

JavaScript 투 포인터 투 포인터 알고리즘 문제 풀이

투 포인터 알고리즘 문제 풀이 | 코딩 테스트에서 자주 등장하는 투 포인터 이해하기

강사 나동빈

JavaScript

투 포인터

투 포인터 알고리즘 문제 풀이

JavaScript 투 포인터
투 포인터 문제 풀이

혼자 힘으로 풀어보기

JavaScript
투 포인터
투 포인터
문제 풀이

문제 제목: 블로그

문제 난이도: ★☆☆☆☆

문제 유형: 투 포인터, 슬라이딩 윈도우, 누적합

추천 풀이 시간: 30분

JavaScript 투 포인터 투 포인터 문제 풀이

문제 풀이 핵심 아이디어

JavaScript
투 포인터
문제 풀이

- N 일간의 방문자 수가 배열 형태로 주어진다.

[요구 사항] X 일 동안 가장 많이 들어온 방문자 수와 그 기간들을 출력한다.

[예시 1] $N = 5, X = 2$ 인 경우를 생각해 보자.

- $arr = [1, 4, 2, 5, 1]$
- 이때, 정답은 다음과 같다.
- $7 \rightarrow$ 가장 많이 들어온 방문자 수: $arr[2]$ 부터 $arr[3]$ 까지(길이는 $X = 2$)
- $1 \rightarrow$ 가장 많이 들어온 방문자 수가 7인 구간은 1개만 존재

JavaScript 투 포인터 투 포인터 문제 풀이

문제 풀이 핵심 아이디어

JavaScript
투 포인터
문제 풀이

- N 일간의 방문자 수가 배열 형태로 주어진다.

[요구 사항] X 일 동안 가장 많이 들어온 방문자 수와 그 기간들을 출력한다.

[예시 2] $N = 7, X = 5$ 인 경우를 생각해 보자.

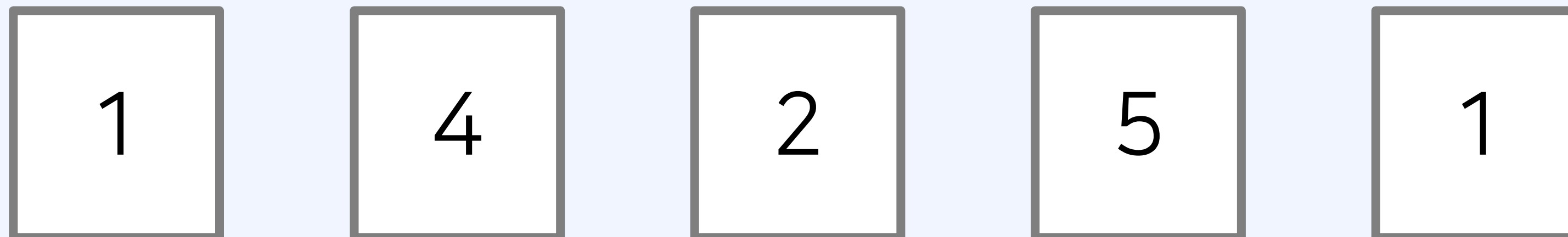
- $arr = [1, 1, 1, 1, 1, 5, 1]$
- 이때, 정답은 다음과 같다.
- $9 \rightarrow$ 가장 많이 들어온 방문자 수: $arr[1]$ 부터 $arr[5]$, $arr[2]$ 부터 $arr[6]$ 두 가지 경우
- $2 \rightarrow$ 가장 많이 들어온 방문자 수가 7인 구간은 2개 존재

