알고리즘 특강
이분탐색

값을 가장 빠르게 찾아내는 방법입니다.
하지만, 정렬이 반드시 수반되기 때문에, 직렬탐색과의 효용성을 비교해야 합니다.

이분탐색

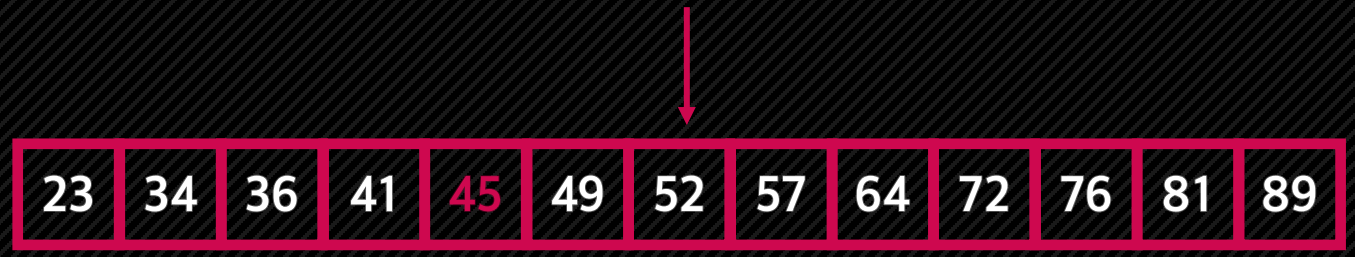
"1에서 100까지 숫자 중에 내가 생각한 수가 뭔지 맞춰봐!"

이분탐색

Binary Search

- 전체 범위를 **이분할** 하여 가운데의 값을 비교하여 두 영역 중 다른 영역으로 이동하여 탐색하는 기법.
- 시간 복잡도는 **O(logN)** 임이 보장됨.
- 탐색 전 리스트가 **반드시 정렬**되어 있어야 함. → 배열을 적게 탐색하는 경우에는 이분탐색을 쓰지 않는 것이 나을 수 있음.

이분탐색



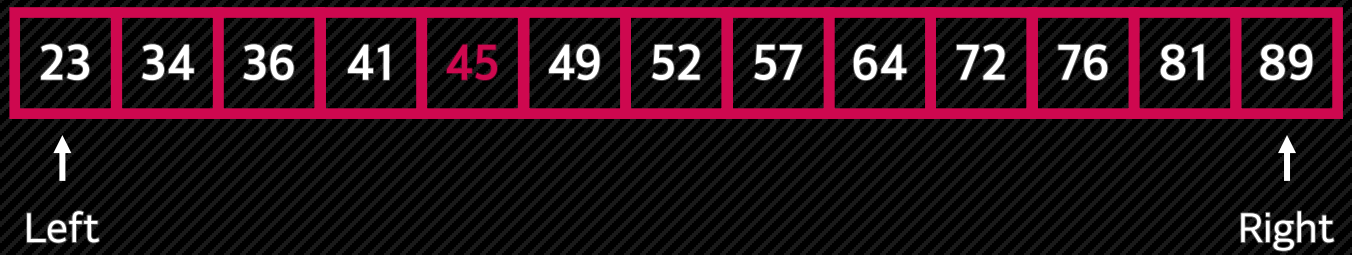
- 핵심은, 가운데다!

이분탐색

```
function BinarySearch(list, target) {
    set left = 0
    set right = last index of list

    while (left < right) do
        set mid = (left + right) / 2
        if (the mid-th value of list is smaller than target) do
            left = mid + 1
        end else do
            right = mid
        end
    end

    return (left + right) / 2
}
```

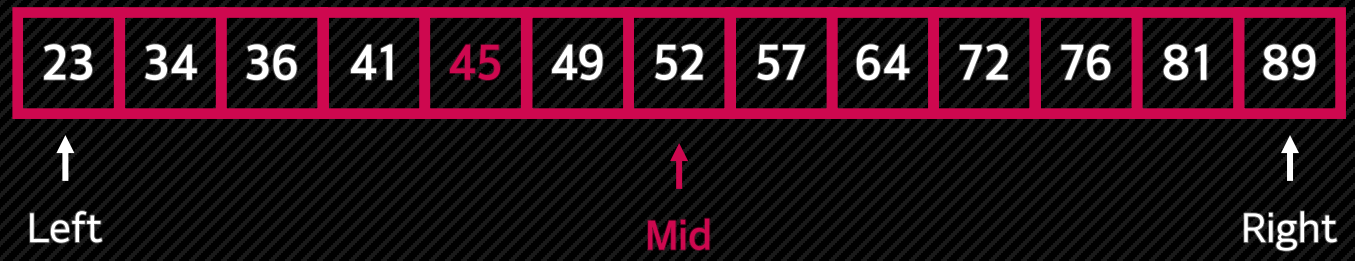


이분탐색

```
function BinarySearch(list, target) {
    set left = 0
    set right = last index of list

    while (left < right) do
        set mid = (left + right) / 2
        if (the mid-th value of list is smaller than target) do
            left = mid + 1
        end else do
            right = mid
        end
    end

    return (left + right) / 2
}
```

