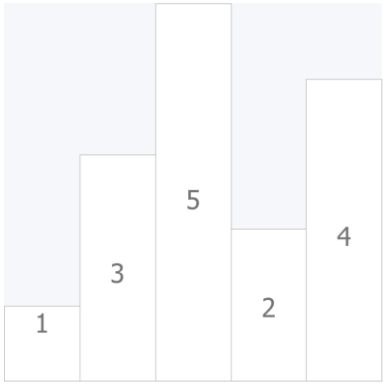
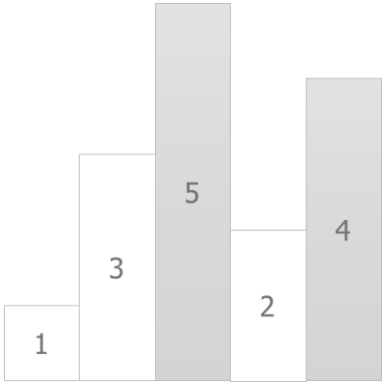
고층 빌딩

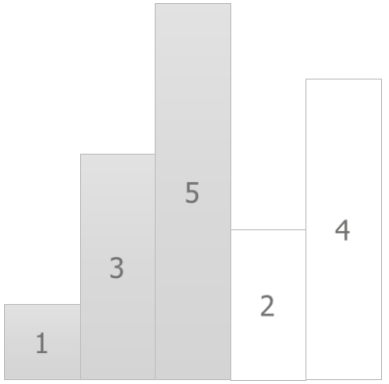
https://www.acmicpc.net/problem/1328

N = 5, L = 3, R = 2 인 경우 가능한 배치





오른쪽에서 봤을 때 2개



왼쪽에서 봤을 때 3개

dp[N][L][R] = 높이가 1 ~ N인 빌딩 N개, 왼쪽에서 L개 보이고, 오른쪽에서 R개 보이는 빌딩 배치의 개수

빌딩이 2 ~ N까지 이미 세워져 있고, 여기에 높이가 1인 빌딩을 추가하는 방식으로 문제를 푼다.

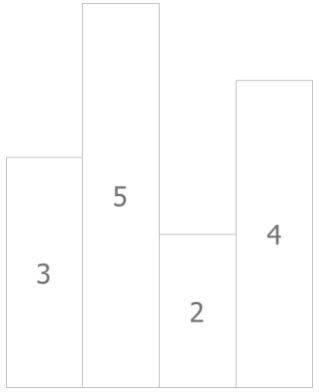
① 빌딩 2 ~ N까지 모두 세워져 있다.

② 여기에 높이가 1인 빌딩을 추가한다.