JavaScript 투 포인터
투 포인터 문제 풀이

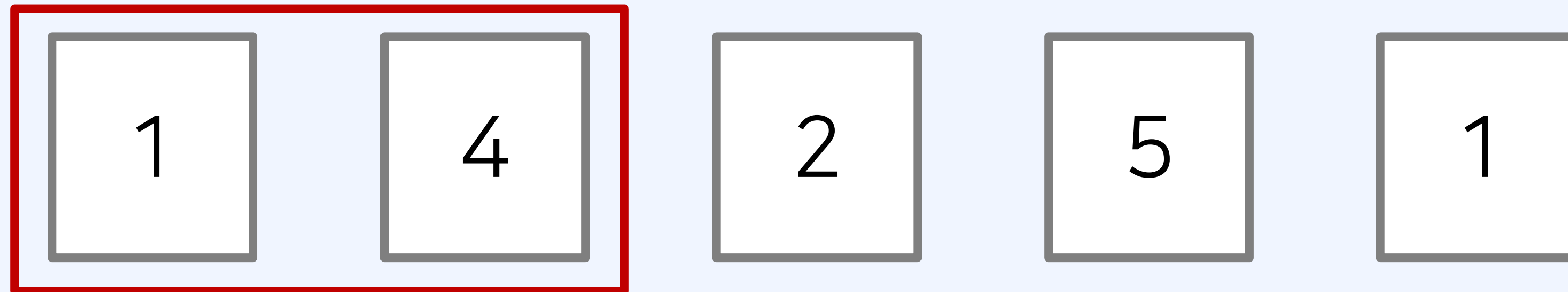
문제 풀이 핵심 아이디어

JavaScript
투 포인터
투 포인터
문제 풀이

- 고정 크기 윈도우를 사용하여 슬라이딩 윈도우를 사용한다.



- 고정 크기 윈도우를 사용하여 슬라이딩 윈도우를 사용한다.
 - 윈도우 크기가 2일 때의 예시는 다음과 같다.



총 합: 5

JavaScript 투 포인터 투 포인터 문제 풀이

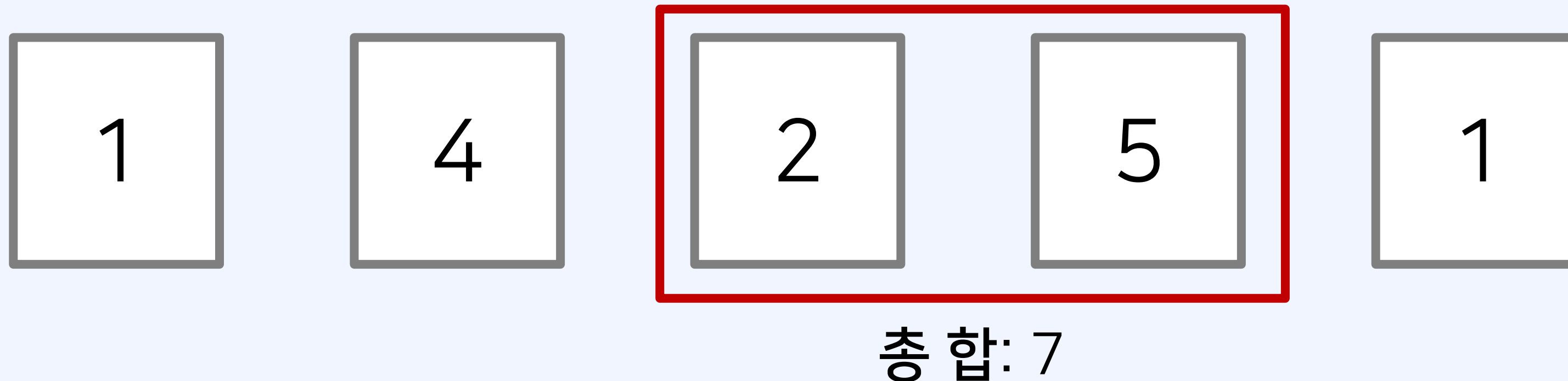
문제 풀이 핵심 아이디어

JavaScript
투 포인터
투 포인터
문제 풀이

- 고정 크기 윈도우를 사용하여 슬라이딩 윈도우를 사용한다.
 - 윈도우 크기가 2일 때의 예시는 다음과 같다.



- 고정 크기 윈도우를 사용하여 슬라이딩 윈도우를 사용한다.
 - 윈도우 크기가 2일 때의 예시는 다음과 같다.

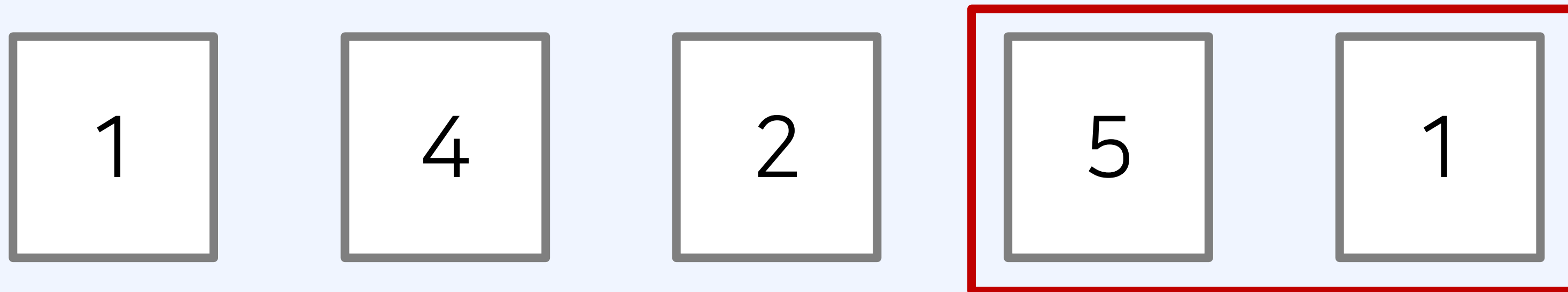


JavaScript 투 포인터 투 포인터 문제 풀이

문제 풀이 핵심 아이디어

JavaScript
투 포인터
투 포인터
문제 풀이

- 고정 크기 윈도우를 사용하여 슬라이딩 윈도우를 사용한다.
 - 윈도우 크기가 2일 때의 예시는 다음과 같다.



