prefix sum && two pointer

2020 Winter 초급

20201546 강효규

prefix sum

✓ 배열의 누적합을 저장해 놓음으로써, 구간합을 $O(1)$에 구하는 테크닉

✓구간 $[0, j]$ 까지 원소의 합을 $pre[j]$ 로 정의하면, 구간 $[i, j]$의 합은 $pre[j] - pre[i-1]$이다.

#11659 구간 합 구하기 4

수 N개가 주어졌을 때, i번째 수부터 j번째 수까지 합을 구하는 프로그램을 작성하시오.

입력

첫째 줄에 수의 개수 N과 합을 구해야 하는 횟수 M이 주어진다. 둘째 줄에는 N개의 수가 주어진다. 수는 1,000보다 작거나 같은 자연수이다. 셋째 줄부터 M개의 줄에는 합을 구해야 하는 구간 i와 j가 주어진다.

#11659 구간 합 구하기 4

✓나이브한 솔루션을 떠올려 보자.

✓각각의 쿼리마다, 구간 $[i, j]$가 주어지면 i부터 j까지의 숫자를 더해준다.

✓이때 각 쿼리마다 $(j - i)$번의 연산이 필요하다.

total time complexity $O(MN)$

#11659 구간 합 구하기 4

prefix sum 을 활용해보자.

✓구간 $[0, j]$ 까지 원소의 합을 $pre[j]$ 로 정의하면, 구간 $[i, j]$의 합은 $pre[j] - pre[i-1]$이다.

val	1	2	3	4	2	5	3	1	1	2
pre	1	3	6	10	12	17	20	21	22	24

$$query\ [4,6] \quad = 4 + 2 + 5 = pre[6] - pre[3] = 17 - 6 = 11$$

#11659 구간 합 구하기 4

✓$prefix\ sum$을 활용해서 각각의 구간합을 $O(1)$에 구하면 $O(M)$으로 AC받을 수 있다.

✓$[1, i]$까지의 누적합을 $pre[i]$로 정의한다고 하였는데, pre를 구현하는 과정을 알아보자.

#11659 구간 합 구하기 4

pre는 $[0, N]$구간을 포함하게 총 $N + 1$개 이상으로 만들어준다.
총 N번 수가 입력될때마다, 누적합이라는 점을 이용하면 $pre[i] = pre[i - 1] + a$가 성립합니다.

```
14      cin >> n >> m;
15      vector<ll>pre(n + 1);
16      for (int i = 1; i <= n; i++) {
17          cin >> a;
18          pre[i] = pre[i - 1] + a;
19      }
```

Total time complexity $O(N)$

#11659 구간 합 구하기 4

```cpp
using namespace std;
using ll = long long;
ll n, m, a, b, c,ans;
int main() {
    ios_base::sync_with_stdio(0);
    cin.tie(0), cout.tie(0);
    cin >> n >> m;
    vector<ll>pre(n + 1);
    for (int i = 1; i <= n; i++) {
        cin >> a;
        pre[i] = pre[i - 1] + a;
    }
    while (m--) {
        cin >> a >> b;
        cout << pre[b] - pre[a - 1] << '\n';
    }
}
```

Accepted

✓ 2차원 배열의 누적합도 생각해 볼 수 있다!

✓이번엔 이차원 배열의 구간합을 $O(1)$에 구해볼 것이다.

#11660 구간 합 구하기 5

N×N개의 수가 N×N 크기의 표에 채워져 있다. (x1, y1)부터 (x2, y2)까지 합을 구하는 프로그램을 작성하시오. (x, y)는 x행 y열을 의미한다.

예를 들어, N = 4이고, 표가 아래와 같이 채워져 있는 경우를 살펴보자.

1	2	3	4
2	3	4	5
3	4	5	6
4	5	6	7

여기서 (2, 2)부터 (3, 4)까지 합을 구하면 3+4+5+4+5+6 = 27이고, (4, 4)부터 (4, 4)까지 합을 구하면 7이다.

표에 채워져 있는 수와 합을 구하는 연산이 주어졌을 때, 이를 처리하는 프로그램을 작성하시오.

#11660 구간 합 구하기 5

✓$pre[i][j]$를 다음과 같이 정의하자.

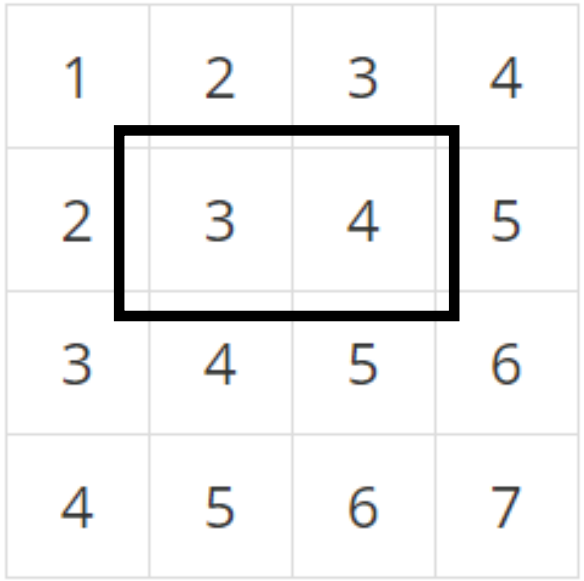
✓ $(0,0)$부터 (i,j)까지 원소를 모두 합친 값.

