8/22 - 3

최백준 choi@startlink.io

0과 1

https://www.acmicpc.net/problem/8111

- 자연수 N이 주어졌을 때 N의 배수 중에서 다음 조건을 만족하는 수를 찾는 문제 (N ≤ 20,000)

1. 0과 1로만 이루어져 있다
2. 1이 적어도 하나 있다
3. 수의 길이가 100 이하이다
4. 수가 0으로 시작하지 않는다

0과 1

https://www.acmicpc.net/problem/8111

- 0과 1로만 이루어져 있으면서
- 길이가 1인 수: 1
- 길이가 2인 수: 10, 11
- 길이가 3인 수: 100, 101, 110, 111
- 길이가 4인 수: 1000, 1001, 1010, 1011, 1100, 1101, 1110, 1111

- 길이가 k인 수는 총 2^k개가 존재한다.

0과 1

https://www.acmicpc.net/problem/8111

- N의 배수를 구하는 것이기 때문에
- 실제로 그 수가 무엇인지 아는 것 보다는 그 수를 N으로 나눈 나머지가 몇 인지 아는 것이 중요

0과 1

https://www.acmicpc.net/problem/8111

- 0과 1로만 이루어져 있으면서
- 길이가 1인 수: 1 (= 1%17)
- 길이가 2인 수: 10 (=(1×10+0)%17 = 10), 11 (=(1×10+1)%17 = 11)
- 길이가 3인 수: 100 (=(10×10+0)%17 = 15), 101 (=(10×10+1)%17 = 16), 110 (=(11×10+0)%17 = 8), 111 (=(11×10+1)%17 = 9)
- 길이가 4인 수: 1000, 1001, 1010, 1011, 1100, 1101, 1110, 1111

- 0과 1로 이루어져 있는 수 중에서 N으로 나눈 나머지는 총 N개 존재한다.

0과 1

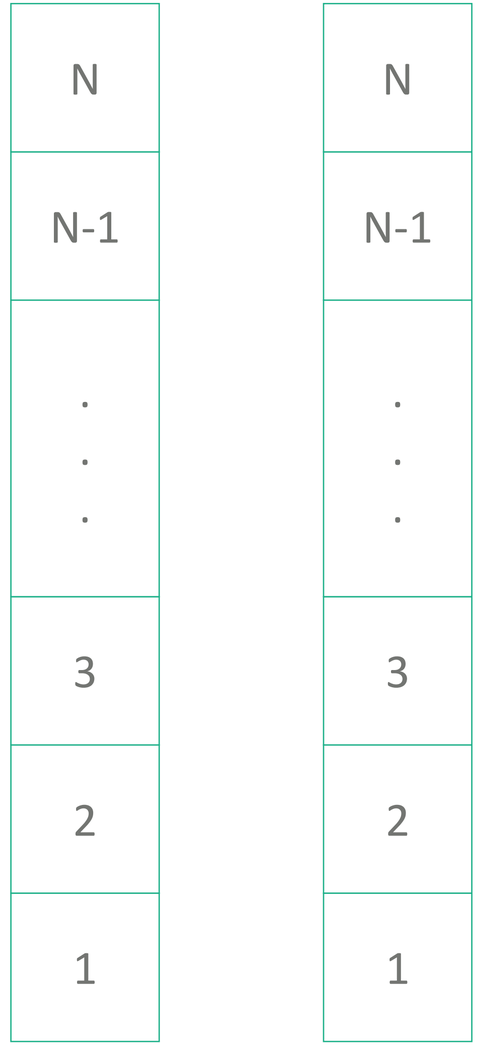
https://www.acmicpc.net/problem/8111

- 소스: http://codeplus.codes/e845e4e89d34454bbbaa627a8a9e1ffe

점프 게임

https://www.acmicpc.net/problem/15558

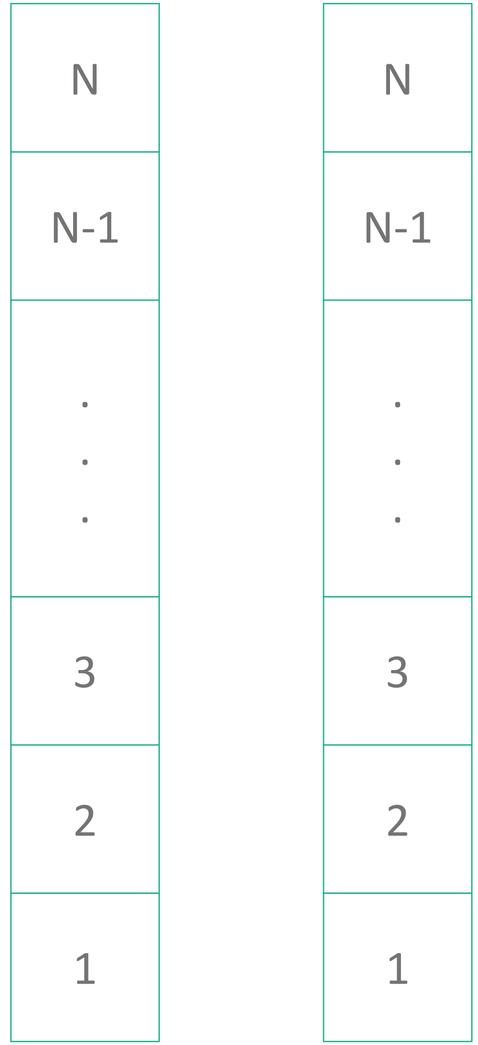
- 오른쪽 그림과 같은 지도가 있다 (N ≤ 100,000)
- 유저가 할 수 있는 행동은 아래 3가지 중 하나이다
- 한 칸 위로, 한 칸 아래로, 옆 칸으로 (+k만큼 이동)
- i초에 i번 칸이 사라진다.
- N번 칸을 넘어갈 수 있는지 구하는 문제



