KUPC2020 spring B: サイコロを転がさないで 解説

writer: suibaka, yamunaku, zaki

2020 年 3 月 20 日

解説

i, j がともに偶数となるマス (i, j) にサイコロを転がせないことが重要です．実はこの制約により，マスの座標ごとに，そこにサイコロを転がしたときに底面となりうる数字が 1 つに確定します．具体的には以下のようになります．

6	3	1	4	6	$\cdots$
2	#	2	#	2	$\cdots$
1	3	6	4	1	$\cdots$
5	#	5	#	5	$\cdots$
6	3	1	4	6	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

よって，入力で与えられたグリッドとこの表を比較することで各マスが有効かどうかが定まります．あとは有効なマスのみを考えたときに $(1, 1)$ と (H, W) が連結かどうかを判定すればよく，計算量は $O(HW)$ となります．

また別解として，サイコロの回転をシミュレートする方法もあります．この解法ではサイコロの回転を実装する必要があるため，比較的実装が重くなってしまいます．

注意：次ページに実装例があります．

解答例

```cpp
#include <bits/stdc++.h>
using namespace std;

constexpr int dy[4] = {0, 1, 0, -1}, dx[4] = {1, 0, -1, 0};
const string dice[4] = {"6314", "2*2*", "1364", "5*5*"};

int main() {
    int h, w; cin >> h >> w;
    vector<string> s(h);
    for(auto& x : s) cin >> x;

    vector<vector<bool>> vis(h, vector<bool>(w));
    function<void(int, int)> dfs = [&] (int y, int x) {
        if(y < 0 || h <= y || x < 0 || w <= x) return;
        if(vis[y][x] || s[y][x] != dice[y % 4][x % 4]) return;
        vis[y][x] = true;
        for(int i = 0; i < 4; ++i) {
            dfs(y + dy[i], x + dx[i]);
        }
    };
    dfs(0, 0);

    cout << (vis[h - 1][w - 1] ? "YES" : "NO") << endl;
}
```