총 합: 6

JavaScript 투 포인터 투 포인터 문제 풀이

정답 코드 예시

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

let [n, x] = input[0].split(" ").map(Number);
let arr = [0, ...input[1].split(" ").map(Number)];

let sum = 0;
for (let i = 1; i <= n; i++) {
    // 1부터 x번째 날의 방문자 수의 합
    if (i <= x) sum += arr[i];
}
let maxSum = sum; // 가장 큰 합
let count = 1; // 기간의 개수
```

JavaScript
투 포인터
문제 풀이

JavaScript 투 포인터 투 포인터 문제 풀이

정답 코드 예시

JavaScript 투 포인터

투 포인터
문제 풀이

```
// 슬라이딩 윈도우 시작
let left = 1;
let right = x;
while (true) { // 윈도우를 한 칸 오른쪽으로 이동하기
    left += 1;
    right += 1;
    if (right > n) break;
    sum = sum + arr[right] - arr[left - 1]; // 합을 계산하여 정답 갱신
    if (maxSum == sum) count += 1;
    else if (maxSum < sum) { // 더 큰 합을 찾은 경우
        maxSum = sum;
        count = 1;
    }
}

if (maxSum == 0) console.log("SAD"); // 정답 출력
else {
    console.log(maxSum);
    console.log(count);
}
```

JavaScript 투 포인터
투 포인터 문제 풀이

혼자 힘으로 풀어보기

JavaScript
투 포인터
투 포인터
문제 풀이

문제 제목: 가장 긴 짝수 연속한 부분 수열 (large)

문제 난이도: ★★☆☆☆

문제 유형: 투 포인터

추천 풀이 시간: 50분

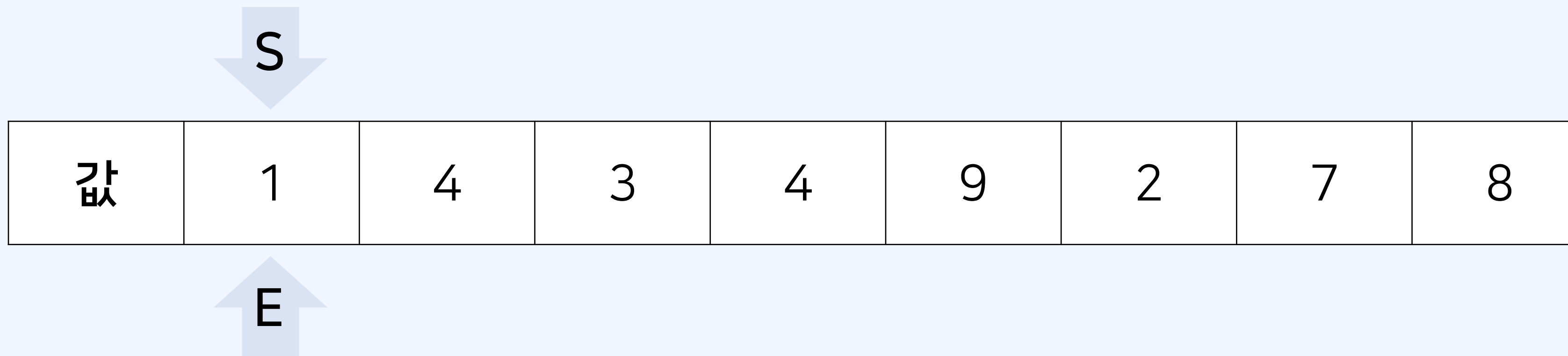
- 길이가 N 인 수열에서 K 개의 원소를 삭제할 수 있다.
 - 결과적으로 짝수로 이루어진 연속한 부분 수열 중에서 가장 긴 것을 계산하면 된다.
