АНАЛИЗ НА РЕШЕНИЕТО НА ЗАДАЧА НИНДЖА

Задачата може да се сведе до задача за намиране на най-кратък път в лабиринт със специални правила кога може и кога не може да се преминава през дадена клетка като тези правила зависят от момента на преминаване през клетките в него.

Важно нещо, което трябва да се съобрази, за да се оптимизира решението на задачата е, че героят в задачата (Лъчо) няма нужда да седи в една стая повече от 1 секунда, за да чака промяна на състоянието на осветлението. Ако Лъчо изчака 2 секунди, осветлението ще си е в първоначалното положение и той ще е загубил 2 секунди в безсмислено чакане, което само ще го забави, а все пак в задачата се търси минимално време.

Съобразявайки това нещо, лесно може да се измисли интуитивно решение, базирано на пълно обхождане на възможните пътища, през които Лъчо би могъл да мине. Във една всяка стая той има избор измежду 4 възможности:

- Веднага щом влезе в стаята Лъчо да тръгне надолу, ако е възможно.
- Веднага щом влезе в стаята Лъчо да тръгне надясно, ако е възможно.
- Лъчо да изчака 1 секунда и след това да тръгне надолу, ако е възможно.
- Лъчо да изчака 1 секунда и след това да тръгне надясно, ако е възможно.

На Лъчо НЕ е възможно да отиде в някоя от посоките ако:

- Стаята, в която Лъчо иска да отиде, се намира извън рамките на зададените размери за *N*.
- В момента, в който Лъчо ще стъпи в другата стая, тя свети.

По-надолу следва примерно решение, базирано на техниката пълно изчерпване. Всеки път, когато достигнем до крайната цел, проверяваме дали времето, за което сме стигнали в текущия път, не е по-добро от най-доброто намерено до сега.

```
#include <iostream>
#define MAXN 15
#define MAXTIME 2000000000
#define INIT_STATE_ON 1
#define INIT_STATE_OFF 0
using namespace std;

int N;
int matrix[MAXN][MAXN];
int bestAnswer = MAXTIME;

void bt(int time, int x, int y)
{
    // Check if the room is light
    if (matrix[x][y] == INIT_STATE_ON && (time % 2) == 1) return; //time = 1, 3, 5...
    if (matrix[x][y] == INIT_STATE_OFF && (time % 2) == 0) return; //time = 2, 4, 6...

    // Check if we have reached the target
    if (x == N - 1 && y == N - 1)
    {
        if (time < bestAnswer) bestAnswer = time;
        return;
    }

    // Don't wait and visit next cells
    if (x + 1 < N) bt(time + 1, x + 1, y);
    if (y + 1 < N) bt(time + 1, x, y + 1);
```

```
    // Wait 1 second and visit next cells
    if (x + 1 < N) bt(time + 2, x + 1, y);
    if (y + 1 < N) bt(time + 2, x, y + 1);
}

int main()
{
    // Read input
    cin >> N;
    for(int i = 0; i < N; i++)
    {
        for(int j = 0; j < N; j++)
        {
            cin >> matrix[i][j];
        }
    }

    // Solve the task using backtracking
    bt(1, 0, 0);

    // Output the answer
    cout << bestAnswer << endl;

    return 0;
}
```

Задачата може да бъде решена доста по-ефективно с техниката динамично оптимиране. В материалите към задачата има прикачени 2 допълнителни решения базирани съответно на едномерно и двумерно динамично оптимиране.

Забележка: За примерния вход от условието на задачата едно от възможните положения на Лъчо в стаите (разписано по секунди) може да бъде: (0,0), (0,1), (0,1), (1,1), (2,1), (2,2), (2,2), (2,3), (2,3), (3,3).

Автор: Николай Костов