm cầu trên đồ thị.
- Tìm miền liên thông trên đồ thị hoặc trên bảng.
- Tô màu hình khép kín.

3. Duyệt theo chiều rộng (BFS - loang)

Ý tưởng: khi có nhiều hướng đi để chọn, ưu tiên chọn những hướng đi đến các vùng lân cận điểm hiện tại.

Tư tưởng: gần đến trước, xa đến sau.

Duyệt theo chiều rộng còn được gọi là Loang, lấy ý tưởng hình loang của dầu trên mặt nước: những vị trí gần nơi dầu rớt xuống sẽ bị nhiễm dầu trước những vị trí xa hơn.

Mô hình duyệt theo chiều rộng

```
BFS(r, c)
{
    mark[r][c] = true; //đánh dấu ô (r, c) đã xét
    Q.Push(r, c); //thêm ô (r, c) vào hàng đợi
    while (!Q.empty) //lặp khi hàng đợi khác rỗng
    {
        (r, c) = Q.Pop; //lấy phần tử đầu của hàng đợi
        for (x, y) ∈ {tập các ô gần kề với ô (r, c)}
            if (mark[x][y] == false) //nếu ô (x, y) chưa xét
            {
                Q.Push(x, y);
                mark[x][y] = true;
            }
    }
}
```

Cài đặt hàm duyệt theo chiều rộng. Hàm `BFS(r,c)` trả về số lượng ô cùng vùng với ô `(r,c)`.

```
int dr[4] = {0, -1, 0, 1};
int dc[4] = {-1, 0, 1, 0};

typedef pair<int, int> pii;
```

```
int BFS(int r, int c)
{
    queue<pii> Q;
    Q.push(pii(r, c));
    mark[r][c] = true;
    int res = 1;
    while (!Q.empty())
    {
        r = Q.front().first, c = Q.front().second;
        Q.pop(); //loại phần tử đầu ra khỏi hàng đợi
        for (int i = 0; i < 4; ++i)
        {
            int x = r + dr[i], y = c + dc[i];
            int v = a[r][c];
            if (ValidCell(x, y) && !mark[x, y] && a[x][y] == v)
            {
                ++res;
                Q.push(pii(x, y));
                mark[x][y] = true;
            }
        }
    }
    return res;
}
```

Duyệt theo chiều rộng được ứng dụng

- Tìm đường đi ngắn nhất trên đồ thị không trọng số
- Giải các bài toán biến đổi trạng thái với yêu cầu tìm số phép biến đổi hoặc số phép di chuyển là ít nhất.
- Tìm miền liên thông trên đồ thị hoặc trên bảng.

Hàm đếm số miền liên thông trên bảng

```
void Solve()
{
    int ans = 0, maxSize = 0;
    for (int x = 0; x < m; ++x)
        for (int y = 0; y < n; ++y)
            if (mark[x][y] == false)
            {
                int size = BFS(x, y); //hoặc int size = DFS(x, y);
                maxSize = max(size, maxSize);
```

```
                ++ans;
            }

    cout<<"số lượng vùng: "<<ans<<"\n";
    cout<<"kích thước vùng lớn nhất: "<<maxSize;
}
```