- 본 문제는 투 포인터(two pointers) 알고리즘으로 해결 가능하다.

인덱스	0	1	2	3	4	5	6	7
값	1	4	3	4	9	2	7	8

- 길이가 N 인 수열에서 K 개의 원소를 삭제할 수 있다.
 - 결과적으로 짝수로 이루어진 연속한 부분 수열 중에서 가장 긴 것을 계산하면 된다.
- 본 문제는 투 포인터(two pointers) 알고리즘으로 해결 가능하다.
 - 짝수인 경우: 단순히 end를 오른쪽으로 이동
 - 홀수인 경우: 최대 K 까지 end를 오른쪽으로 이동

< K 가 3일 때의 예시 >

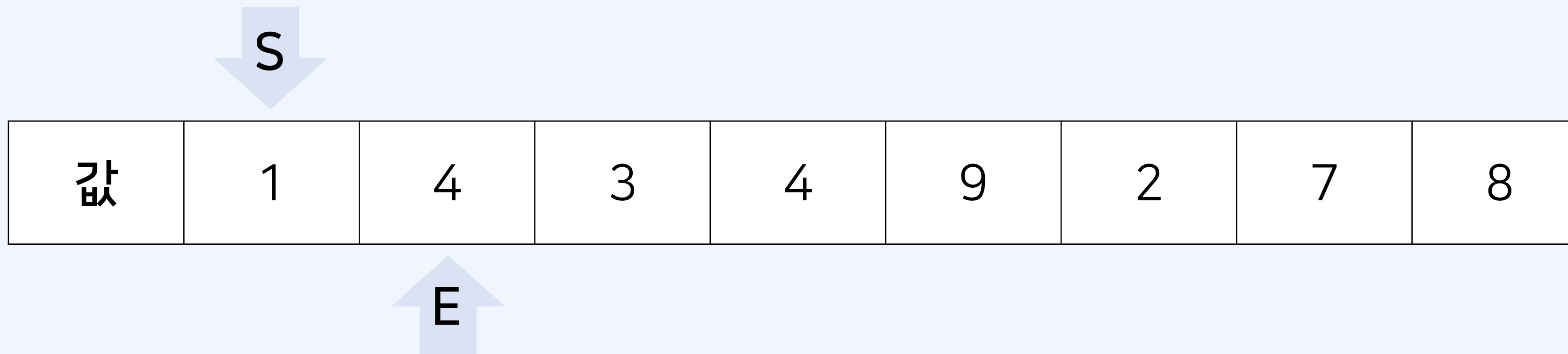
지운 문자의 개수: 1



- 길이가 N 인 수열에서 K 개의 원소를 삭제할 수 있다.
 - 결과적으로 짝수로 이루어진 연속한 부분 수열 중에서 가장 긴 것을 계산하면 된다.
- 본 문제는 투 포인터(two pointers) 알고리즘으로 해결 가능하다.
 - 짝수인 경우: 단순히 end를 오른쪽으로 이동
 - 홀수인 경우: 최대 K 까지 end를 오른쪽으로 이동

< K 가 3일 때의 예시 >

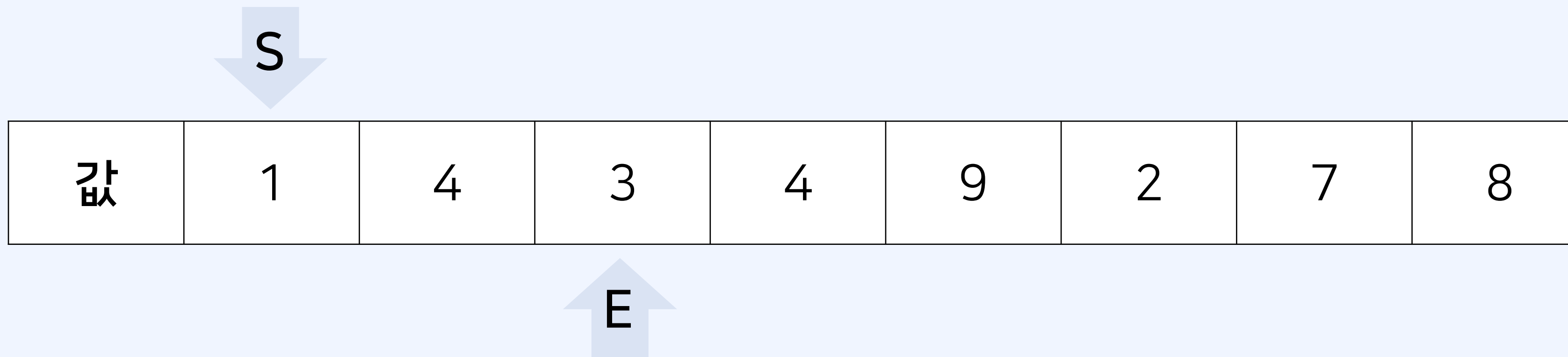
지운 문자의 개수: 1



- 길이가 N 인 수열에서 K 개의 원소를 삭제할 수 있다.
 - 결과적으로 짝수로 이루어진 연속한 부분 수열 중에서 가장 긴 것을 계산하면 된다.
- 본 문제는 투 포인터(two pointers) 알고리즘으로 해결 가능하다.
 - 짝수인 경우: 단순히 end를 오른쪽으로 이동
 - 홀수인 경우: 최대 K 까지 end를 오른쪽으로 이동

< K 가 3일 때의 예시 >

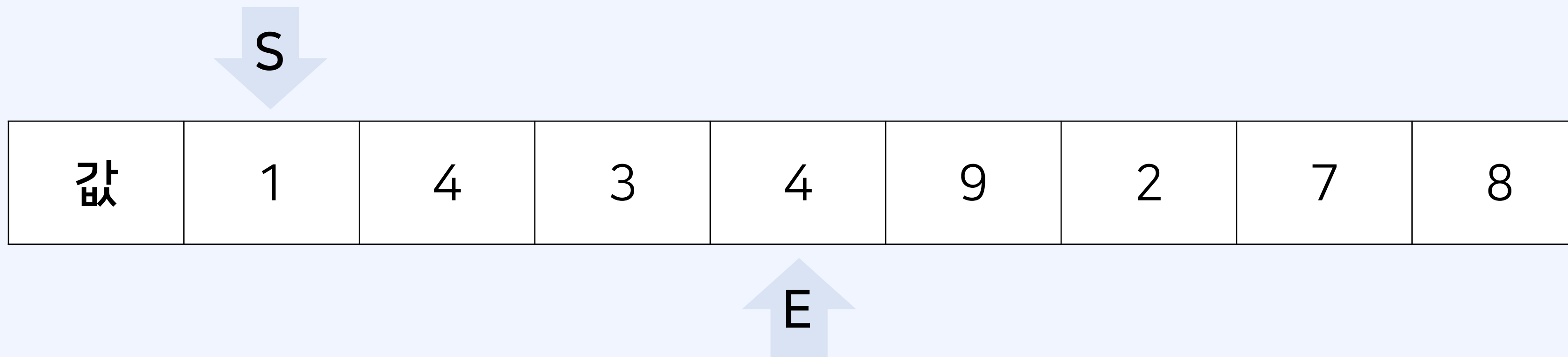
지운 문자의 개수: 2



- 길이가 N 인 수열에서 K 개의 원소를 삭제할 수 있다.
 - 결과적으로 짝수로 이루어진 연속한 부분 수열 중에서 가장 긴 것을 계산하면 된다.
- 본 문제는 투 포인터(two pointers) 알고리즘으로 해결 가능하다.
 - 짝수인 경우: 단순히 end를 오른쪽으로 이동
 - 홀수인 경우: 최대 K 까지 end를 오른쪽으로 이동

< K 가 3일 때의 예시 >

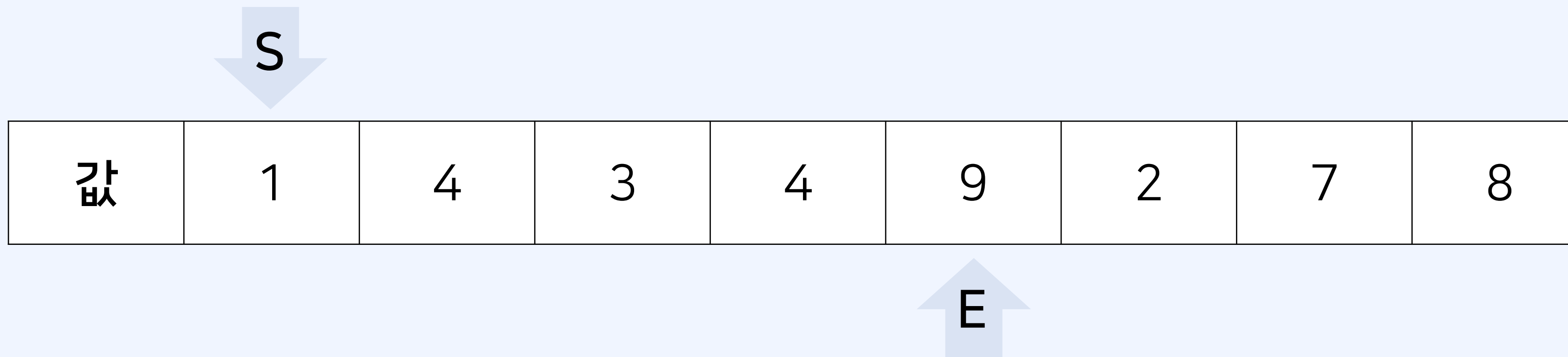
지운 문자의 개수: 2



- 길이가 N 인 수열에서 K 개의 원소를 삭제할 수 있다.