점프 게임

https://www.acmicpc.net/problem/15558

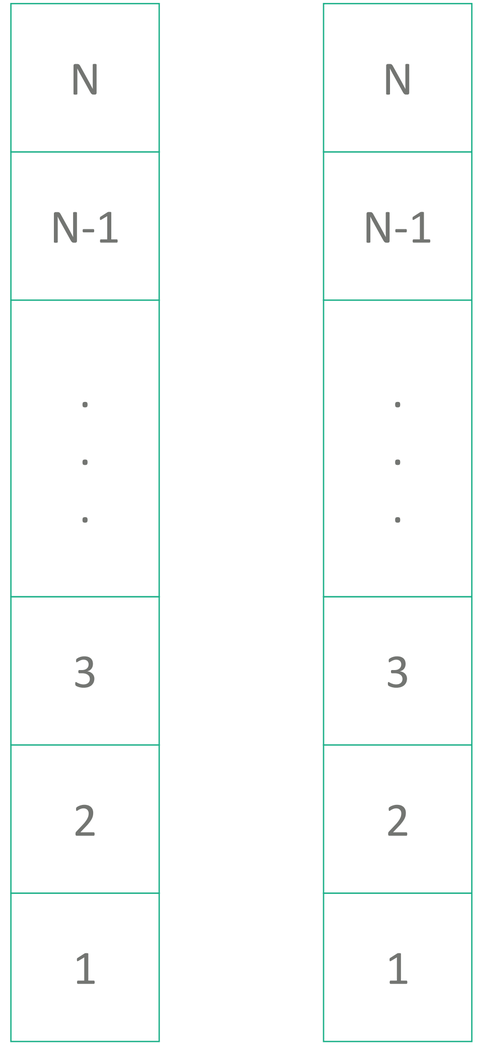
- 만약, 칸이 사라지는 조건이 없으면, BFS로 해결할 수 있다.



점프 게임

https://www.acmicpc.net/problem/15558

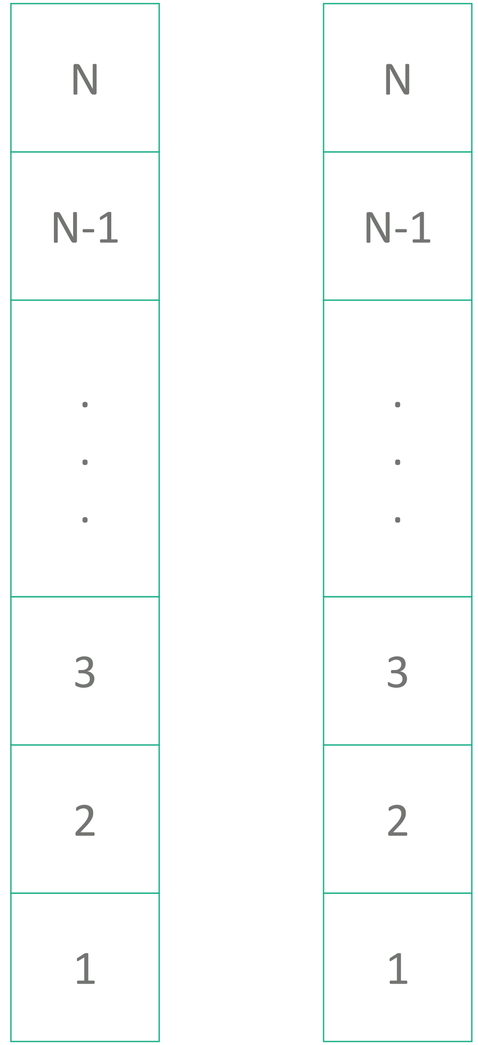
- BFS는 어떤 칸을 방문하는 최단 거리를 구하게 되는데
- i번 칸을 방문한 초 ≥ i 이면 방문할 수 있는 것이다.



점프 게임

https://www.acmicpc.net/problem/15558

- 소스: http://codeplus.codes/77c1b67da6014fbeb9e8160e4c067770



수들의 합 2

https://www.acmicpc.net/problem/2003

- N개의 수로 된 수열 A[1], A[2], …, A[N] 이 있다
- 이 수열의 i번째 수부터 j번째 수까지의 합 A[i]+A[i+1]+…+A[j-1]+A[j]가 M이 되는 경우의 수를 구하는 문제

수들의 합 2

https://www.acmicpc.net/problem/2003

- 이 문제를 풀 수 있는 총 3가지 시간복잡도로 해결할 수 있다.

수들의 합 2

https://www.acmicpc.net/problem/2003

- A[i]+A[i+1]+…+A[j-1]+A[j] == M 이 되는 (i, j) 쌍의 개수를 찾는 문제와 같다.
- i를 정하고, j를 정하고, 합을 계산하면 $O(N^3)$로 계산할 수 있다.

```
for (int i=0; i<n; i++) {
    for (int j=i; j<n; j++) {
        int sum = 0;
        for (int k=i; k<=j; k++) {
            sum += a[k];
        }
        if (sum == m) ans += 1;
    }
}
```

수들의 합 2

https://www.acmicpc.net/problem/2003

- i = a, j = b인 경우에 합을 구한 다음 과정은
- i = a, j = b+1의 합을 구하는 과정이다.
- 그런데, A[a] + A[a+1] + … + A[b]와 A[a] + A[a+1] + … + A[b] + A[b+1]의 차이는 A[b+1]밖에 없다.
- 합은 변하지 않는데 여러 번 구하는 것은 중복된 연산으로 없앨 수 있다.