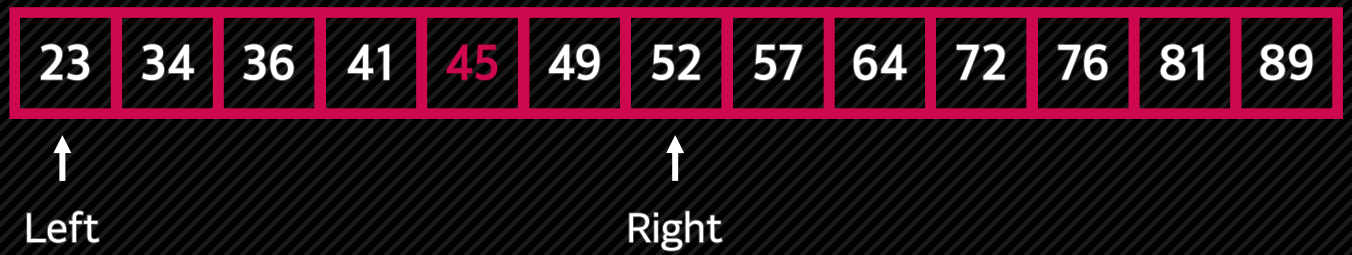


이분탐색

```
function BinarySearch(list, target) {
    set left = 0
    set right = last index of list

    while (left < right) do
        set mid = (left + right) / 2
        if (the mid-th value of list is smaller than target) do
            left = mid + 1
        end else do
            right = mid
        end
    end

    return (left + right) / 2
}
```

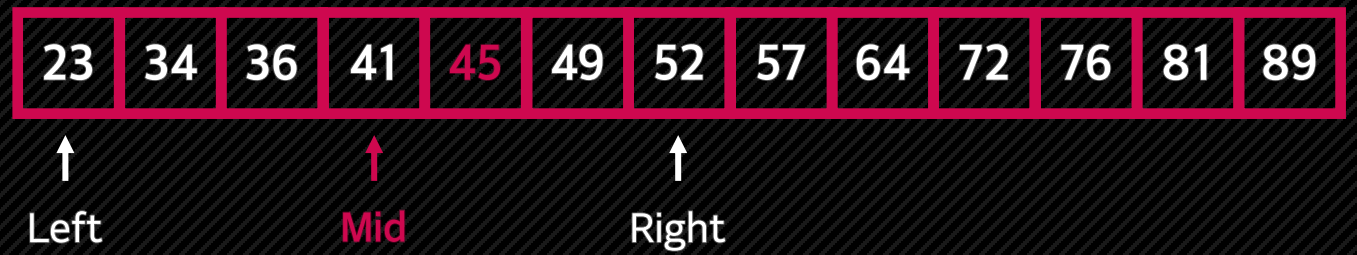


이분탐색

```
function BinarySearch(list, target) {
    set left = 0
    set right = last index of list

    while (left < right) do
        set mid = (left + right) / 2
        if (the mid-th value of list is smaller than target) do
            left = mid + 1
        end else do
            right = mid
        end
    end

    return (left + right) / 2
}
```