 - 결과적으로 짝수로 이루어진 연속한 부분 수열 중에서 가장 긴 것을 계산하면 된다.
- 본 문제는 투 포인터(two pointers) 알고리즘으로 해결 가능하다.
 - 짝수인 경우: 단순히 end를 오른쪽으로 이동
 - 홀수인 경우: 최대 K 까지 end를 오른쪽으로 이동

< K 가 3일 때의 예시 >

지운 문자의 개수: 3



- 길이가 N 인 수열에서 K 개의 원소를 삭제할 수 있다.
 - 결과적으로 짝수로 이루어진 연속한 부분 수열 중에서 가장 긴 것을 계산하면 된다.
- 본 문제는 투 포인터(two pointers) 알고리즘으로 해결 가능하다.
 - 짝수인 경우: 단순히 end를 오른쪽으로 이동
 - 홀수인 경우: 최대 K 까지 end를 오른쪽으로 이동

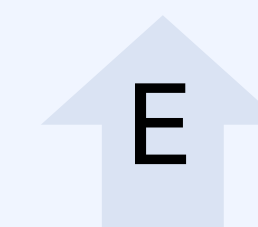
< K 가 3일 때의 예시 >

지운 문자의 개수: 3



값	1	4	3	4	9	2	7	8
---	---	---	---	---	---	---	---	---


현재까지의 정답: 3



- 길이가 N 인 수열에서 K 개의 원소를 삭제할 수 있다.
 - 결과적으로 짝수로 이루어진 연속한 부분 수열 중에서 가장 긴 것을 계산하면 된다.
- 본 문제는 투 포인터(two pointers) 알고리즘으로 해결 가능하다.
 - 짝수인 경우: 단순히 end를 오른쪽으로 이동
 - 홀수인 경우: 최대 K 까지 end를 오른쪽으로 이동

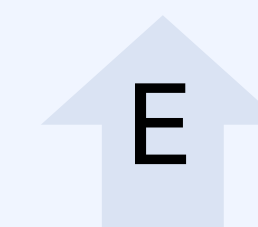
< K 가 3일 때의 예시 >

지운 문자의 개수: 2



값	1	4	3	4	9	2	7	8
---	---	---	---	---	---	---	---	---

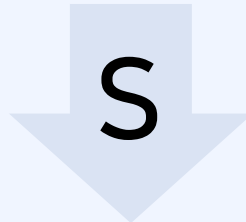
현재까지의 정답: 3



- 길이가 N 인 수열에서 K 개의 원소를 삭제할 수 있다.
 - 결과적으로 짝수로 이루어진 연속한 부분 수열 중에서 가장 긴 것을 계산하면 된다.
- 본 문제는 투 포인터(two pointers) 알고리즘으로 해결 가능하다.
 - 짝수인 경우: 단순히 end를 오른쪽으로 이동
 - 홀수인 경우: 최대 K 까지 end를 오른쪽으로 이동

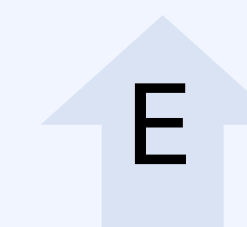
< K 가 3일 때의 예시 >