그렇다면 쿼리 $(a,b)(c,d)$가 입력되면 출력되야 하는 ans는

$$ans = pre[c][d] - pre[a-1][\mathrm{d}] - pre[\mathrm{c}][b-1] + pre[a-1][b-1]$$

를 만족한다.

#11660 구간 합 구하기 5



구간의 합을 구하기 위해서는?

#11660 구간 합 구하기 5

$+Pre[2][3]$

1	2	3	4
2	3	4	5
3	4	5	6
4	5	6	7

우선 $pre[2][3]$구간을 더해준다.

#11660 구간 합 구하기 5

$pre[2][1]$구간을 빼준다.

#11660 구간 합 구하기 5

$-Pre[1][3]$

$pre[1][3]$구간을 빼준다.

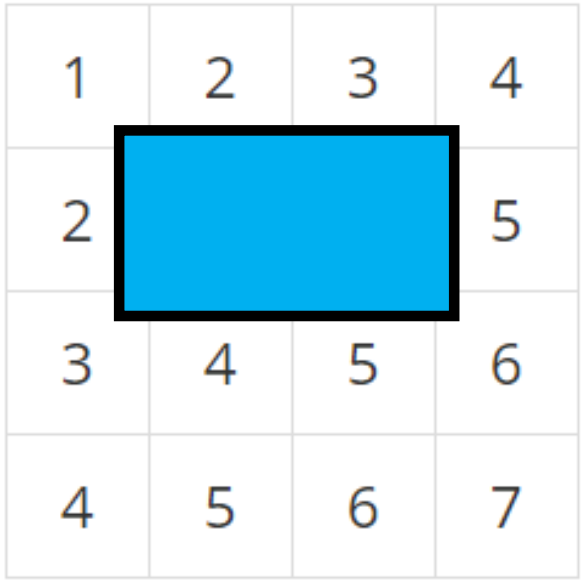
#11660 구간 합 구하기 5

$+Pre[1][1]$

1	2	3	4
2	3	4	5
3	4	5	6
4	5	6	7

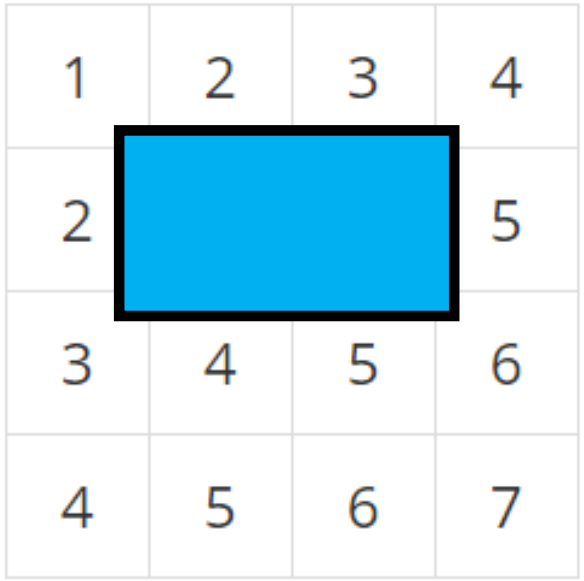
$pre[1][1]$구간이 중복됐으므로 더해준다.

#11660 구간 합 구하기 5



구간의 합만 남게 된다!

#11660 구간 합 구하기 5



$Pre[i][j]$ 2차원 배열을 어떻게 채워 나갈까?

#11660 구간 합 구하기 5

다시 포함 배제의 원리를 이용해보자!

$$pre[i][j] = pre[i][j-1] + pre[i-1][j] - pre[i-1][j-1] + A[i][j];$$

식을 만족한다.
그래서 $pre[i][j]$을 for 반복문을 통해
i우선 기준 그리고 j기준으로 순회하는 방식으로 pre 배열을 채워 나갈 수 있다.

#11660 구간 합 구하기 5

```
15      cin >> n >> m;
16      for(int i=1;i<=n;i++)
17          for (int j = 1; j <= n; j++) {
18              cin >> a;
19              pre[i][j] = pre[i][j - 1] + pre[i - 1][j] - pre[i - 1][j - 1] + a;
20          }
21      while (m--) {
22          cin >> a >> b >> c >> d;
23          cout << pre[c][d] - pre[a - 1][d] - pre[c][b - 1] + pre[a - 1][b - 1] << '\n';
24      }
```

Accepted

two pointer

✓ 배열내에서 각자 다른 원소를 가리키고 있는 2개의 포인트를 조작해가며 문제를 해결하는 방법!

✓ 자료가 1차원으로 주어진 문제에서, 생각해볼 수 있는 테크닉.

#2003 수들의 합 2

문제

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 프로그램을 작성하시오.

입력

첫째 줄에 N(1 ≤ N ≤ 10,000), M(1 ≤ M ≤ 300,000,000)이 주어진다. 다음 줄에는 A[1], A[2], ..., A[N]이 공백으로 분리되어 주어진다. 각각의 A[x]는 30,000을 넘지 않는 자연수이다.