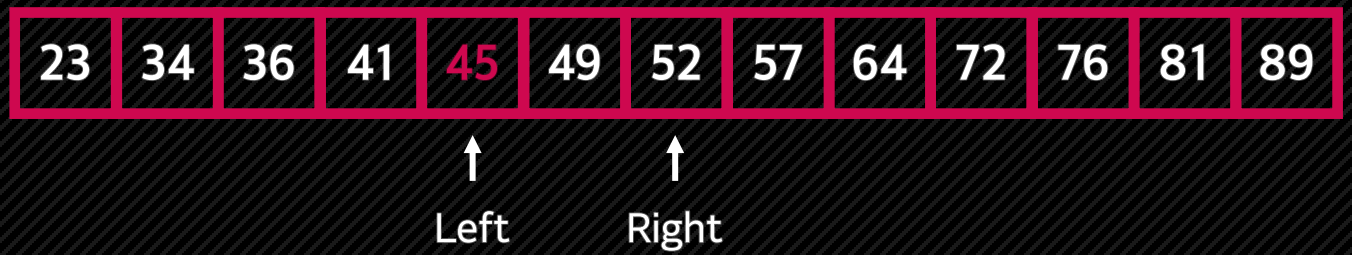


이분탐색

```
function BinarySearch(list, target) {
    set left = 0
    set right = last index of list

    while (left < right) do
        set mid = (left + right) / 2
        if (the mid-th value of list is smaller than target) do
            left = mid + 1
        end else do
            right = mid
        end
    end

    return (left + right) / 2
}
```

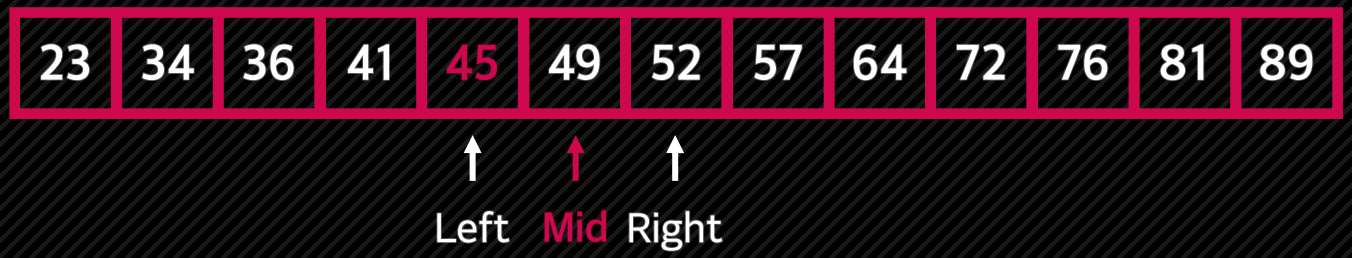


이분탐색

```
function BinarySearch(list, target) {
    set left = 0
    set right = last index of list

    while (left < right) do
        set mid = (left + right) / 2
        if (the mid-th value of list is smaller than target) do
            left = mid + 1
        end else do
            right = mid
        end
    end

    return (left + right) / 2
}
```

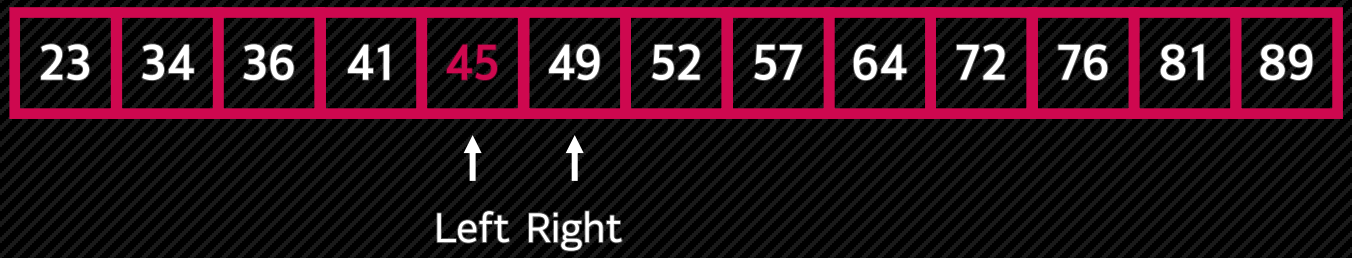


이분탐색

```
function BinarySearch(list, target) {
    set left = 0
    set right = last index of list

    while (left < right) do
        set mid = (left + right) / 2
        if (the mid-th value of list is smaller than target) do
            left = mid + 1
        end else do
            right = mid
        end
    end

    return (left + right) / 2
}
```

