오늘은 백준 실버 1 문제

```c
// C99
#include <stdio.h>

// 매개변수 둘 중 작은 값을 리턴하는 함수
int min(int n1, int n2)
{
    return n1 < n2 ? n1 : n2;
}

int main(void)
{
    int N; // 집의 수(2 ≤ N ≤ 1,000)
    int R, G, B; // 각 집을 빨강, 초록, 파랑으로 칠하는 비용(1 ≤ R, G, B ≤ 1,000)
    int before[3]; // before[n]: i번 이전의 모든 집을 칠하고, i번 집을 n(0: 빨강, 1: 초록, 
    int now[3] = { 0, }; // now[n]: i + 1번 이전의 모든 집을 칠하고, i + 1번 집을 n(0: 빨강

    scanf("%d", &N);
    for (int i = 0; i < N; i++)
    {
        scanf("%d%d%d", &R, &G, &B); // i + 1번 집을 빨강, 초록, 파랑으로 칠하는 비용을 입력

        for (int j = 0; j < 3; j++) // i가 증가함에 따라 이전 i에서의 now[j]는 before[j]가
            before[j] = now[j];
        now[0] = min(before[1], before[2]) + R;
        now[1] = min(before[0], before[2]) + G;
        now[2] = min(before[0], before[1]) + B;
    }
    printf("%d", min(min(now[0], now[1]), now[2]));

    return 0;
}
```

1149번: RGB거리
1149번 제출 맞은 사람 숏코딩 재채점 결과 디버그 채점 현황 강의 RGB…
www.acmicpc.net

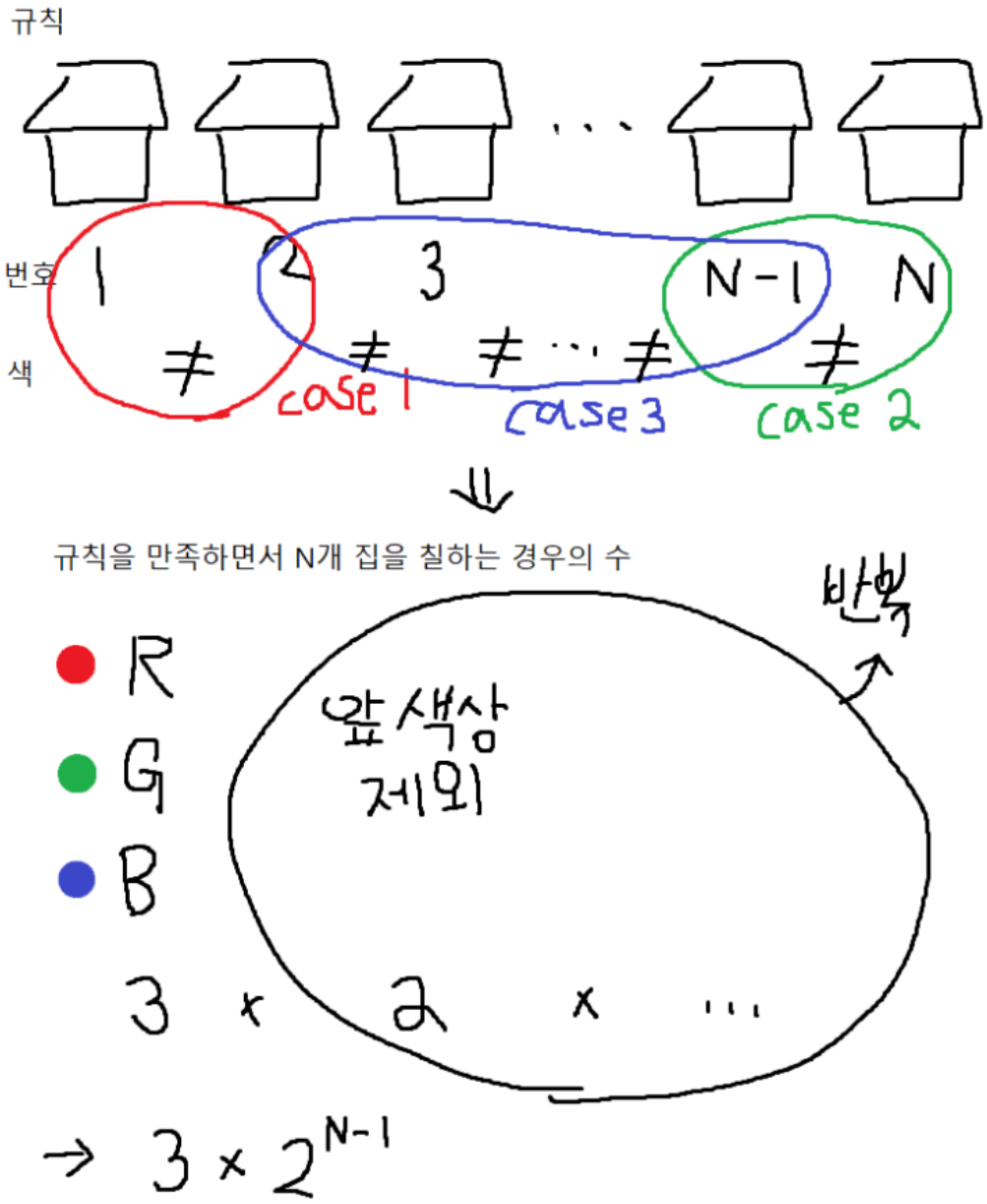
이해하고 정리하는 데 충분한 시간이 필요한 문제입니다
각 집의 색상을 칠할 때 고려해야 할 규칙부터 정리해보겠습니다
코드와 주석을 함께 보면서 읽으면 좀 더 이해하기 수월할 것이라 생각합니다

