<https://programmers.co.kr/learn/courses/30/lessons/64061>

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34 | package programmers.c30.e64061;  public class Main1 {  static class Solution {  public int solution(int[][] board, int[] moves) {  int[] stack = new int[board.length \* board.length];  int top = 0, count = 0;  for (int 열 : moves) {  --열;  for (int 행 = 0; 행 < board.length; ++행) {  if (board[행][열] != 0) {  if (stack[top] != board[행][열])  stack[++top] = board[행][열];  else {  --top;  count += 2;  }  board[행][열] = 0;  break;  }  }  }  return count;  }  }  public static void main(String[] args) {  int[][] board = {{0,0,0,0,0}, {0,0,1,0,3}, {0,2,5,0,1}, {4,2,4,4,2}, {3,5,1,3,1}};  int[] moves = {1,5,3,5,1,2,1,4};  Solution sol = new Solution();  System.out.println(sol.solution(board, moves));  }  } |

|  |
| --- |
| public int solution(int[][] board, int[] moves) {  int[] stack = new int[board.length \* board.length]; // 바구니. 바구니 최대 크기는 게임판 크기  int top = 0, count = 0; // top: 바구니의 top index, count: 제거된 인형 수  for (int 열 : moves) {  --열; // 열의 값 1..N을 0..N-1로 수정  for (int 행 = 0; 행 < board.length; ++행) {  if (board[행][열] != 0) { // 그 열의 가장 위 인형을 찾는다  if (stack[top] != board[행][열]) // 그 인형이 바구니 인형과 다르다면  stack[++top] = board[행][열]; // 바구니에 쌓는다  else { // 바구니 인형과 같다면  --top; // 바구니 상단 인형 제거  count += 2; // 제거된 인형의 수 +2  }  board[행][열] = 0; // 게임판에서 인형 제거  break; // 현재 열의 인형을 제거했으므로, 다음 열로 넘어간다.  }  }  }  return count;  } |