

문제해결기법 - 201921725 안성현

스 위 치

<2021/11/14>

■ 문제 소개 및 접근법

1-1] 스위치

① 문제

전등 n 개가 일렬로 서 있다. 각 전등은 켜져 있을 수도 있고 꺼져 있을 수도 있다. 전등마다 스위치가 하나씩 붙어 있는데, 이 스위치를 누르면 해당 전등뿐만 아니라 바로 왼쪽에 있는 것과 바로 오른쪽에 있는 전등의 상태도 바뀌게 된다. 1번 전등과 n 번 전등은 두 개의 전등 상태만 바꾼다. 이 때 모든 전등을 끄는데 필요한 최소 스위치 조작 횟수를 계산하는 프로그램을 작성하라.

(첫째 줄에 전등의 개수를 나타내는 양의 정수 n 이 입력된다. 단, $n \leq 100,000$ 이다. 둘째 줄에는 n 개의 정수가 주어지는데, i 번째 정수는 i 번 전등이 켜져 있으면 1이고 꺼져 있으면 0이다. 입력으로 주어진 전등을 모두 끌 수 있으면 최소 스위치 조작 횟수를 출력한다. 모든 전등을 끌 수 있는 방법이 없으면 'impossible'을 출력한다.)

② 설명

▶ 필자는 '탐욕법(Greedy Algorithm)'으로 문제를 풀었다. '탐욕법'은 현재 상황에서 가장 좋은 방법을 선택하는 알고리즘이다. 현재 상황은 '최소 클릭 횟수를 구하는 것'이니 모든 스위치에 대해 (클릭 할지/말지)는 딱 한 번만 결정하고 조작하는 것이 최선이다. 모든 스위치에 대해 두 번 이상 조작해서 모든 전등을 끌 수도 있겠지만 한 번씩만 조작해서 구한 횟수가 정답이라고 믿고 문제를 풀었다. 만약 한 번씩만 조작해서 문제를 풀었는데 켜진 전구가 존재한다면 모든 전등을 끌 수 있는 방법이 없다고 생각했다. 참고로 탐욕법이 최적 해를 보장하는지는 알 수 없다. '결정'은 앞 전구의 상태를 보고 진행했다. 문제에 의하면 현재 스위치를 클릭하면 앞 전구의 상태도 변하게 된다. 따라서 앞 전구가 켜져 있으면 현재 스위치를 클릭해서 꺼지게 하고, 앞 전구가 꺼져 있으면 현재 스위치를 클릭하지 않아서 꺼진 상태로 내버려 두었다. 이 과정을 첫 번째 스위치를 제외한 모든 스위치에 대해서 진행했다.

(만약 현재 전구 상태를 보고 스위치를 조작하면 이전에 조작해서 상태가 변한 전구까지 영향을 미치게 되므로 이전 전구 상태를 확인한 것이다.) 첫 번째 스위치는 앞 전구가 존재하지 않기 때문이다. 하지만 첫 번째 스위치를 클릭하거나 클릭하지 않을 때 최소 횟수가 나올 수도 있으니 케이스를 둘로 나누어서 문제를 풀었다. 결국 첫 번째 스위치를 클릭한 후 나머지 스위치들을 조작하는 경우, 첫 번째 스위치를 클릭하지 않고 나머지 스위치들을 조작하는 경우 중 값이 작게 나온 것을 정답으로 채택했다.