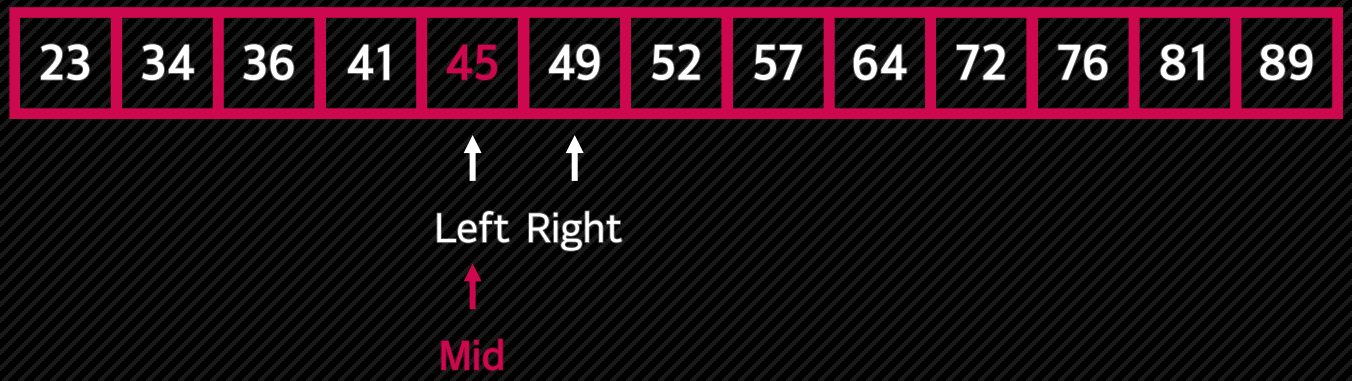


이분탐색

```
function BinarySearch(list, target) {
    set left = 0
    set right = last index of list

    while (left < right) do
        set mid = (left + right) / 2
        if (the mid-th value of list is smaller than target) do
            left = mid + 1
        end else do
            right = mid
        end
    end

    return (left + right) / 2
}
```

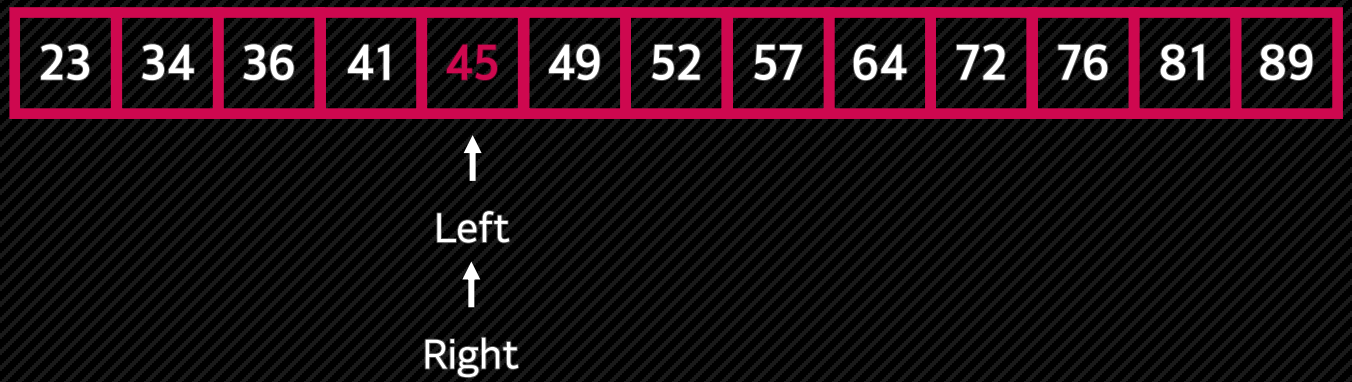


이분탐색

```
function BinarySearch(list, target) {
    set left = 0
    set right = last index of list

    while (left < right) do
        set mid = (left + right) / 2
        if (the mid-th value of list is smaller than target) do
            left = mid + 1
        end else do
            right = mid
        end
    end

    return (left + right) / 2
}
```



순차탐색 VS 이분탐색

	Linear Search	Binary Search
시간 복잡도	O(N)	O(logN)
N = 4	4회	2회
N = 32	32회	5회
N = 128	128회	7회
N = 65,536	65,536회	16회
N = 4,294,967,296	4,294,967,296회	32회

순차탐색 VS 이분탐색

	Linear Search	Binary Search
시간 복잡도	O(N)	O(logN)
N = 4	4회	2회
N = 32	32회	5회
N = 128	128회	7회
N = 65,536	65,536회	16회
N = 4,294,967,296	4,294,967,296회	32회

- 그러나, 이분탐색은 리스트가 **반드시 정렬**되어 있어야 함. → 상황에 따라 순차탐색이 더 나을 수 있다!
- 즉, O(N) 이하로 해결될 수 있는 방법이 있다면 굳이 이분탐색을 사용할 필요가 없음.

Silver 4 - 수 찾기

요약

- N개의 정수 A[1], A[2], …, A[N]이 주어져 있을 때, 이 안에 X라는 정수가 존재하는지 알아내는 프로그램을 작성하시오.