수들의 합 2

https://www.acmicpc.net/problem/2003

- A[i]+A[i+1]+…+A[j-1]+A[j] == M 이 되는 (i, j) 쌍의 개수를 찾는 문제와 같다.
- 합을 계산할 때, 합을 각각의 i에 대해서 누적하면 $O(N^2)$로 계산할 수 있다.

```
for (int i=0; i<n; i++) {
    int sum = 0;
    for (int j=i; j<n; j++) {
        sum += a[j];
        if (sum == m) ans += 1;
    }
}
```

수들의 합 2

https://www.acmicpc.net/problem/2003

- i = a, j = b의 합이 M보다 작았고, i = a, j = b+1의 합이 M보다 큰 경우를 생각해보자
- 식으로 나타내면 다음과 같다.
  - A[a] + A[a+1] + … + A[b] < M
  - A[a] + A[a+1] + … + A[b+1] > M
- 이 경우 j를 계속 증가시키는 것은 의미가 없기 때문에, i를 증가시켜야 한다.
- 그런데
- i = a+1이고, a ≤ j ≤ b인 경우에서 합이 M이 되는 경우는 있을 수가 없다.
- A[a+1] + … + A[b] == M 이라면 A[a] + A[a+1] + … + A[b] > M 이기 때문에, 위의 조건에 모순이기 때문이다.
- 따라서, 이런 경우는 i만 1증가시키면 된다.

수들의 합 2

https://www.acmicpc.net/problem/2003

- A[i]+A[i+1]+…+A[j-1]+A[j] == M 이 되는 (i, j) 쌍의 개수를 찾는 문제와 같다.
- 합을 계산할 때, 합을 각각의 i에 대해서 누적하면 $O(N^2)$로 계산할 수 있다.

```
for (int i=0; i<n; i++) {
    int sum = 0;
    for (int j=i; j<n; j++) {
        sum += a[j];
        if (sum == m) ans += 1;
    }
}
```

수들의 합 2

https://www.acmicpc.net/problem/2003

- 뒷 페이지의 설명에서 i는 L(왼쪽)로, j는 R(오른쪽)으로 표현했다.

수들의 합 2

https://www.acmicpc.net/problem/2003

- 찾으려고 하는 수: 5
- 합: 1

수들의 합 2

https://www.acmicpc.net/problem/2003

- 찾으려고 하는 수: 5
- 합: 3

수들의 합 2

https://www.acmicpc.net/problem/2003

- 찾으려고 하는 수: 5
- 합: 6

수들의 합 2

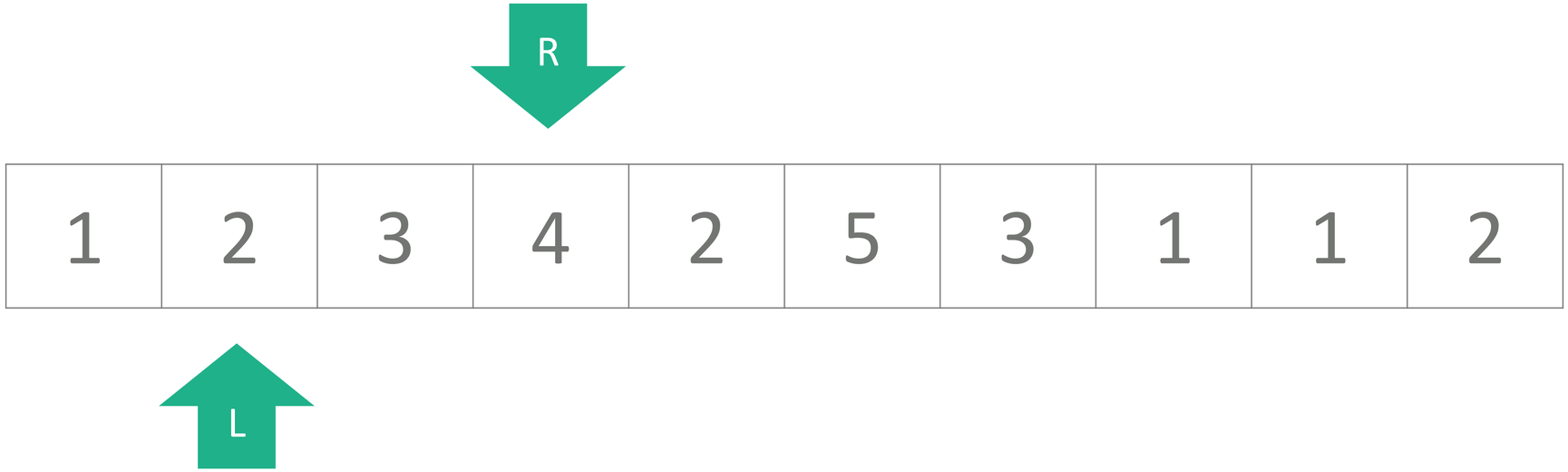
https://www.acmicpc.net/problem/2003

- 찾으려고 하는 수: 5
- 합: 5 (찾았다!)

