Dial's Algorthm

Giới thiệu thuật toán Dial's

Thuật toán Dial's là thuật toán tìm các đường đi ngắn nhất từ đỉnh s bất kì đến các đỉnh còn lại trong đồ thị (có trọng số không âm và nhỏ hơn hoặc bằng K).

Thuật toán Dial's có độ phức tạp là $O(KN + M)$, trong đó N là số đỉnh, M là số cạnh của đồ thị.

Cấu trúc dữ liệu và giải thuật Dial's

- **Cấu trúc dữ liệu:**

Mảng d: lưu lại giá trị của đường đi ngắn nhất ($d[i]$: giá trị đường đị ngắn nhất $s \to i$), ban đầu gán bằng ∞.

Mảng $color$: đánh dấu các đỉnh trong đồ thị, ban đầu tất cả bằng $false$.

$K + 1$ hàng đợi q: hàng đợi các đỉnh trong đồ thị. (mảng hàng đợi)

Con trỏ ptr: con trỏ hàng đợi, ban đầu gán bằng 0.

- **Thuật toán:**

Bước 1: Đưa đỉnh s vào hàng đợi $q[0]$, gán $d[s] = 0$.

Bước 2: Trong khi hàng đợi $q[ptr]$ không có đỉnh thì gán $ptr \to (ptr + 1)\ \%\ (K + 1)$

Bước 3: Lấy đỉnh u ra từ hàng đợi $q[ptr]$.

Bước 4: Nếu đỉnh u đã được đánh dấu $(color[u] = true)$ thì quay lại bước 2.

Bước 5: Đánh dấu đỉnh u $(color[u] \to true)$. Duyệt các đỉnh v kề với đỉnh u. Nếu đỉnh v chưa được đánh dấu $(color[u] = false)$ thì thêm đỉnh v vào trong hàng đợi $q\left[\left(ptr\ +\ dist(u, v)\right)\ \%\ (K + 1)\right]$

và gán $d[v] = \min\left(d[v], d[u] + dist(u, v)\right)$.

Bước 6: Nếu các hàng đợi đều rỗng thì kết thúc thuật toán, ngược lại quay lại bước 2.

Thuật toán Dial's trên đồ thị lưới

Áp dụng thuật toán Dial's cho bài toán ở *mục thuật toán Dijkstra*, tham khảo cách cài đặt sau đây:

```cpp
#include <bits/stdc++.h>
using namespace std;
const int N = 1e3 + 5;
int dx[] = { -1, 1, 0, 0 };
int dy[] = { 0, 0, 1, -1 };
int n, m, X, Y, U, V;
int d[N][N]; // d[i][j]=độ dài đường đi từ ngắn nhất (x, y) -> (u, v)
char ch[N][N]; // lưới
queue<pair<int, int>> q[10]; // mảng hàng đợi - K = 9
bool color[N][N]; // đánh dấu các ô trong lưới, ban đầu khởi tạo false
bool inGrid(int r, int c) {
    if (r >= 1 && r <= n && c >= 1 && c <= m) return true;
    return false;
}
int weight(char c) {
    if (c == 'R' || c == 'G') return 0;
    return c - 48;
}
int main() {
    ios_base::sync_with_stdio(0);
    cin.tie(0);
    cin >> n >> m;
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= m; j++) {
            cin >> ch[i][j];
            if (ch[i][j] == 'G')
                X = i, Y = j;
            else if (ch[i][j] == 'R')
                U = i, V = j;
            d[i][j] = 10 * (n + m);
        }
    }
    q[0].push({ X, Y }); // đưa ô (X, Y) vào hàng đợi q[0]
    d[X][Y] = 0;
    int ptr = 0;
    while (true) {
        int cycle = ptr;
        while (q[ptr].empty()) {
```

```
            ptr = (ptr + 1) % 10;
            if (ptr == cycle) break;
        }
        if (q[ptr].empty()) break;
        int x = q[ptr].front().first;
        int y = q[ptr].front().second;
        q[ptr].pop(); // lấy ô (x, y) ra khỏi hàng đợi
        if (color[x][y]) continue;
        color[x][y] = true;
        for (int i = 0; i < 4; i++) {
            int u = x + dx[i],v = y + dy[i];//xét ô (u,v) kề với (x,y)
            if (!inGrid(u, v) || color[u][v]) continue;
            d[u][v] = min(d[u][v], d[x][y] + weight(ch[u][v]));
            q[(ptr + weight(ch[u][v])) % 10].push({u, v});
        }
    }
    cout << d[U][V];
}
```

- ***Mô phỏng thuật toán Dial trên lưới***