제약조건

- N의 범위는 $1 <= N <= 100,000$ 이다.
- 찾아야 하는 수의 개수 M의 범위는 $1 <= M <= 100,000$ 이다.
- 모든 수의 범위는 $-2^{31} < N_i < 2^{31}$ 이다.

in

```
for i in t:
    if i in _list:
        print(1)
    else:
        print(0)
```

- List 구조에서의 in 메소드는 정렬 여부와 상관없이 순차탐색을 진행함.
- 잦은 자료의 검색에 최적화된 자료구조 (ex. Dict, Set) 이 아닌 이상, in은 최대한 피하는게 좋다!

Silver 4 - 듣보잡

요약

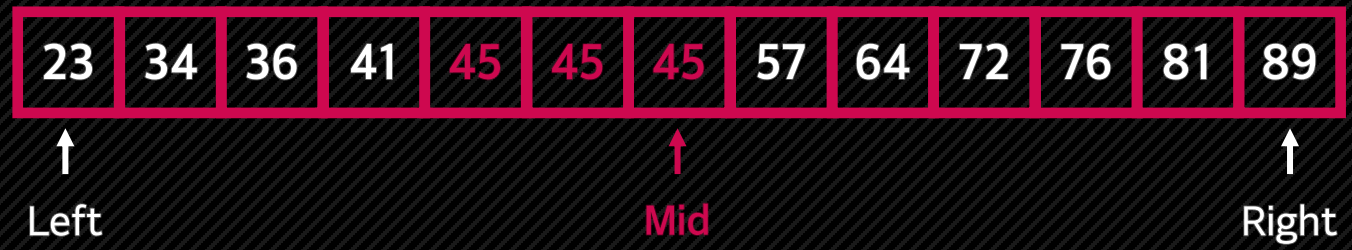
- 듣지도 못한 사람의 명단과, 보지도 못한 사람의 명단이 주어질 때, 듣도 보도 못한 사람의 명단을 구하는 프로그램을 작성하시오.

제약조건

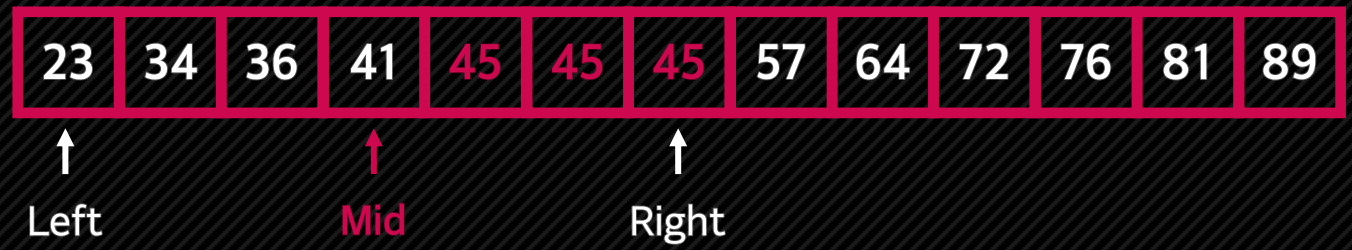
- 듣지도 못한 사람의 범위는 1 <= N <= 500,000 이다.
- 보지도 못한 사람의 범위는 1 <= M <= 500,000 이다.

이런 경우는 어떨까?

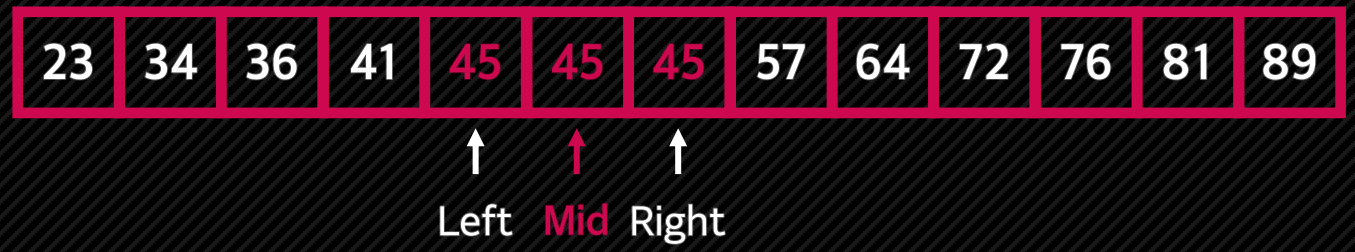
이런 경우는 어떨까?