#2003 수들의 합 2

$range\ sum\ ==\ m$ 이 되는 경우의 수를 묻는 문제

몇 가지 경우로 나누어 나이브한 솔루션을 먼저 떠올려 보자

1	2	3	4	2	5	3	1	1	2

#2003 수들의 합 2

Solution 1

$i<j$인 구간 $[i,j]$에 대하여 각각의 sum을 $O(j-i)$에 구하는 방법

```
14      vector<ll>v;
15      int n, m, ans = 0;
16      cin >> n >> m;
17      for (int i = 0; i < n; i++) {
18          cin >> a;
19          v.push_back(a);
20      }
21      for (int i = 0; i < n; i++)
22          for (int j = 0; j < n; j++) {
23              ll temp_sum = 0;
24              for (int k = i; k <= j; k++)temp_sum += v[k];
25              if (temp_sum == m)ans++;
26          }
27      cout << ans;
```

Total time complexity $O(N^3)$

#2003 수들의 합 2

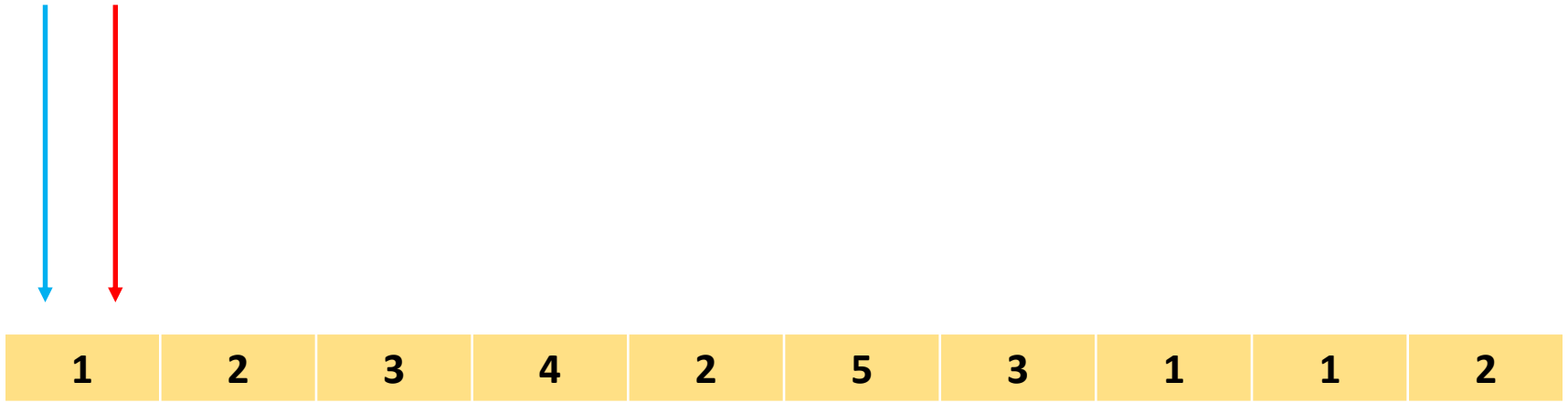
Solution 2

prefix sum을 이용하여, $i<j$인 구간 $[i,j]$에 대하여 각각의 sum을 $O(1)$에 구하는 방법

```
15      int n, m, ans = 0;
16      cin >> n >> m;
17      vector<ll>v, pre(n + 1);
18      for (int i = 1; i <= n; i++) {
19          cin >> a;
20          pre[i] = pre[i - 1] + a;
21          v.push_back(a);
22      }
23      for (int i = 1; i <= n; i++)
24          for (int j = 1; j <= n; j++) {
25              ll temp_sum = pre[j] - pre[i - 1];
26              if (temp_sum == m)ans++;
27          }
28      cout << ans;
```

