JavaScript 다이나믹 프로그래밍

다이나믹 프로그래밍 문제 풀이

다이나믹 프로그래밍 이해하기 | 코딩 테스트에서 자주 등장하는 다이나믹 프로그래밍 이해하기

강사 나동빈

JavaScript 다이나믹 프로그래밍

다이나믹 프로그래밍 문제 풀이

쉬운 계단 수: https://www.acmicpc.net/problem/10844

혼자 힘으로 풀어보기

문제 제목: 쉬운 계단 수

문제 난이도: ★★★☆☆

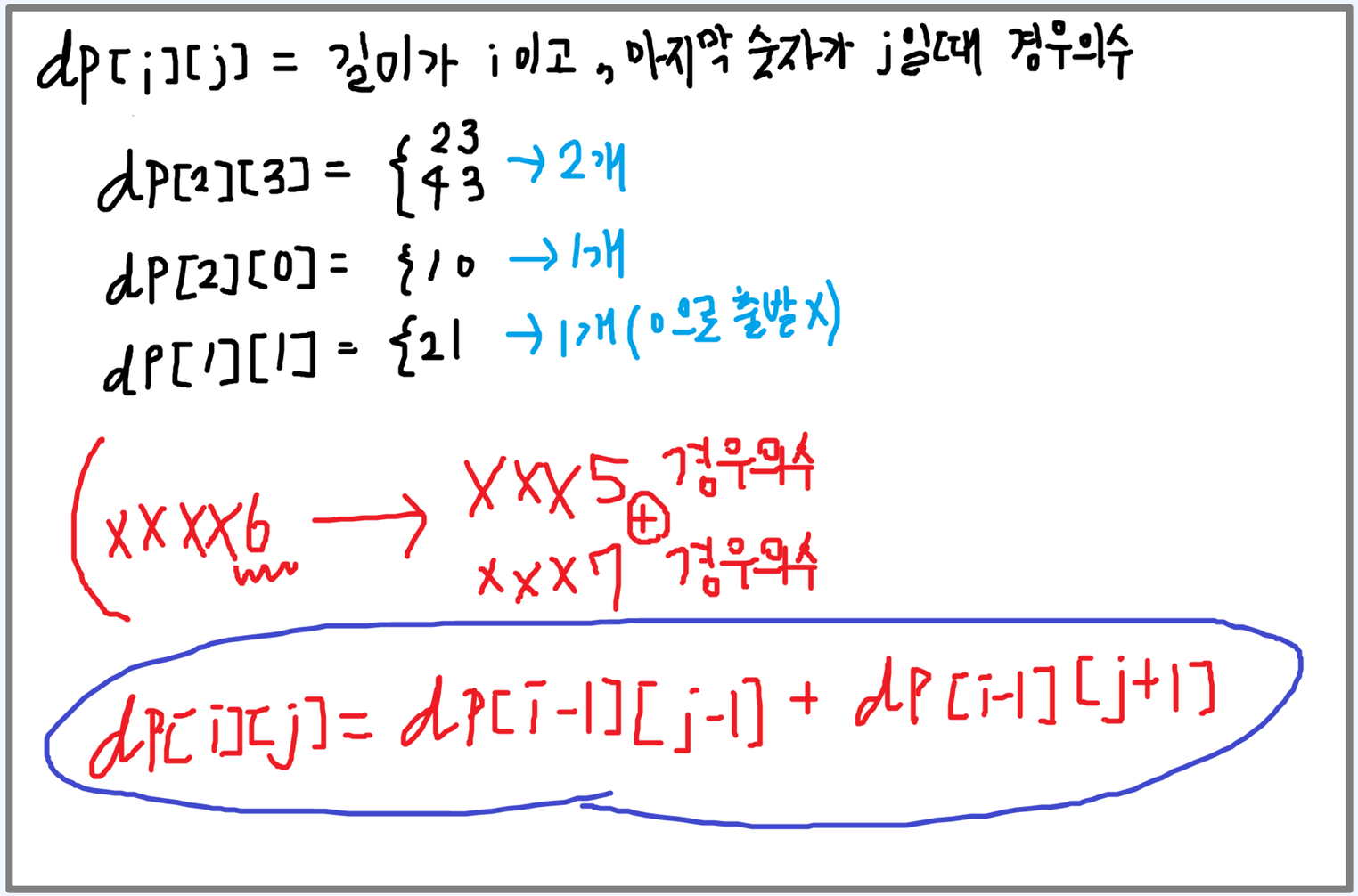
문제 유형: 다이나믹 프로그래밍

추천 풀이 시간: 40분

쉬운 계단 수: https://www.acmicpc.net/problem/10844

JavaScript 동적 계획법
다이나믹 프로그래밍

문제 해결 아이디어



쉬운 계단 수: https://www.acmicpc.net/problem/10844

JavaScript 동적 계획법 정답 코드 예시
다이나믹 프로그래밍

```
let file = require('fs').readFileSync('/dev/stdin');
let input = file.toString().split('\n');

let n = Number(input[0]);
let dp = Array.from(Array(n + 1), () => new Array(10).fill(0));

dp[1][0] = 0; // 0으로 시작하는 수는 계단수가 아님
for (let j = 1; j <= 9; j++) {
  dp[1][j] = 1;
}

for (let i = 2; i <= n; i++) { // 점화식에 따라서 DP 테이블 채우기
  for (let j = 0; j <= 9; j++) {
    dp[i][j] = 0;
    if (j > 0) dp[i][j] += dp[i - 1][j - 1];
    if (j < 9) dp[i][j] += dp[i - 1][j + 1];
    dp[i][j] %= Number(1e9);
  }
}

let result = 0;
for (let j = 0; j <= 9; j++) {
  result += dp[n][j];
  result %= Number(1e9);
}
console.log(result);
```

RGB 거리: https://www.acmicpc.net/problem/1149

JavaScript 동적 계획법
다이나믹 프로그래밍

혼자 힘으로 풀어보기

문제 제목: RGB 거리

문제 난이도: ★★★☆☆

문제 유형: 다이나믹 프로그래밍

추천 풀이 시간: 40분

RGB 거리: https://www.acmicpc.net/problem/1149

문제 해결 아이디어

[문제 조건] 인접한 집과 색상이 달라야 한다.

- **점화식**을 도출하기 위해 몇 가지 예시를 고려해 볼 수 있다.

집 1	집 2	집 3
빨강: 26	**빨강**: 49	**빨강**: 13
초록: 40	**초록**: 60	**초록**: 89
파랑: 83	**파랑**: 57	**파랑**: 99

RGB 거리: https://www.acmicpc.net/problem/1149

문제 해결 아이디어

[문제 조건] 인접한 집과 색상이 달라야 한다.