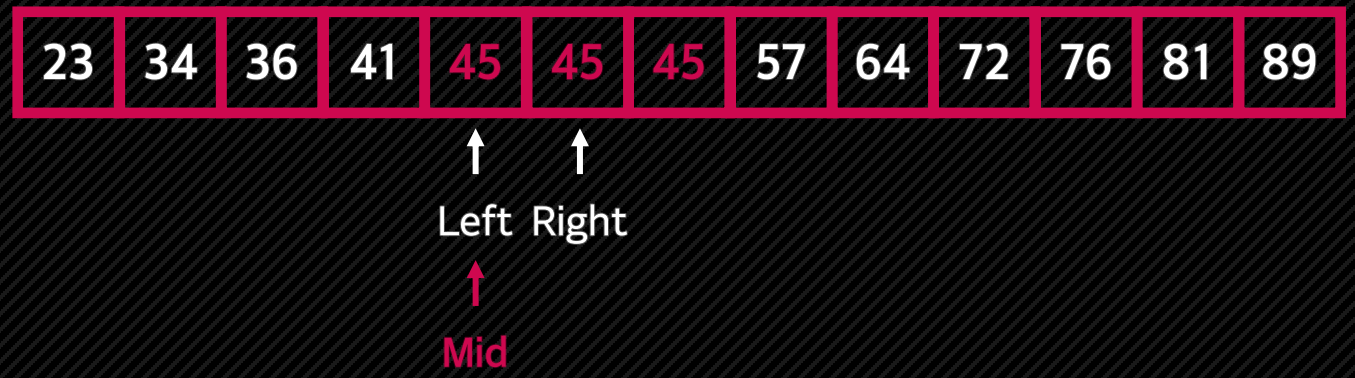
이런 경우는 어떨까?



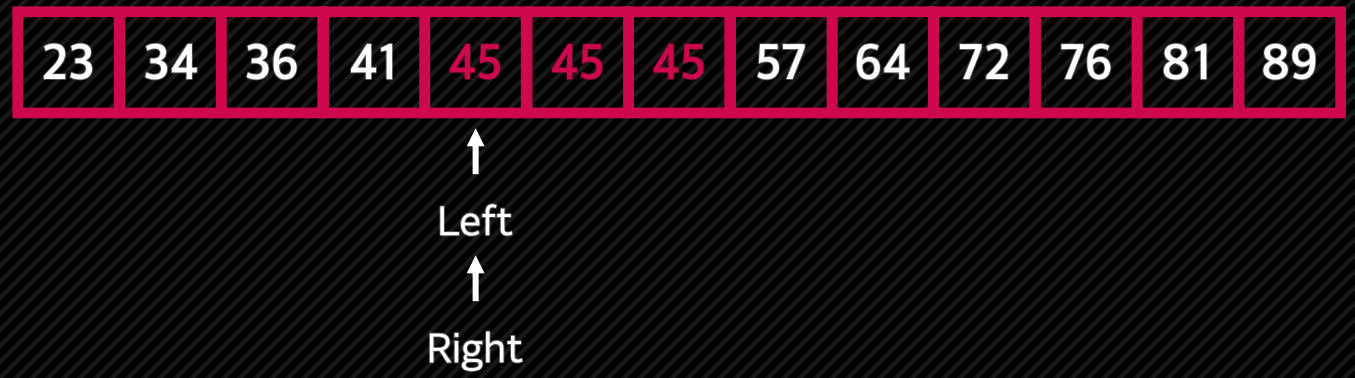
이런 경우는 어떨까?



이런 경우는 어떨까?



이런 경우는 어떨까?

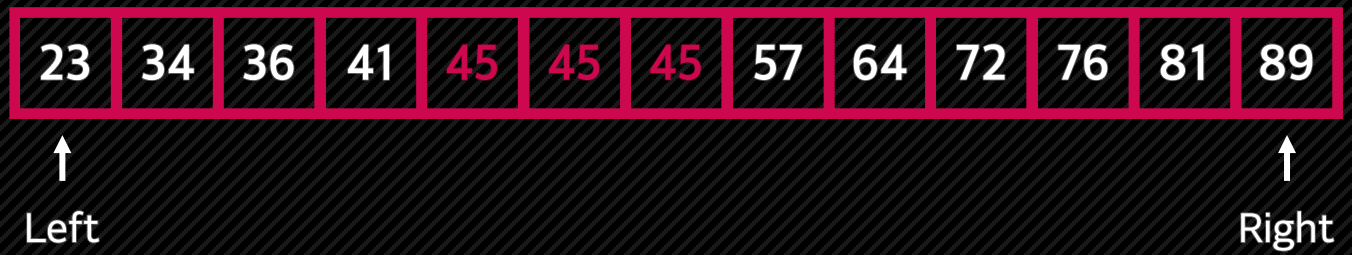


살짝만 알고리즘을 바꿔보자!

```
function BinarySearch(list, target) {
    set left = 0
    set right = last index of list

    while (left < right) do
        set mid = (left + right) / 2
        if (the mid-th value of list is smaller or same than target) do
            left = mid + 1
        end else do
            right = mid
        end
    end

    return (left + right) / 2
}
```