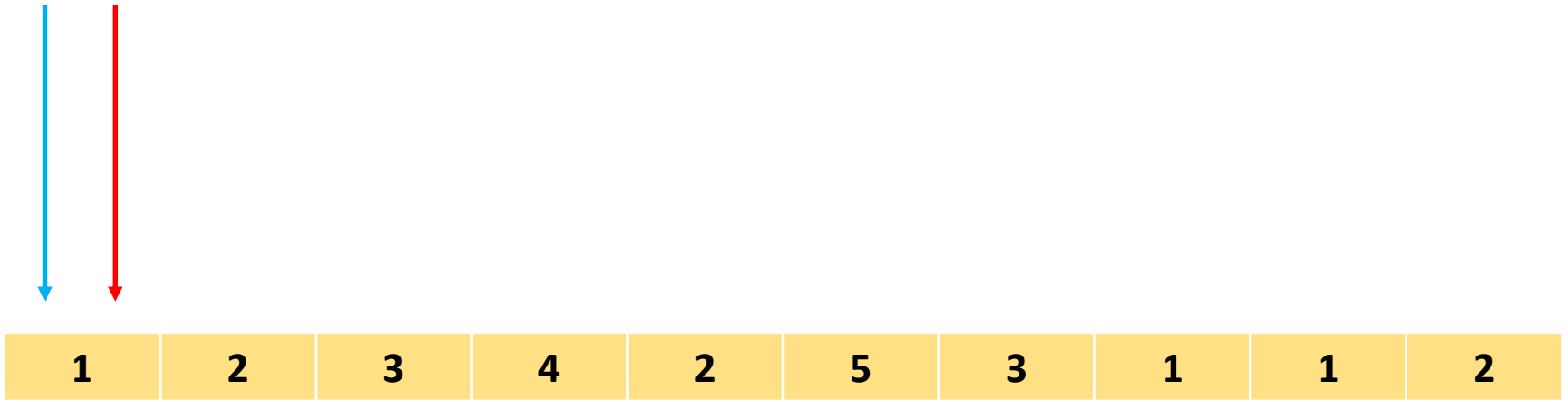
Total time complexity $O(N^2)$

#2003 수들의 합 2



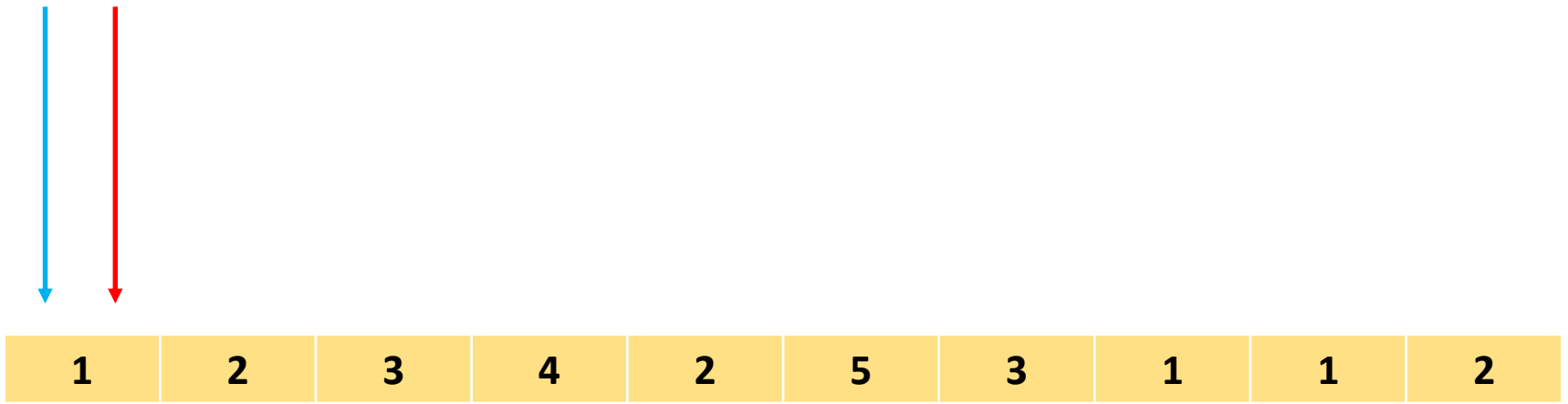
그럼 투 포인터 기법을 활용하여 문제를 어떻게 풀까?

#2003 수들의 합 2



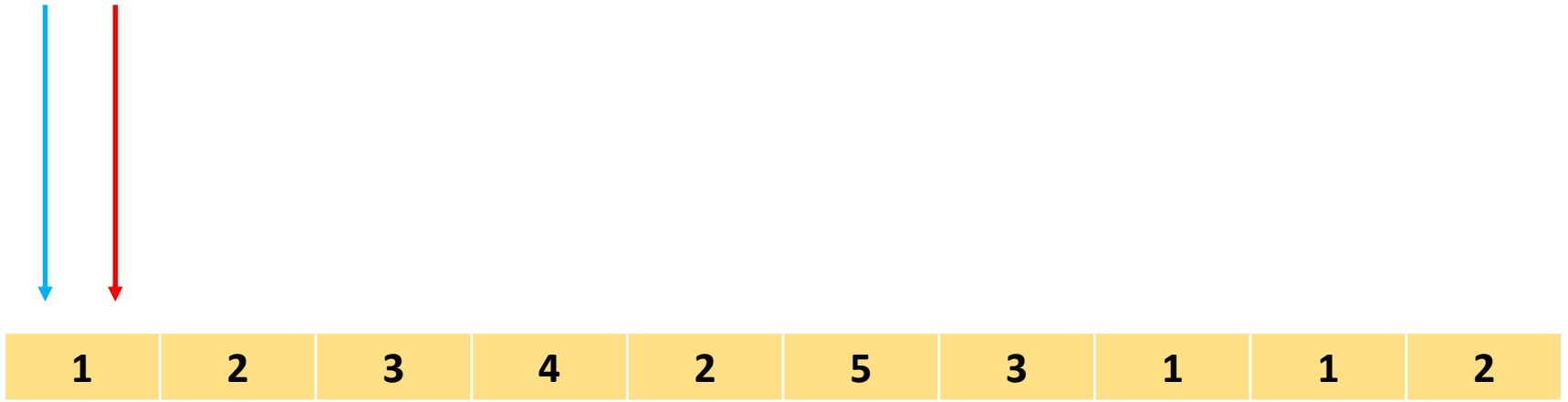
두 개의 변수를 만든다. 각각의 변수는 배열의 위치를 가리키고 있다.
그리고 앞서는 변수를 편의상 a, 뒤의 변수를 편의상 b 라고 하자.

#2003 수들의 합 2



우리는 포인터부터 포인터까지의 합을 구할 것이다.
구간합이 m이라면 구간 한 개를 찾은 것이다.

