백준 1981번 - 배열에서 이동

시간제한	메모리 제한	제출	정답	맞은 사람	정답 비율
1초	256MB	6205	1564	1029	23.819%

문제

n×n짜리의 배열이 하나 있다. 이 배열의 (1, 1)에서 (n, n)까지 이동하려고 한다. 이동할 때는 상, 하, 좌, 우의 네 인접한 칸으로만 이동할 수 있다.

이와 같이 이동하다 보면, 배열에서 몇 개의 수를 거쳐서 이동하게 된다. 이동하기 위해 거쳐 간 수들 중 최댓값과 최솟값의 차이가 가장 작아지는 경우를 구하는 프로그램을 작성하시오.

입력

첫째 줄에 n(2 ≤ n ≤ 100)이 주어진다. 다음 n개의 줄에는 배열이 주어진다. 배열의 각 수는 0보다 크거나 같고, 200보다 작거나 같은 정수이다.

출력

첫째 줄에 (최대 - 최소)가 가장 작아질 때의 그 값을 출력한다.

예제 입력1

```
5
1 1 3 6 8
1 2 2 5 5
4 4 0 3 3
8 0 2 3 4
4 3 0 2 1
```

예제 출력1

```
2
```

출처

출처

알고리즘 분류

- 그래프 이론
- 그래프 탐색
- 이분 탐색
- 너비 우선 탐색

접근 방법

단순하게 backtracking으로 (1, 1) → (n, n) 까지의 모든 경로를 구한 다음 최적해를 찾는 방식은 시간 초과를 받을 것이 당연하다. ($O(2^{N^2})$)

최대값과 최솟값의 범위가 그렇게 크지 않으므로, 임의의 최댓값(`mx`)와 최솟값(`mn`)을 잡은 다음

$mn \leq block \leq mx$ 사이의 block들만 밟으면서 (n, n) 까지 도달할 수 있는지에 대하여 생각해보자.

그렇다면, 최대 200 * 200 번 그래프 탐색을 수행하고 한번의 그래프 탐색마다 $O(N^2)$의 비용이 든다.

여기서 parametric search를 적용할 수 있다.

`f(k) := (최대 - 최소) 값이 k 라고 가정하였을 때, (1, 1) -> (n, n)까지 도달 가능한가?`

위와 같이 함수를 잡을 경우, 특정 k값을 기준으로 true/false가 이분화된다는 것은 자명하다.

$f(k)$가 true일 경우, 더 작은 범위의 k에 대하여 추가적인 탐색을 수행하는 방식으로 문제를 해결할 수 있다.

소스코드

```cpp
#define FASTIO cin.tie(0)->sync_with_stdio(false), cout.tie(0)
////////////////////////////////////////////////////////////////
#include <bits/stdc++.h>
using namespace std;
int N, board[105][105], ans = 1e9;
const int dir[4][2] = {{-1, 0}, {1, 0}, {0, -1}, {0, 1}};
bool check(int mn, int mx){
    queue<pair<int,int>> Q;
    if(board[1][1] < mn || board[1][1] > mx)
        return false;
    Q.push({1, 1});

    bool visited[105][105] = {false, };
    while(!Q.empty()){
```

```
        auto [y, x] = Q.front(); Q.pop();
        if(y == N && x == N && (mn <= board[y][x] && board[y][x] <= mx))
            return true;
        for(int d=0; d<4; d++){
            int ny = y + dir[d][0], nx = x + dir[d][1];
            if(ny <= 0 || nx <= 0 || ny > N || nx > N || visited[ny][nx])
                continue;
            if(board[ny][nx] < mn || board[ny][nx] > mx)
                continue;

            visited[ny][nx] = true;
            Q.push({ny, nx});
        }
    }
    return false;
}
int main(void){
    FASTIO;
////////////////////////////////////////////////////////////////
    cin >> N;
    for(int i=1; i<=N; i++){
        for(int j=1; j<=N; j++){
            cin >> board[i][j];
        }
    }

    int l = 0, r = 200;
    while(l <= r){
        int m = (l + r) / 2;
        // 1. mx >= mn
        // 2. m = mx - mn
        // 2-1) mx := [0, 200], mn := [0, mx]
        bool isOk = false;
        for(int mx=0; mx<=200 && !isOk; mx++){
            int mn = mx - m;
            // mn 이상 mx 이하의 칸만 밟고 (n, n)에 도달가능한가?
            isOk = check(mn, mx);
        }
        if(isOk){
            ans = m;
            r = m - 1;
        }else{
            l = m + 1;
        }
    }
    cout << ans ;
    return 0;
}
```