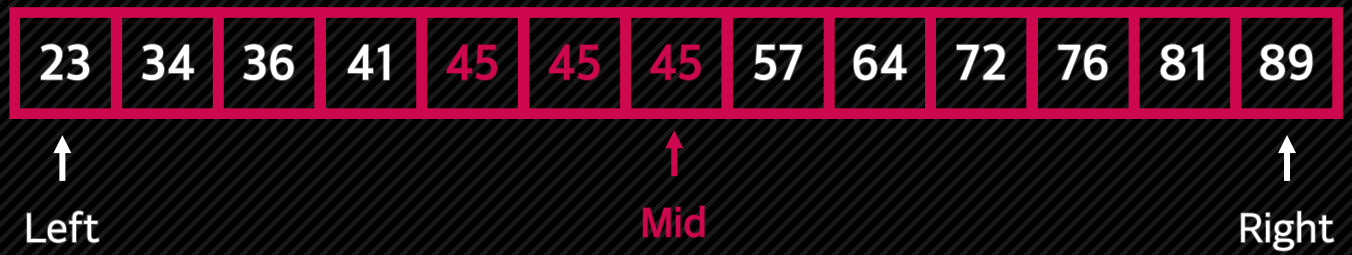


살짝만 알고리즘을 바꿔보자!

```
function BinarySearch(list, target) {
    set left = 0
    set right = last index of list

    while (left < right) do
        set mid = (left + right) / 2
        if (the mid-th value of list is smaller or same than target) do
            left = mid + 1
        end else do
            right = mid
        end
    end

    return (left + right) / 2
}
```

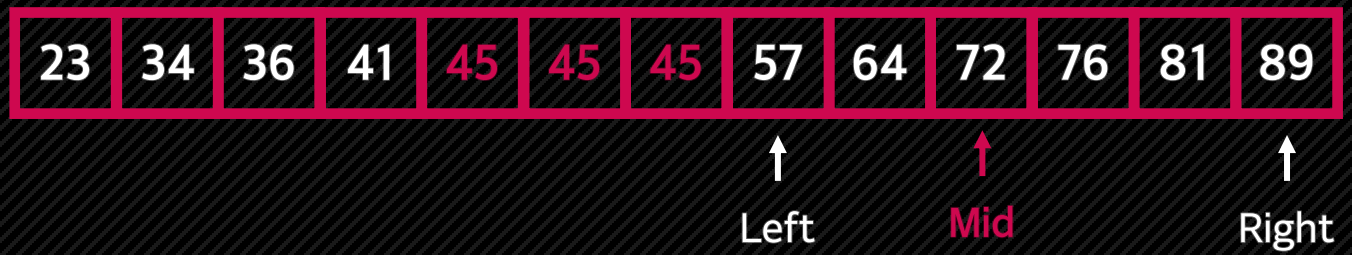


살짝만 알고리즘을 바꿔보자!

```
function BinarySearch(list, target) {
    set left = 0
    set right = last index of list

    while (left < right) do
        set mid = (left + right) / 2
        if (the mid-th value of list is smaller or same than target) do
            left = mid + 1
        end else do
            right = mid
        end
    end

    return (left + right) / 2
}
```