#2003 수들의 합 2

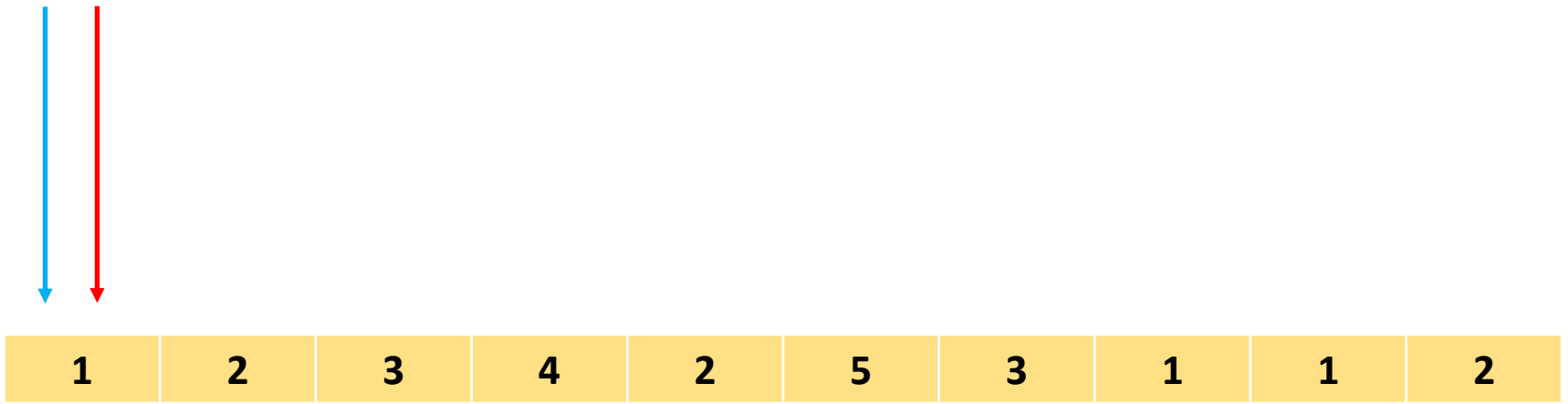


구간합이 찾으려 하는 m 이하라면, b 에 1을 더해준다.
구간합의 값이 m 보다 크다면 a 에 1을 더해준다.
그리고 이를 b가 배열의 범위를 벗어날 때까지 반복해준다.