지운 문자의 개수: 3



값	1	4	3	4	9	2	7	8
---	---	---	---	---	---	---	---	---


현재까지의 정답: 3



- 길이가 N 인 수열에서 K 개의 원소를 삭제할 수 있다.
 - 결과적으로 짝수로 이루어진 연속한 부분 수열 중에서 가장 긴 것을 계산하면 된다.
- 본 문제는 투 포인터(two pointers) 알고리즘으로 해결 가능하다.
 - 짝수인 경우: 단순히 end를 오른쪽으로 이동
 - 홀수인 경우: 최대 K 까지 end를 오른쪽으로 이동

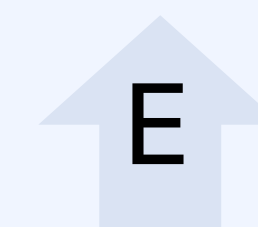
< K 가 3일 때의 예시 >

지운 문자의 개수: 3



값	1	4	3	4	9	2	7	8
---	---	---	---	---	---	---	---	---

현재까지의 정답: 4



JavaScript 투 포인터 투 포인터 문제 풀이

정답 코드 예시

JavaScript 투 포인터

투 포인터
문제 풀이

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

let [n, k] = input[0].split(" ").map(Number);
let arr = input[1].split(" ").map(Number);

let result = 0;
let eraseCount = 0;
for (let start = 0, end = 0; start < n; start++) {
    while (end < n) {
        if (arr[end] % 2 == 0) end++; // 짝수인 경우 end 증가
        else { // 홀수인 경우
            if (eraseCount == k) break; // 더 지울 수 없다면 종료
            // 지울 수 있는 여건이 있다면 지우기
            eraseCount++;
            end++;
        }
    }
    result = Math.max(result, end - start - eraseCount); // 범위의 길이 계산
    // start를 한 칸 오른쪽으로 이동할 때, 가능하다면 지울 수 있는 개수 증가
    if (arr[start] % 2 == 1) eraseCount--;
}
console.log(result);
```