규칙은 다음과 같습니다

> 1번 집의 색은 2번 집의 색과 같지 않아야 한다.

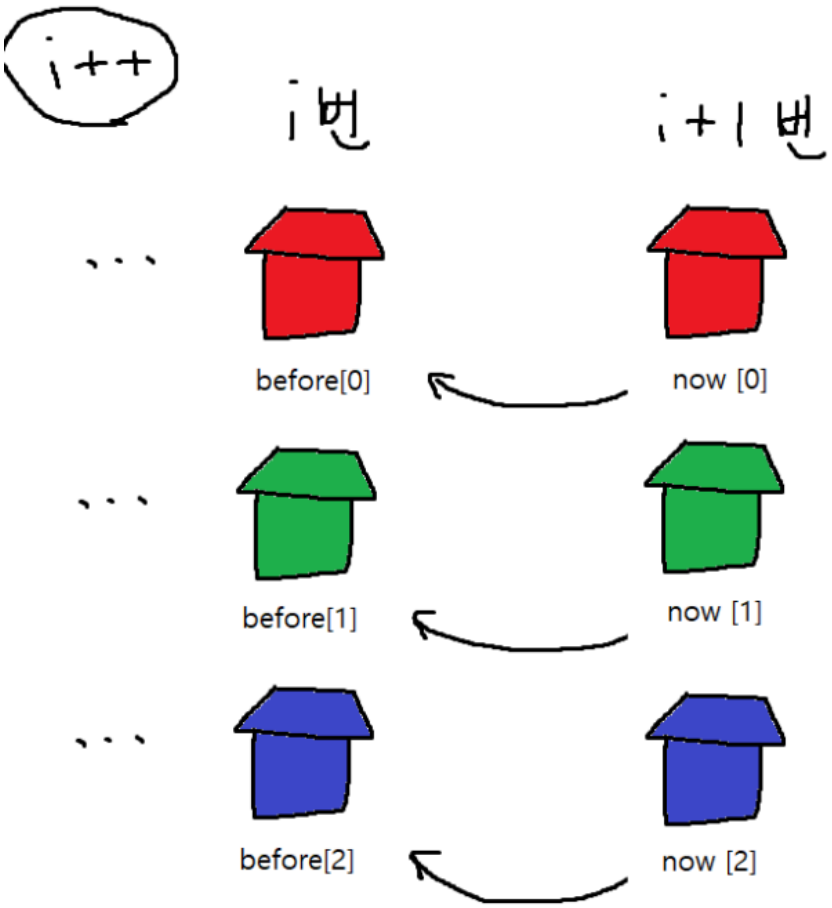
> N번 집의 색은 N-1번 집의 색과 같지 않아야 한다.
> i(2 ≤ i ≤ N-1)번 집의 색은 i-1번, i+1번 집의 색과 같지 않아야 한다.
>
> https://www.acmicpc.net/problem/1149