#2003 수들의 합 2

$m = 5$

ans = 0



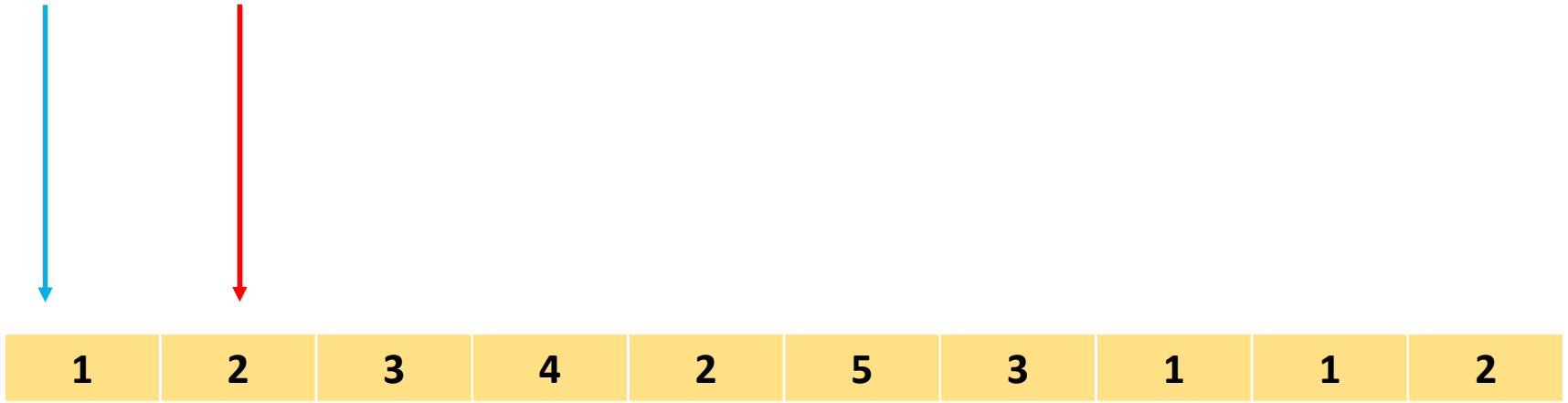
$a = 0, b = 0$

$Range\ sum = 1$

#2003 수들의 합 2

$m = 5$

ans = 0



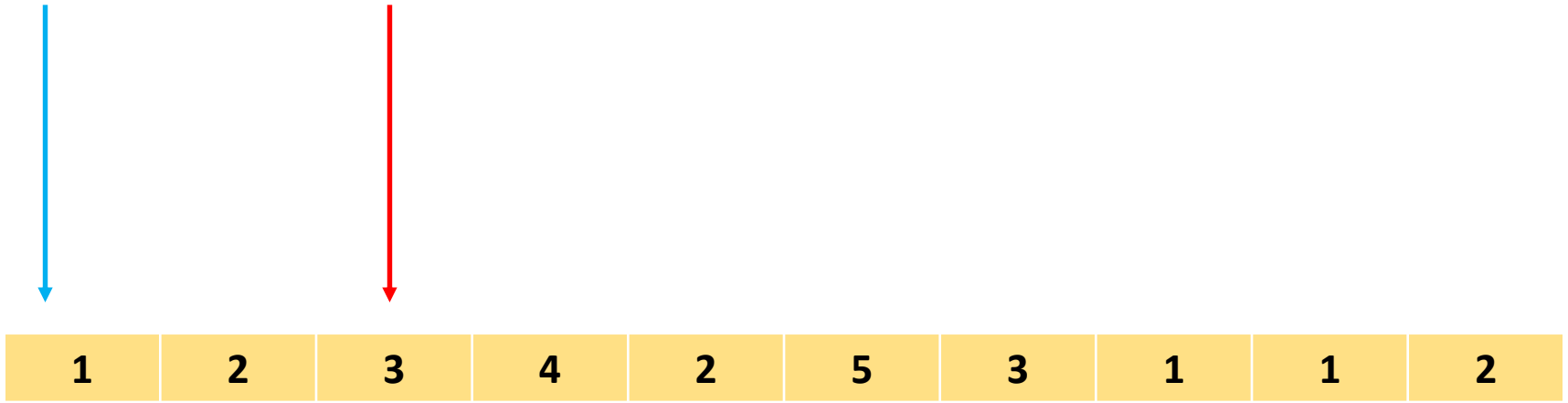
$a = 0, b = 1$

$Range\ sum = 6$

#2003 수들의 합 2

$m = 5$

ans = 0



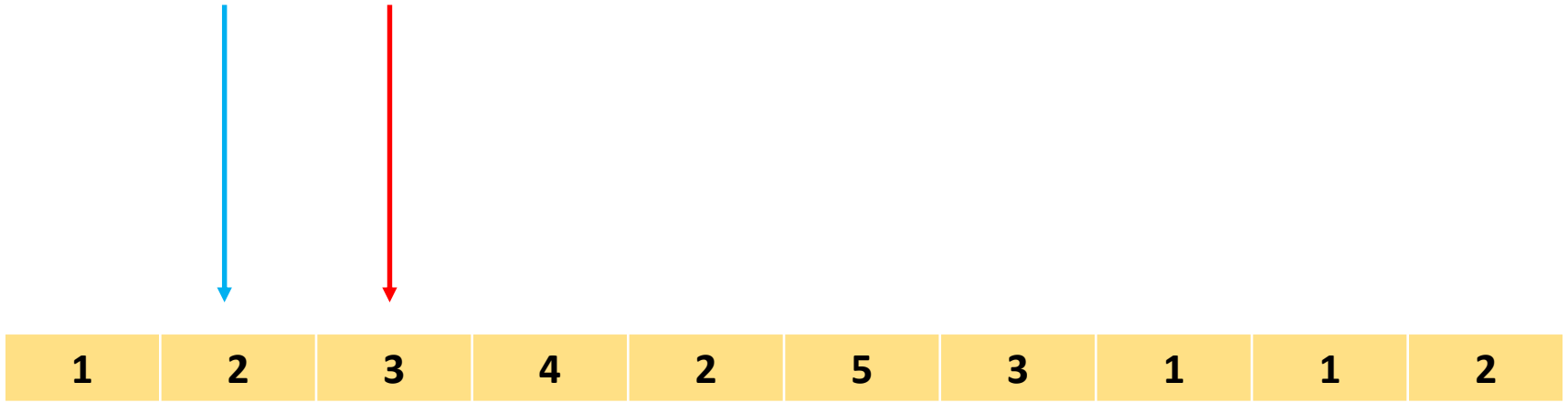
$a = 0, b = 2$

$Range\ sum = 6$

#2003 수들의 합 2

$m = 5$

ans = 1



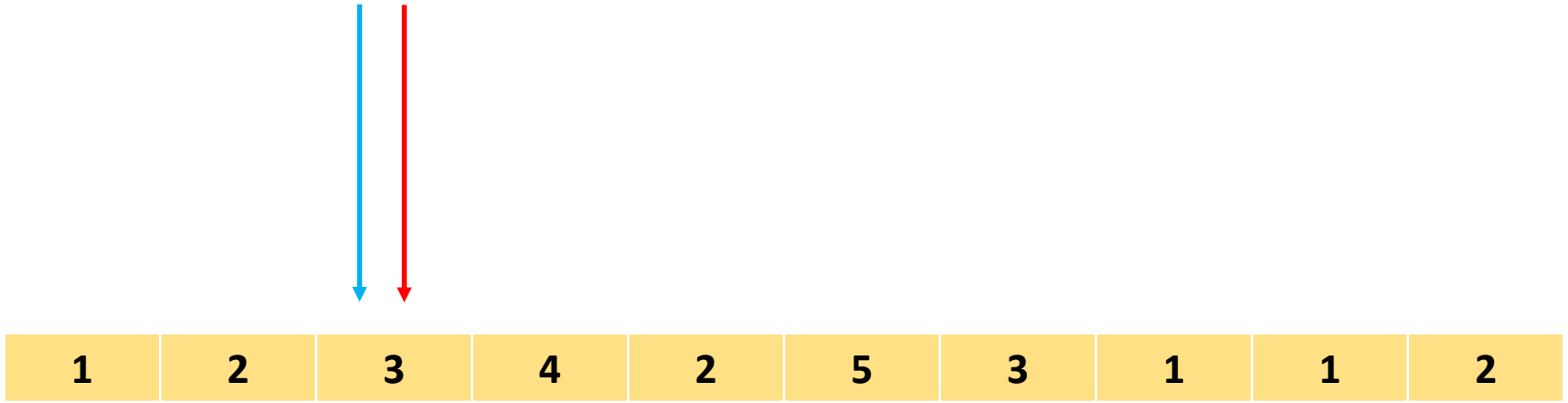
$a = 1, b = 2$

$Range\ sum = 5$

#2003 수들의 합 2

$m = 5$

ans = 1



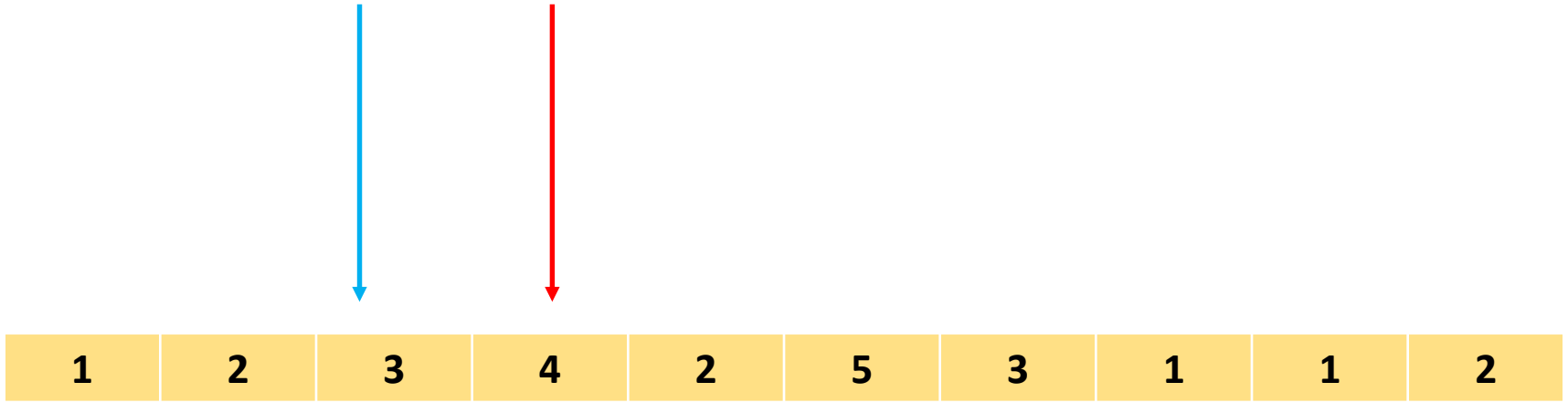
$a = 2, b = 2$

$Range\ sum = 3$

#2003 수들의 합 2

$m = 5$

ans = 1



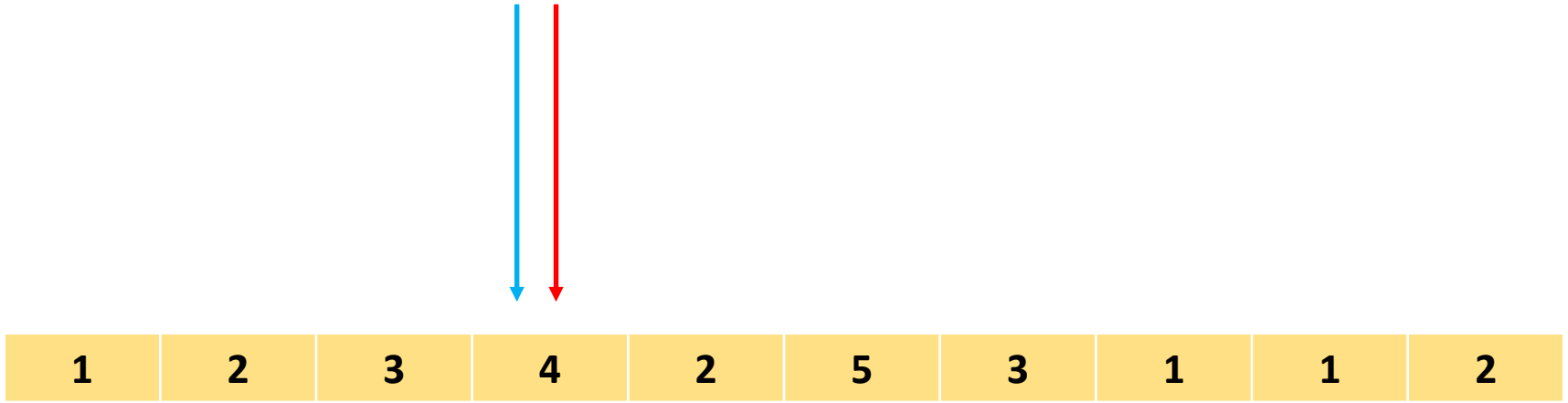
$a = 2, b = 3$

$Range\ sum = 7$

#2003 수들의 합 2

$m = 5$

ans = 1



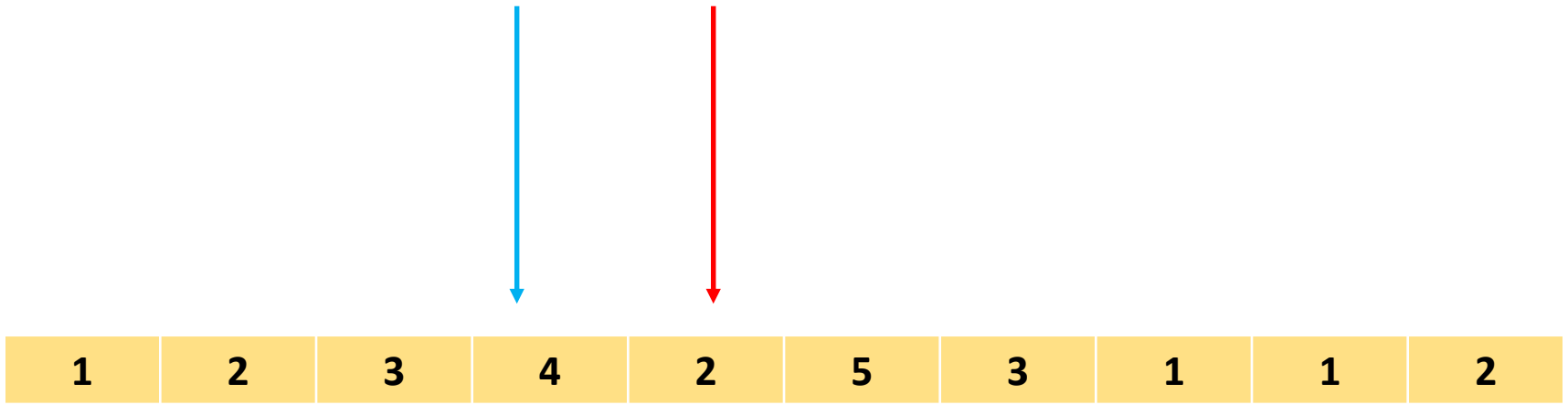
$a = 3, b = 3$

$Range\ sum = 4$

#2003 수들의 합 2

$m = 5$

ans = 1



$a = 3, b = 4$

$Range\ sum = 6$

and so on…

Total time complexity $O(N)$

#2003 수들의 합 2

✓ 정당성 증명

- a 와 b 는 모두 순서대로 지나간다.
- b 는$[0, n-1]$을 모두 최소 한번은 지나간다.

어떤 구간 $[i,j]$ 의 합이 m이라고 가정하자.
증명은 b가 $j-1$인 상태 에서 j 로 넘어갈때 a가 i이하라는 것과 동치가 된다.

✓ 귀류법으로 증명하겠다.