수들의 합 2

https://www.acmicpc.net/problem/2003

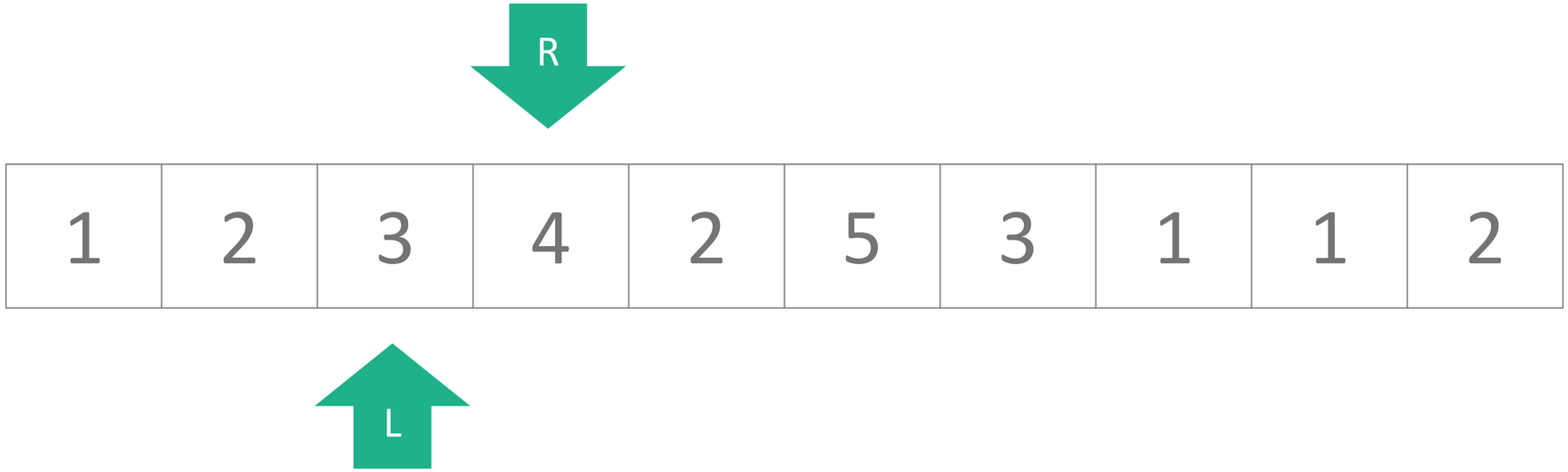
- 찾으려고 하는 수: 5
- 합: 9



수들의 합 2

https://www.acmicpc.net/problem/2003

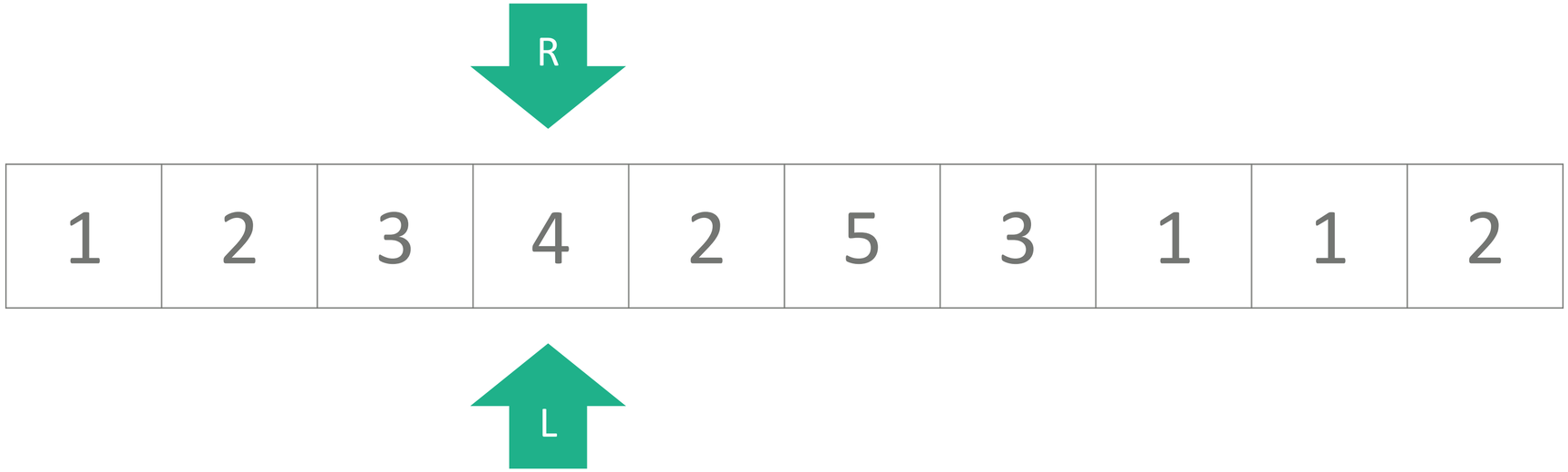
- 찾으려고 하는 수: 5
- 합: 7



수들의 합 2

https://www.acmicpc.net/problem/2003