소스 코드와 실행 결과

2-1] 소스 코드와 실행 결과

① 코드

```
#pragma warning (disable:4996)
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#define convert(value) value < 1 ? 1:0 // 0->1 , 1->0 convert

void click(bool* lamp, int i, int n) {
    /*
    첫 번째 스위치 클릭 -> i, i+1번째 전등 상태 변경
    마지막 스위치 클릭 -> i-1, i번째 전등 상태 변경
    나머지는 i-1, i, i+1번째 전등 상태 변경
    */
    if (i != 0)
        lamp[i - 1] = convert(lamp[i - 1]);
    if (i != n - 1)
        lamp[i + 1] = convert(lamp[i + 1]);
    lamp[i] = convert(lamp[i]);
}

int switch_control(bool* lamp, bool* lamp_temp, int n) {
    // 스위치 클릭 횟수 (0번 스위치 click, 0번 스위치 non-click)
    int clk, nclk;
    clk = nclk = 0;

    // 전등이 모두 꺼졌는지 확인하는 변수 (0번 스위치 click, 0번 스위치 non-click)
    int check, n_check;

    // 0번 스위치 non-click
    for (int i = 1; i < n; i++) {
        if (lamp[i - 1] != 0) {
            click(lamp, i, n);
            nclk++;
        }
    }
    n_check = lamp[n - 1];

    // 0번 스위치 click
    click(lamp_temp, 0, n);
    clk++;
    for (int i = 1; i < n; i++) {
        if (lamp_temp[i - 1] != 0) {
            click(lamp_temp, i, n);
            clk++;
        }
    }
}
```

```

        }
    }
    check = lamp_temp[n - 1];

    // 전등이 모두 꺼졌는지 확인 -> 클릭 횟수 반환
    if (check == 0 && n_check == 0)
        return clk < nclk ? clk : nclk;
    else if (check == 0)
        return clk;
    else if (n_check == 0)
        return nclk;
    else
        return 0;
}

int main() {
    int n = 0; // 전등의 수
    scanf("%d", &n);

    bool* lamp = (bool*)malloc(sizeof(bool)*n); // 전등 상태 배열
    bool* lamp_temp = (bool*)malloc(sizeof(bool)*n); // Temp 배열

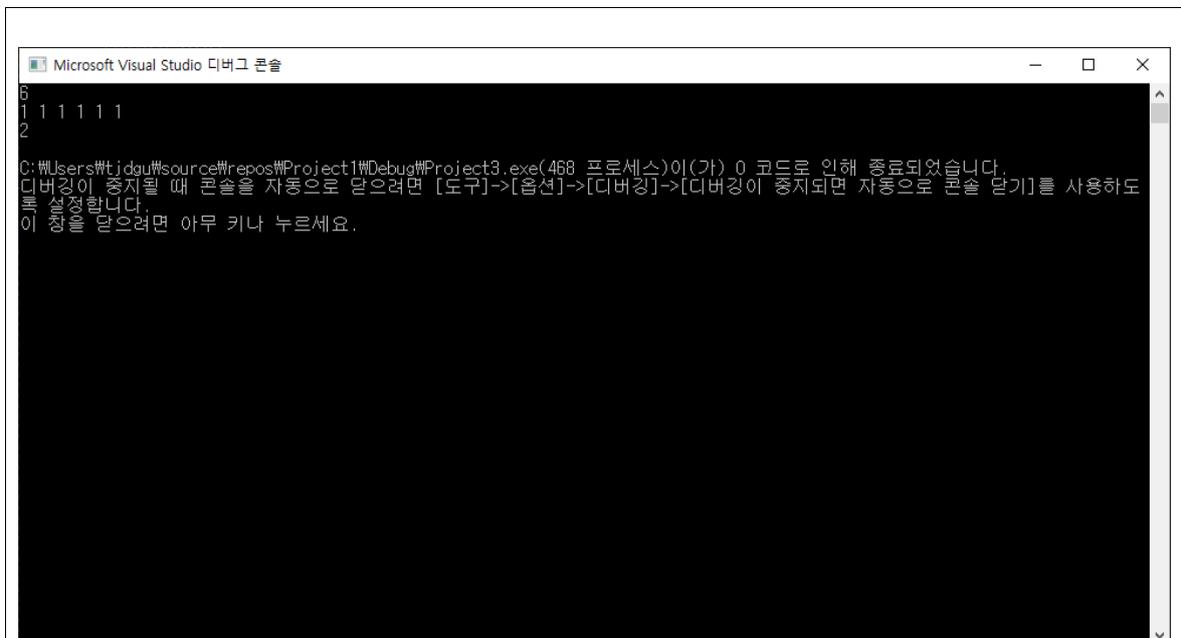
    // 전등 상태 입력 받기
    for (int i = 0; i < n; i++) {
        scanf("%d", &lamp[i]);
        lamp_temp[i] = lamp[i];
    }

    // 모든 전등을 끄기 위한 클릭 횟수 출력
    int num = switch_control(lamp, lamp_temp, n);
    if (num)
        printf("%d\n", num);
    else
        printf("impossible\n");

    return 0;
}

```

② DEV-C++ 컴파일러 이용 실행 결과



```
Microsoft Visual Studio 디버그 콘솔
6
1 1 1 1 1 1
2
C:\Users\#tj#dgu\source\repos\Project1\Debug\Project3.exe(468 프로세스)이(가) 0 코드로 인해 종료되었습니다.
디버깅이 중지될 때 콘솔을 자동으로 닫으려면 [도구]->[옵션]->[디버깅]->[디버깅이 중지되면 자동으로 콘솔 닫기]를 사용하도록 설정합니다.
이 창을 닫으려면 아무 키나 누르세요.
```

입력: n(전등의 개수)=6, 모두 켜져 있는 상태

출력: 2(최소 횟수), 2번 스위치와 5번 스위치를 누르면 전등이 모두 꺼짐