구간 $[x,y]$에 대해 $[x-1,y]$에서 $[x,y]$로 가는 것을 a를 통해 가는 것, $[x,y-1]$에서 $[x,y]$로 가는 것을 b를 통해 가는 것이라고 하자.

#2003 수들의 합 2

a가 i초과라면 $sum[a-1,j] < sum[i,j] = m$이므로 a를 통해 $[a,j]$에 도달 할 순없다.

그렇다면 $[a,j-1]$구간에서 b를 1더해서 도달했다는 것이다.
그런데, $[a,j-1]$구간도 $[a-1,j-1]$이 m보다 작으므로 마찬가지로, b를 통해서 도달해야 한다.

이게 계속 반복되다 보면 구간 $[a,j]$에 도달하기 위해서 b를 통해서만 도달해야 한다.
a는 1이상이다. 따라서 모순이다.

#2003 수들의 합 2

```
14      cin >> n >> m;
15      vector<ll>v;
16      for (int i = 0; i < n; i++) {
17          cin >> a;
18          v.push_back(a);
19      }
20      a = 0;
21      while (1) {
22          if (sum >= m) sum -= v[a++];
23          else if (b  == n) break;
24          else sum += v[b++];
25          if (sum == m) ans++;
26      }
27      cout << ans;
```

Accepted

#1484 다이어트

성원이는 다이어트를 시도중이다. 성원이는 정말 정말 무겁기 때문에, 저울이 부셔졌다.

성원이의 힘겨운 다이어트 시도를 보고만 있던 엔토피아는 성원이에게 새로운 저울을 선물해 주었다.

성원이는 엔토피아가 선물해준 저울 위에 올라갔다. “안돼!! G 킬로그램이나 더 쪘어ㅜㅠ”라고 성원이가 말했다. 여기서 말하는 G킬로그램은 성원이의 현재 몸무게의 제곱에서 성원이가 기억하고 있던 몸무게의 제곱을 뺀 것이다.

성원이의 현재 몸무게로 가능한 것을 모두 출력하는 프로그램을 작성하시오.

입력

첫째 줄에 G가 주어진다. G는 100,000보다 작거나 같은 자연수이다.