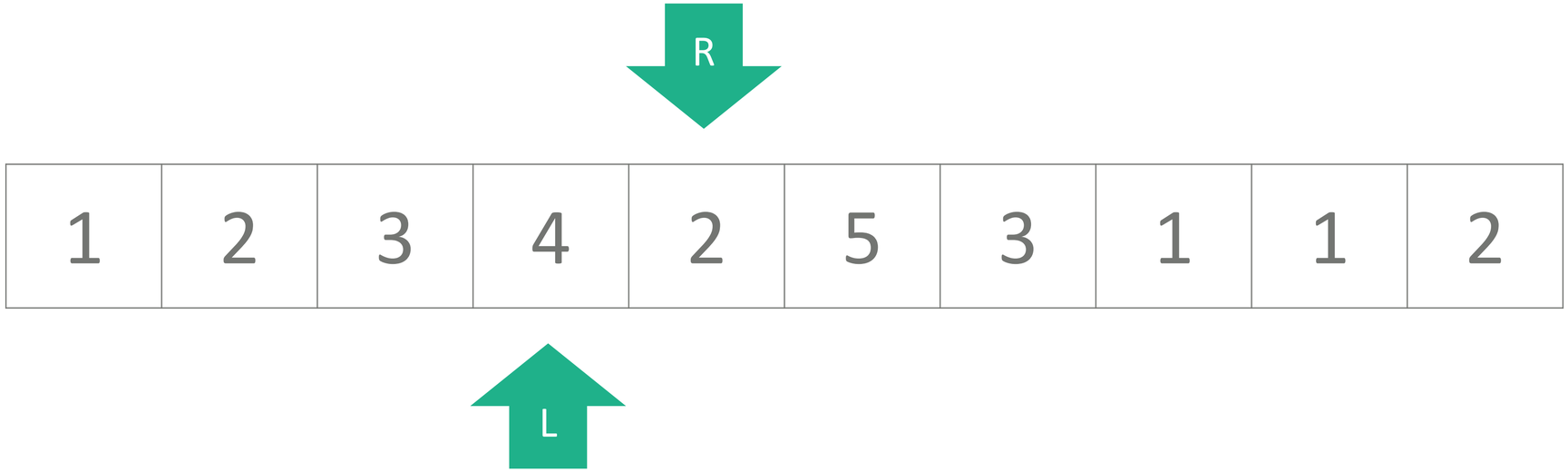
- 찾으려고 하는 수: 5
- 합: 4



수들의 합 2

https://www.acmicpc.net/problem/2003

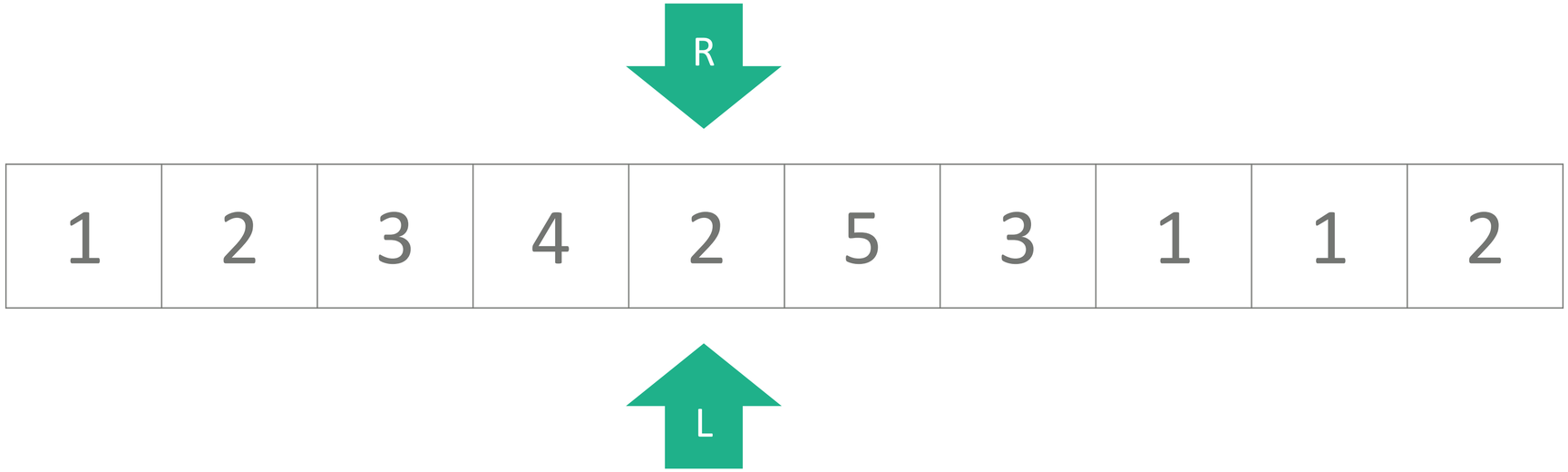
- 찾으려고 하는 수: 5
- 합: 6



수들의 합 2

https://www.acmicpc.net/problem/2003

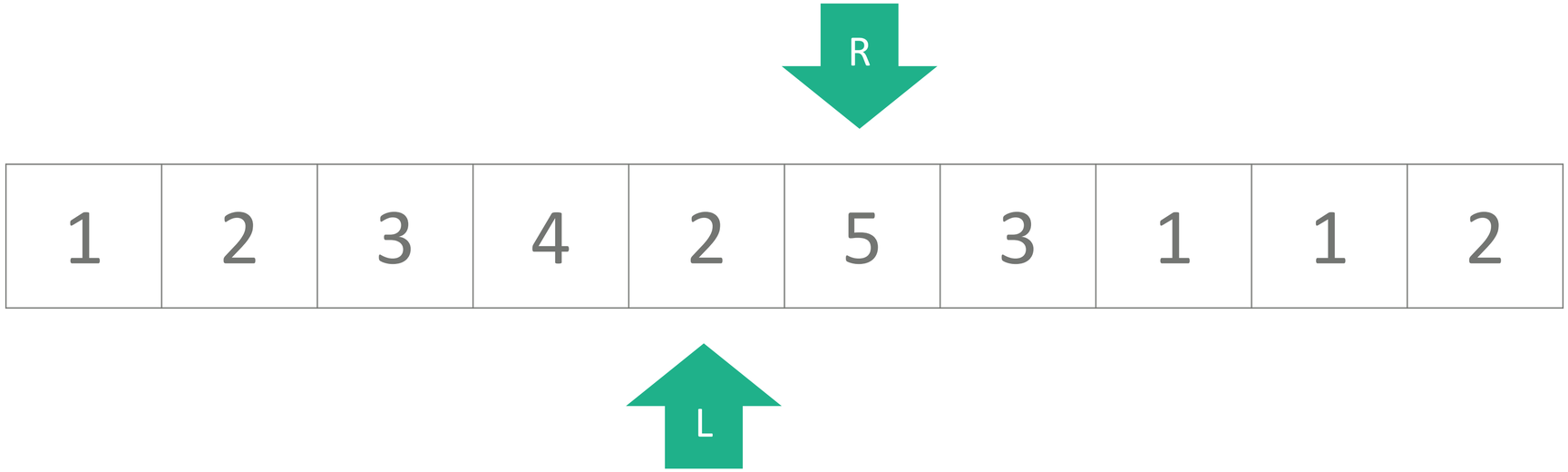
- 찾으려고 하는 수: 5
- 합: 2



수들의 합 2