JavaScript 투 포인터
투 포인터 문제 풀이

혼자 힘으로 풀어보기

JavaScript
투 포인터
투 포인터
문제 풀이

문제 제목: 배열 합치기

문제 난이도: ★☆☆☆☆

문제 유형: 투 포인터, 정렬

추천 풀이 시간: 20분

JavaScript 투 포인터
투 포인터 문제 풀이

문제 풀이 핵심 아이디어

JavaScript
투 포인터
투 포인터
문제 풀이

- 정렬되어 있는 배열 A 와 B 가 주어진다.
- 두 배열을 이어 붙인 뒤에 정렬한 결과를 출력한다.

JavaScript 투 포인터
투 포인터 문제 풀이

문제 풀이 핵심 아이디어

JavaScript
투 포인터
문제 풀이

[해결 방법] 두 배열 A 와 B 가 이미 정렬되어 있다.

- 따라서, 두 배열에서의 각각 원소 위치를 가리키는 포인터 i 와 j 를 사용한다.
- 병합 정렬 알고리즘에서 두 개의 배열을 병합할 때 사용하는 알고리즘과 동일하다.

JavaScript 투 포인터

투 포인터 문제 풀이

문제 풀이 핵심 아이디어

- 각 부분 배열은 이미 정렬된 상태로 본다.
- 각 부분 배열에 대하여 **첫째 원소부터 시작하여 하나씩 확인**한다.
- 총 원소의 개수가 N 개일 때, $O(N)$ 의 시간 복잡도가 요구된다.



문제 풀이 핵심 아이디어

- 각 부분 배열은 이미 정렬된 상태로 본다.
- 각 부분 배열에 대하여 **첫째 원소부터 시작하여 하나씩 확인**한다.
- 총 원소의 개수가 N 개일 때, $O(N)$ 의 시간 복잡도가 요구된다.



JavaScript 투 포인터

투 포인터 문제 풀이

문제 풀이 핵심 아이디어

- 각 부분 배열은 이미 정렬된 상태로 본다.
- 각 부분 배열에 대하여 **첫째 원소부터 시작하여 하나씩 확인**한다.
- 총 원소의 개수가 N 개일 때, $O(N)$ 의 시간 복잡도가 요구된다.



JavaScript 투 포인터

투 포인터 문제 풀이

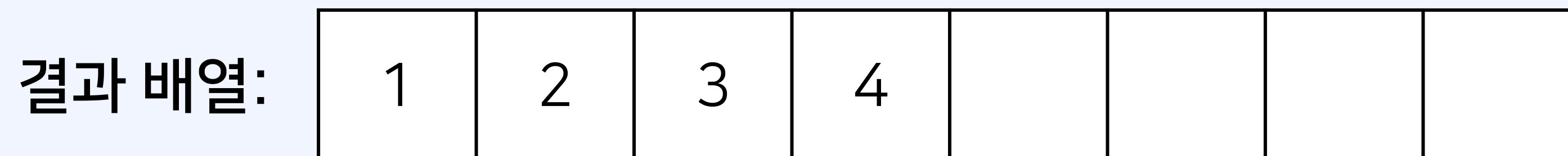
문제 풀이 핵심 아이디어

- 각 부분 배열은 이미 정렬된 상태로 본다.
- 각 부분 배열에 대하여 **첫째 원소부터 시작하여 하나씩 확인**한다.
- 총 원소의 개수가 N 개일 때, $O(N)$ 의 시간 복잡도가 요구된다.



문제 풀이 핵심 아이디어

- 각 부분 배열은 이미 정렬된 상태로 본다.
- 각 부분 배열에 대하여 **첫째 원소부터 시작하여 하나씩 확인**한다.
- 총 원소의 개수가 N 개일 때, $O(N)$ 의 시간 복잡도가 요구된다.