1번 집의 색은 2번 집의 색과 같지 않아야 한다. → case 1
N번 집의 색은 N-1번 집의 색과 같지 않아야 한다. → case 2
i(2 ≤ i ≤ N-1)번 집의 색은 i-1번, i+1번 집의 색과 같지 않아야 한다. → case 3



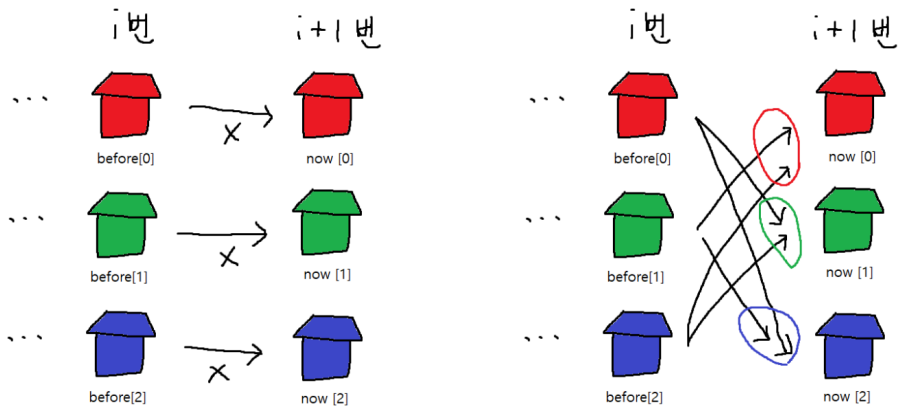
집을 번호순, 오름차순으로 왼쪽부터 정렬했다고 생각해보면
case 1, case 2, case 3는 각각 맨 왼쪽, 맨 오른쪽, 맨 끝이 아닌 모든 집에 적용되는 조건입니다
집을 색칠할 때 인접한 집과는 다른 색으로 칠하라는 말로 규칙은 정리가 되겠네요

규칙을 만족하면서 N개 집을 칠하는 모든 경우의 수는
N이 커질수록 기하급수적으로 늘어납니다
전부 일일이 비교할 수 없는 노릇이니 적절한 알고리즘을 생각해야 합니다



구현부 설명을 해보겠습니다
마지막 집을 빨강, 초록, 파랑으로 칠하기까지 든 비용의 최솟값을 구하고
마지막에 그 중 가장 작은 값을 구하는 구조로 설계했습니다

먼저 i + 1번 집을 색칠하는 비용을 색상별로 입력받습니다
i가 증가하면 이전의 now 값은 이제 before 값이 됩니다
선언부에서 before를 따로 초기화해주지 않은 이유입니다



인접한 집과 다른 색을 칠해야 하므로
now가 선택할 수 있는 before는
자신과 다른 인덱스를 가진 before 두 가지입니다

최솟값을 구하는 문제이므로 더 작은 값을 가진 before를 선택하고
선택한 before와 해당하는 색으로 i + 1번 집을 칠하는 비용(R, G, B)을 더하면
now를 구할 수 있습니다

이제부터는 반복입니다
반복문이 끝나고
now 중 가장 작은 값을 출력하면 됩니다