https://www.acmicpc.net/problem/2003

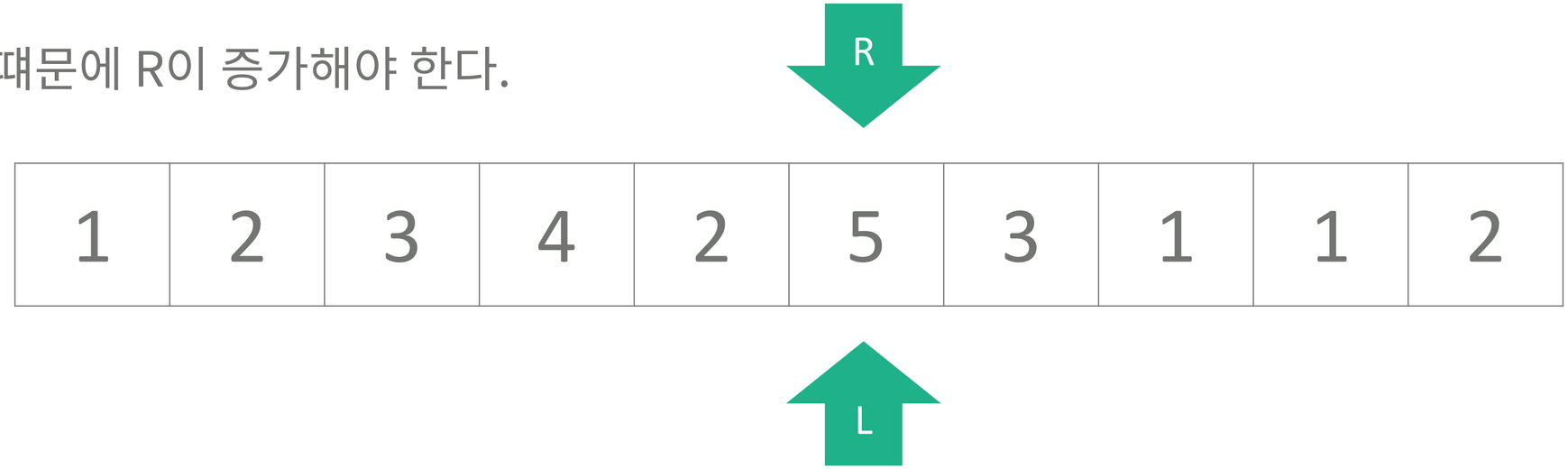
- 찾으려고 하는 수: 5
- 합: 7



수들의 합 2

https://www.acmicpc.net/problem/2003

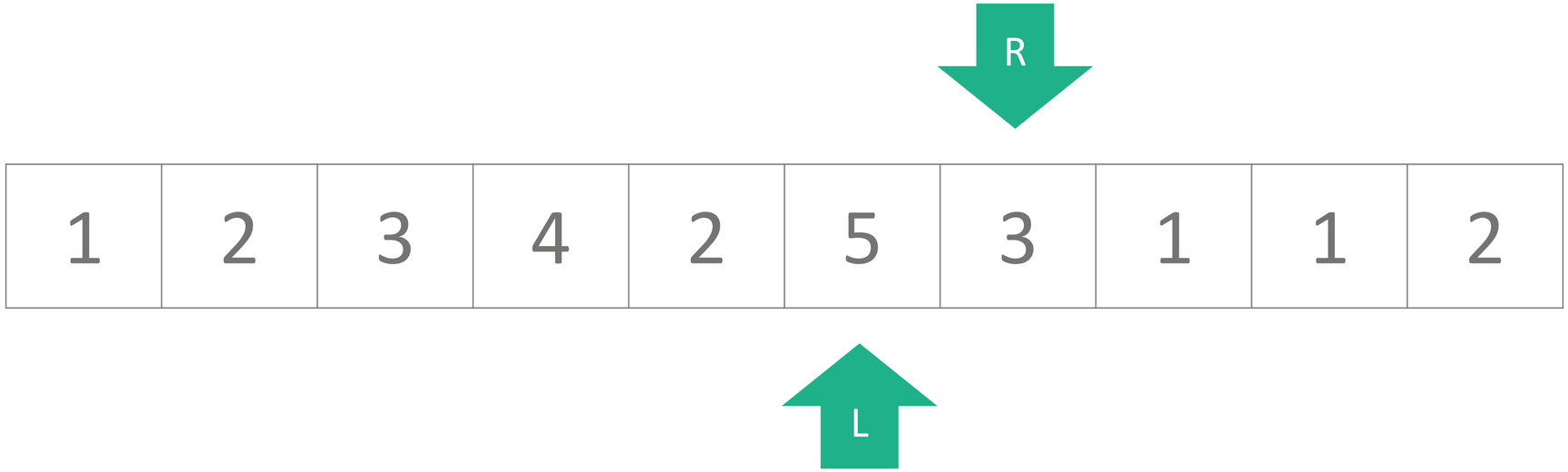
- 찾으려고 하는 수: 5
- 합: 5 (찾았다!)
- 같은 경우에는 L, R 둘 중에 아무거나 증가해도 상관없지만
- 이런 경우 때문에 R이 증가해야 한다.