- **점화식**을 도출하기 위해 몇 가지 예시를 고려해 볼 수 있다.

집 1	집 2	집 3
빨강: 26	**빨강**: 49	**빨강**: 13
초록: 40	**초록**: 60	**초록**: 89
파랑: 83	**파랑**: 57	**파랑**: 99

RGB 거리: https://www.acmicpc.net/problem/1149

문제 해결 아이디어

- 앞에서부터 차례대로 집을 확인한다.
- 현재 집 a_n에서 j라는 색을 사용한다면?

→ 앞집 a_{n-1}에서는 j라는 색을 사용하지 않는 조건에 한하여 최적의 해(최솟값)을 고려한다.

```
// 요구사항: 인접한 집이 동일한 색을 안 쓸 때, 색칠하는 모든 경우의 수 계산
for (let i = 1; i < n; i++) { // 집을 하나씩 확인하며
  for (let j = 0; j < 3; j++) { // j번째 색을 사용할 때의 최솟값은?
    for (let k = 0; k < 3; k++) { // 앞집에서 k번째 색을 쓴다면
      if (j != k) d[i][j] = Math.min(d[i][j], arr[i][j] + d[i - 1][k]);
    }
  }
}
```

RGB 거리: https://www.acmicpc.net/problem/1149

JavaScript 동적 계획법 정답 코드 예시
다이나믹 프로그래밍

```
// readline 모듈보다는 fs를 이용해 파일 전체를 읽기
let file = require('fs').readFileSync('/dev/stdin');
let input = file.toString().split('\n');

let n = Number(input[0]); // 집의 개수

arr = [];
d = [];
for (let i = 0; i < n; i++) {
  let [r, g, b] = input[i + 1].split(' ').map(Number);
  // 가능한 최댓값으로 초기화
  d.push(new Array(3).fill(1000000));
  arr.push([r, g, b]);
}
```

RGB 거리: https://www.acmicpc.net/problem/1149

JavaScript 동적 계획법 정답 코드 예시
다이나믹 프로그래밍

```
// 첫 번째 집은 그대로 최솟값으로 기록
d[0][0] = arr[0][0];
d[0][1] = arr[0][1];
d[0][2] = arr[0][2];

// 요구사항: 인접한 집이 동일한 색을 안 쓸 때, 색칠하는 모든 경우의 수 계산
for (let i = 1; i < n; i++) { // 집을 하나씩 확인하며
  for (let j = 0; j < 3; j++) { // j번째 색을 사용할 때의 최솟값은?
    for (let k = 0; k < 3; k++) { // 앞집에서 k번째 색을 쓴다면
      if (j != k) d[i][j] = Math.min(d[i][j], arr[i][j] + d[i - 1][k]);
    }
  }
}
console.log(Math.min(d[n - 1][0], d[n - 1][1], d[n - 1][2]));
```