#1484 다이어트

투 포인터를 알고 있다면 쉽게 응용할 수 있는 문제.

제곱수 만을 가지고 있는 배열을 만들고,
차이가 G가 되는 경우를 투 포인터로 찾는다고 생각해 보자.

그러면, 또 다시 자연수만을 가지고 있는 증가수열 이니까, 차이가 G이상일때는 a를 더해주고 G미만일때는 b에 1을 더해주는 식으로 투 포인터를 가지고 해결할 수 있다.

#1484 다이어트

```
14      ll a = 0, b = 0, n;
15      vector<ll>v, ans;
16      cin >> n;
17      for (ll i = 1; i <= 200000; i++)
18          v.push_back(i * i);
19      while (1) {
20          if (b >= v.size())break;
21          ll dif = v[b] - v[a];
22          if (dif < n)b++;
23          if (dif > n)a++;
24          if (dif == n) {
25              ans.push_back(sqrt(v[b]));
26              b++;
27          }
28      }
29      if (!ans.size())cout << "-1";
30      for (int i = 0; i < ans.size(); i++)
31          cout << ans[i] << '\n';
32  }
```

Accepted

#추천문제

필수문제

11659	3 구간 합 구하기 4
11660	1 구간 합 구하기 5
2003	3 수들의 합 2
1484	3 다이어트
1253	3 좋다

연습문제

10025	4 게으른 백곰
2559	3 수열
10211	3 Maximum Subarray
20438	2 출석체크
20002	5 사과나무
2143	3 두 배열의 합
20543	2 폭탄 던지는 태영이
1940	4 주몽
15565	1 귀여운 라이언
2470	5 두 용액
2230	5 수 고르기
1806	4 부분합
1644	3 소수의 연속합
7453	2 합이 0인 네 정수