③ 2 ~ N 까지 모두 세워져 있을 때, 빌딩을 추가하는 방법의 수 N개

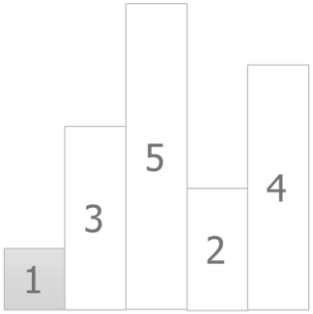
고층 빌딩

https://www.acmicpc.net/problem/1328



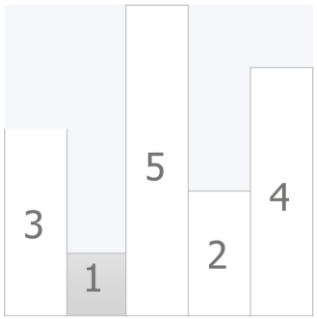
2~5 까지 빌딩이 모두 있을 때, 높이가 1인 빌딩을 추가하는 방법

- 빌딩은 3, 5, 2, 4로 세워져 있다고 가정한다.
- 왼쪽에서 2개, 오른쪽에서 2개가 보인다.
- N = 4, L = 2, R = 2



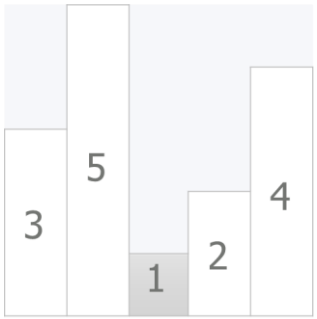
N = 5, L = 3, R = 2

N+=1, L+=1



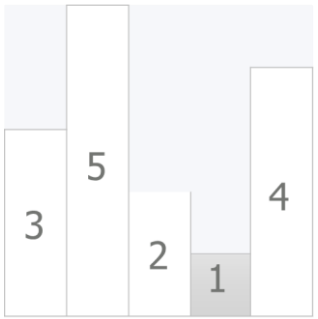
N = 5, L = 2, R = 2

N+=1



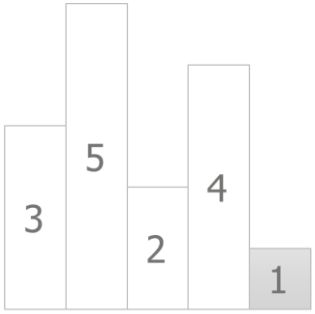
N = 5, L = 2, R = 2

N+=1



N = 5, L = 2, R = 2

N+=1



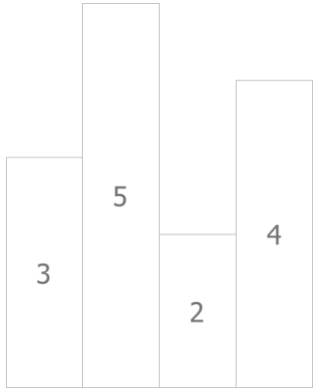
N = 5, L = 2, R = 3

N+=1, R+=1

- 가운데 끼워넣는 경우는 L과 R이 변하지 않는다.
- 가장 앞에 넣는 경우는 왼쪽에서 보이는 것이 하나 증가하고 가장 뒤는 오른쪽 증가

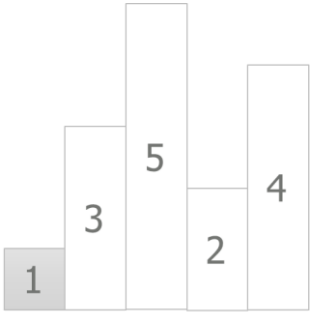
고층 빌딩

https://www.acmicpc.net/problem/1328



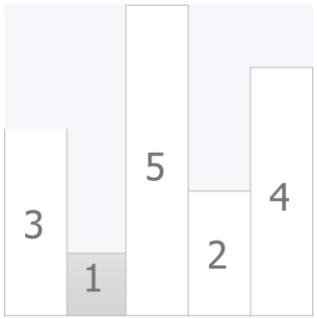
2~5 까지 빌딩이 모두 있을 때, 높이가 1인 빌딩을 추가하는 방법

- 빌딩은 3, 5, 2, 4로 세워져 있다고 가정한다.
- 왼쪽에서 2개, 오른쪽에서 2개가 보인다.
- N = 4, L = 2, R = 2



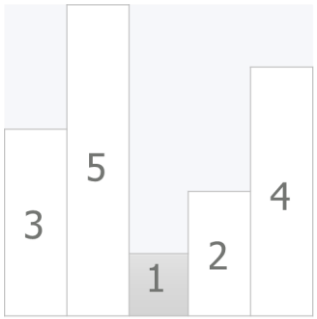
N = 5, L = 3, R = 2

N+=1, L+=1



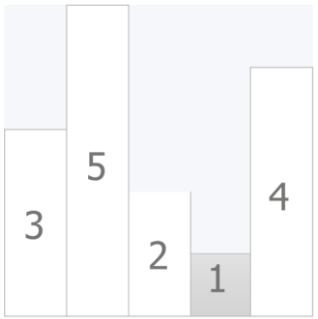
N = 5, L = 2, R = 2

N+=1



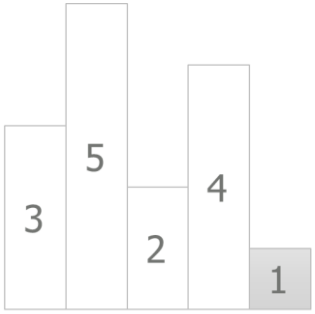
N = 5, L = 2, R = 2

N+=1



N = 5, L = 2, R = 2

N+=1



N = 5, L = 2, R = 3

N+=1, R+=1

- 가운데 끼워넣는 경우는 L과 R이 변하지 않는다.
- 가장 앞에 넣는 경우는 왼쪽에서 보이는 것이 하나 증가하고 가장 뒤는 오른쪽 증가

고층 빌딩

https://www.acmicpc.net/problem/1328

dp[N][L][R] = 빌딩 N개, 왼쪽에서 L개 보임, 오른쪽에서 R개 보일 때 빌딩 배치의 개수

① 가장 왼쪽에 빌딩 1이 있는 경우
- L이 하나 증가해야 한다.
- dp[N][L][R] += dp[N-1][L-1][R]

② 가장 오른쪽에 빌딩 1이 있는 경우
- R이 하나 증가해야 한다.
- dp[N][L][R] += dp[N-1][L][R-1]

③ 사이에 빌딩 1이 있는 경우
- 추가할 수 있는 경우가 N – 2개 존재
- dp[N-1][L][R] * (N-2)

∴ dp[N][L][R] = dp[N-1][L-1][R] + dp[N-1][L][R-1] + dp[N-1][L][R]*(N-2)

고층 빌딩

https://www.acmicpc.net/problem/1328

```
ll mod = 1000000007;
ll solve(int n, int l, int r) {
    if (n == 1 && l == 1 && r == 1) return 1LL;
    if (n == 0 || l == 0 || r == 0) return 0;

    if (dp[n][l][r] != -1) return dp[n][l][r];

    dp[n][l][r] = 0;
    dp[n][l][r] += solve(n - 1, l - 1, r) + solve(n - 1, l, r - 1) + solve(n - 1, l, r)*(n - 2);
    dp[n][l][r] %= mod;

    return dp[n][l][r];
}
```

```
dp[1][1][1] = 1LL;
for (int i = 2; i <= N; ++i) {
    for (int j = 1; j <= L; ++j) {
        for (int k = 1; k <= R; ++k) {
            dp[i][j][k] = dp[i - 1][j - 1][k] + dp[i - 1][j][k - 1]
                        + dp[i - 1][j][k] * (i - 2);
            dp[i][j][k] %= mod;
        }
    }
}
```