수들의 합 2

https://www.acmicpc.net/problem/2003

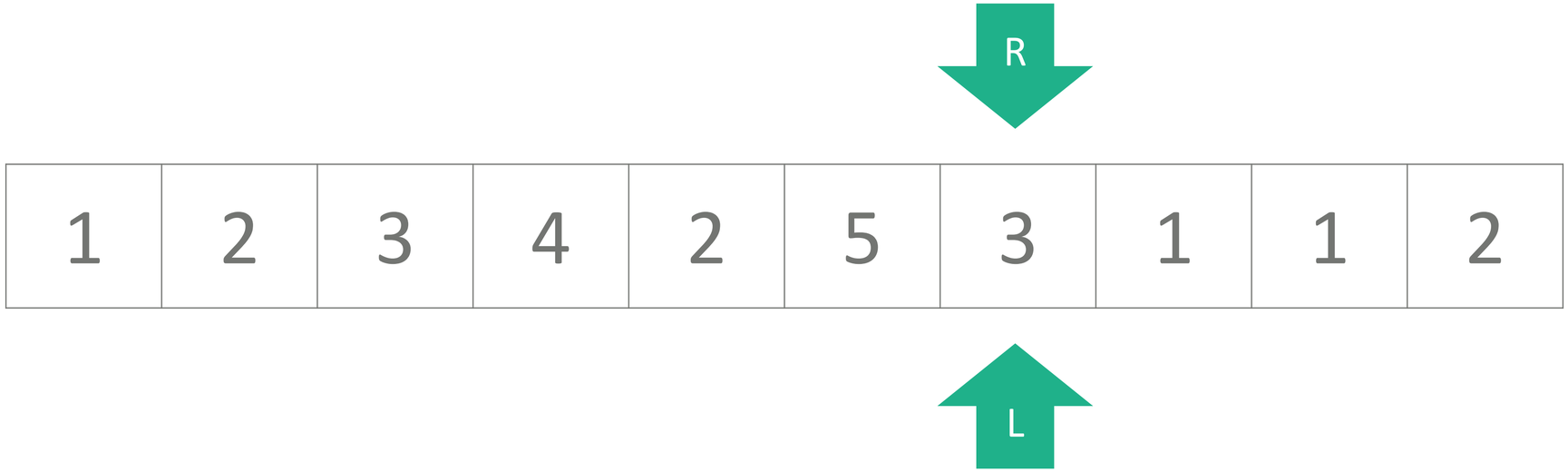
- 찾으려고 하는 수: 5
- 합: 8



수들의 합 2

https://www.acmicpc.net/problem/2003

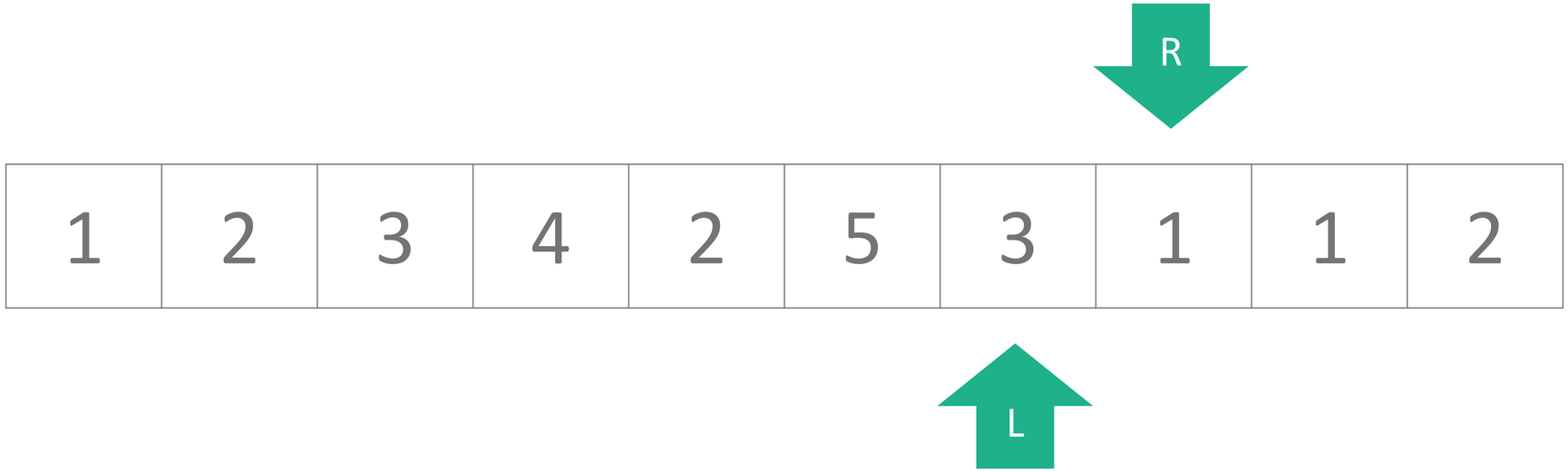
- 찾으려고 하는 수: 5
- 합: 3



수들의 합 2

https://www.acmicpc.net/problem/2003

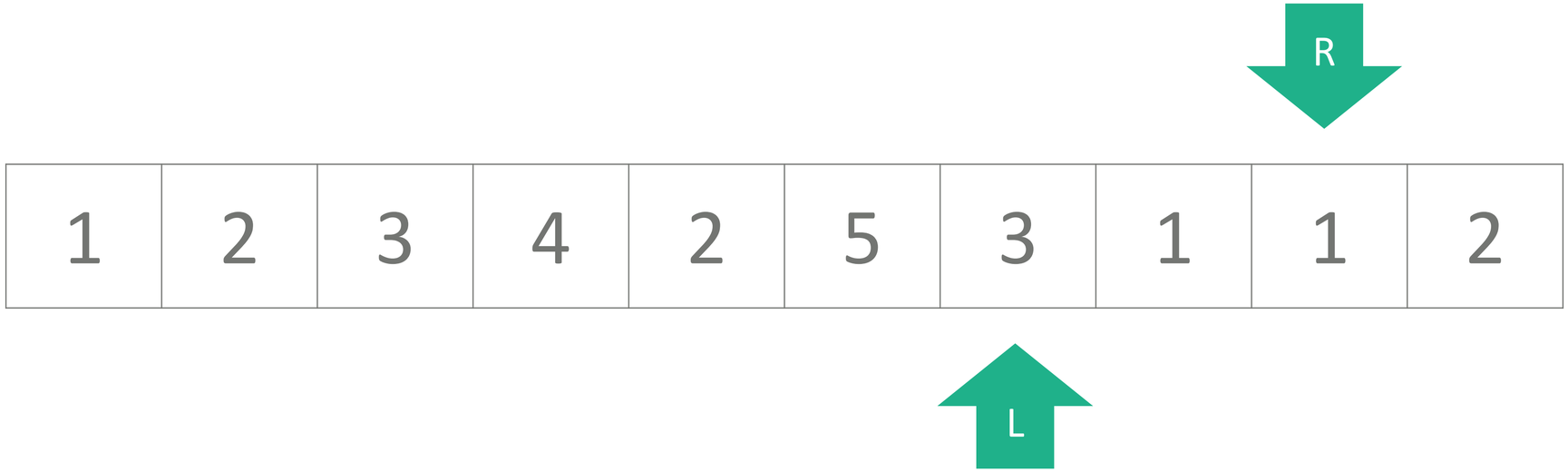
- 찾으려고 하는 수: 5
- 합: 4



수들의 합 2

https://www.acmicpc.net/problem/2003

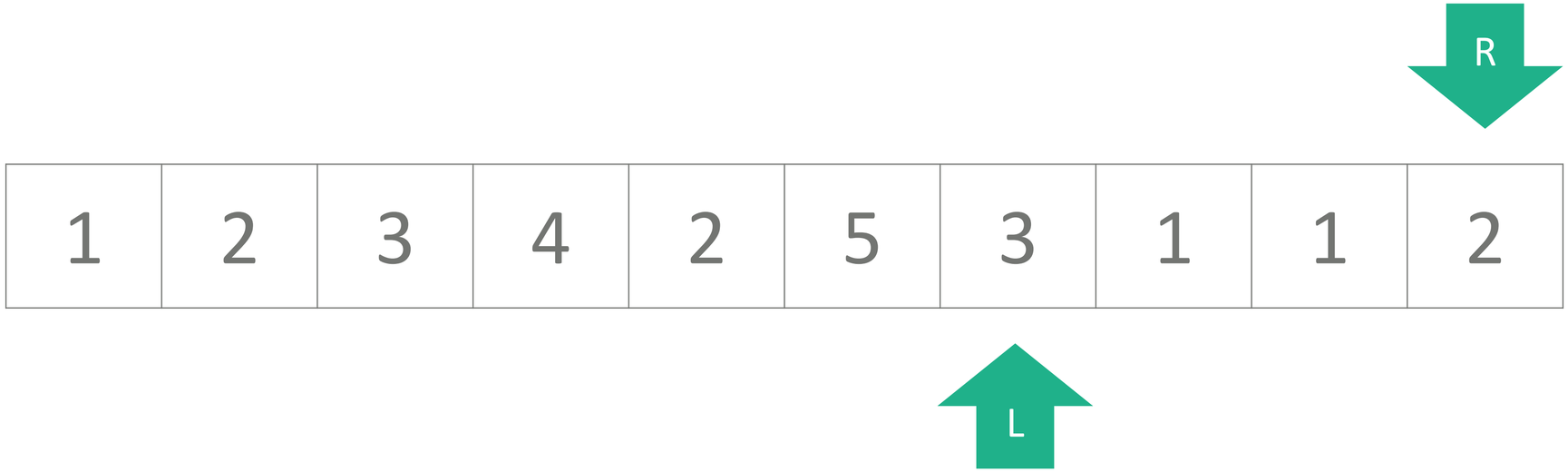
- 찾으려고 하는 수: 5
- 합: 5 (찾았다!)



수들의 합 2

https://www.acmicpc.net/problem/2003

- 찾으려고 하는 수: 5
- 합: 7



수들의 합 2

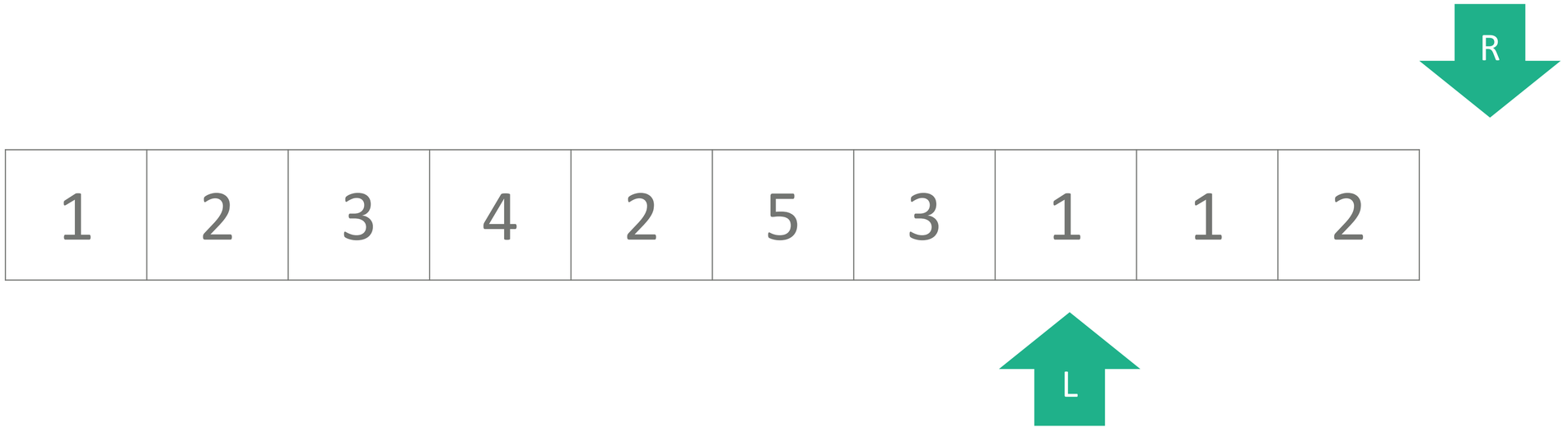
https://www.acmicpc.net/problem/2003

- 찾으려고 하는 수: 5
- 합: 4

수들의 합 2

https://www.acmicpc.net/problem/2003

- 찾으려고 하는 수: 5
- 끝



수들의 합 2

https://www.acmicpc.net/problem/2003

```
int left=0, right=0, sum=a[0], ans = 0;
while (left <= right && right < n) {
    if (sum < m) {
        right += 1;
        sum += a[right];
    } else if (sum == m) {
        ans += 1;
        right += 1;
        sum += a[right];
    } else if (sum > m) {
        sum -= a[left];
        left++;
    }
}
```

수들의 합 2

https://www.acmicpc.net/problem/2003

- 총 시간 복잡도는 L과 R이 $L \leq R$을 유지하면서 끝까지 가기 때문에, O(N) + O(N) = O(N)이다.

수들의 합 2

https://www.acmicpc.net/problem/2003

- 소스: http://codeplus.codes/24059c599d594141b13118b4a72a9ae9