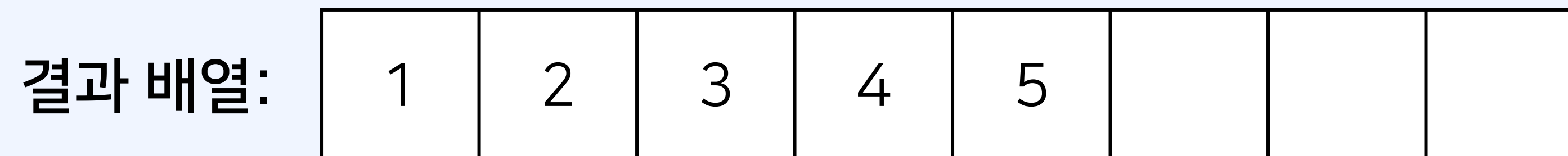


JavaScript 투 포인터

투 포인터 문제 풀이

문제 풀이 핵심 아이디어

- 각 부분 배열은 이미 정렬된 상태로 본다.
- 각 부분 배열에 대하여 **첫째 원소부터 시작하여 하나씩 확인**한다.
- 총 원소의 개수가 N 개일 때, $O(N)$ 의 시간 복잡도가 요구된다.

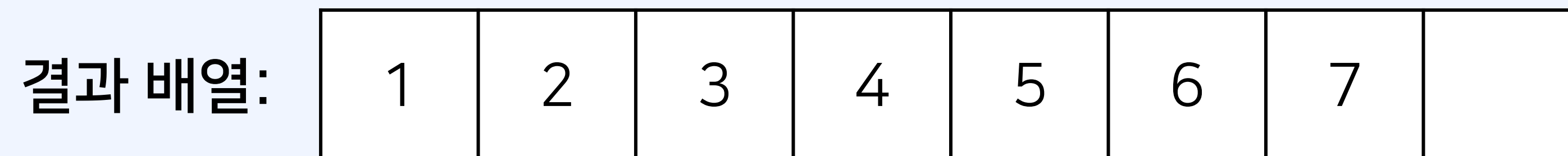
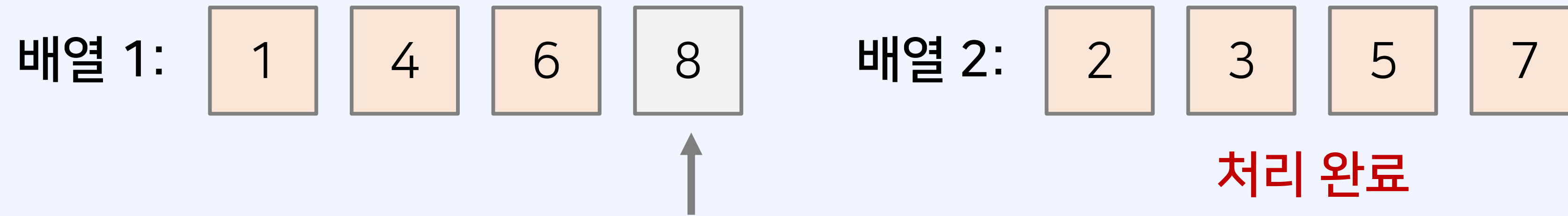


JavaScript 투 포인터
투 포인터 문제 풀이

문제 풀이 핵심 아이디어

JavaScript
투 포인터
투 포인터
문제 풀이

- 각 부분 배열은 이미 정렬된 상태로 본다.
- 각 부분 배열에 대하여 **첫째 원소부터 시작하여 하나씩 확인**한다.
- 총 원소의 개수가 N 개일 때, $O(N)$ 의 시간 복잡도가 요구된다.



JavaScript 투 포인터 투 포인터 문제 풀이

정답 코드 예시

JavaScript
투 포인터
문제 풀이

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

let [n, m] = input[0].split(" ").map(Number);
let a = input[1].split(" ").map(Number);
let b = input[2].split(" ").map(Number);

let result = [];
let i = 0;
let j = 0;

while (i < n && j < m) { // 배열 A와 배열 B에서 차례대로 더 작은 원소 넣기
    if (a[i] < b[j]) { // 배열 A의 원소가 더 작다면
        result.push(a[i]);
        i += 1;
    }
    else { // 배열 B의 원소가 더 작다면
        result.push(b[j]);
        j += 1;
    }
}
```

JavaScript 투 포인터 투 포인터 문제 풀이

정답 코드 예시

```
// 각 배열에 남아있는 원소들을 순차적으로 삽입
while (i < n) {
    result.push(a[i]);
    i += 1;
}
while (j < m) {
    result.push(b[j]);
    j += 1;
}

// 결과 배열의 각 원소를 공백 기준으로 출력
console.log(result.join(" "));
```

JavaScript
투 포인터
투 포인터
문제 풀이