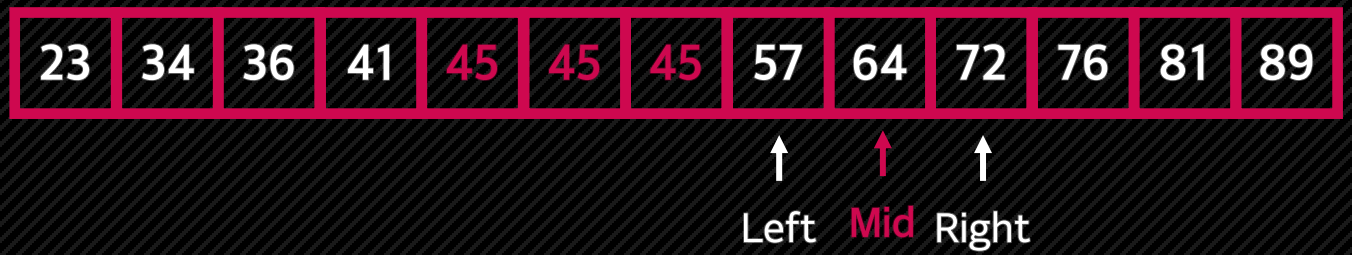


살짝만 알고리즘을 바꿔보자!

```
function BinarySearch(list, target) {
    set left = 0
    set right = last index of list

    while (left < right) do
        set mid = (left + right) / 2
        if (the mid-th value of list is smaller or same than target) do
            left = mid + 1
        end else do
            right = mid
        end
    end

    return (left + right) / 2
}
```

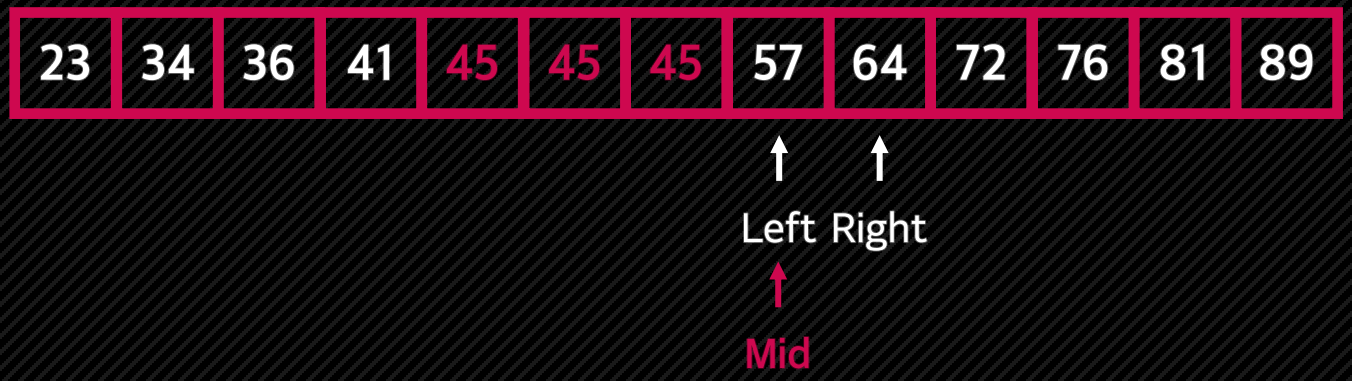


살짝만 알고리즘을 바꿔보자!

```
function BinarySearch(list, target) {
    set left = 0
    set right = last index of list

    while (left < right) do
        set mid = (left + right) / 2
        if (the mid-th value of list is smaller or same than target) do
            left = mid + 1
        end else do
            right = mid
        end
    end

    return (left + right) / 2
}
```

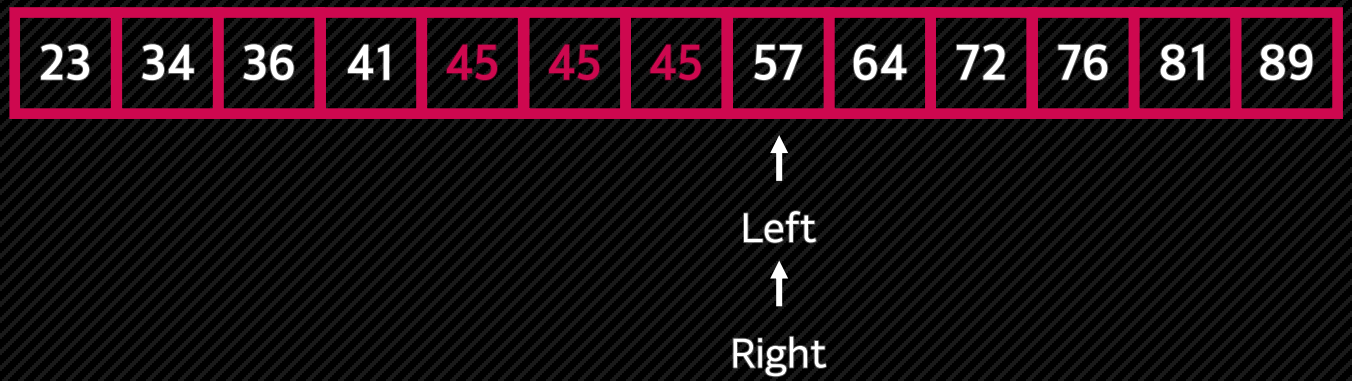


살짝만 알고리즘을 바꿔보자!

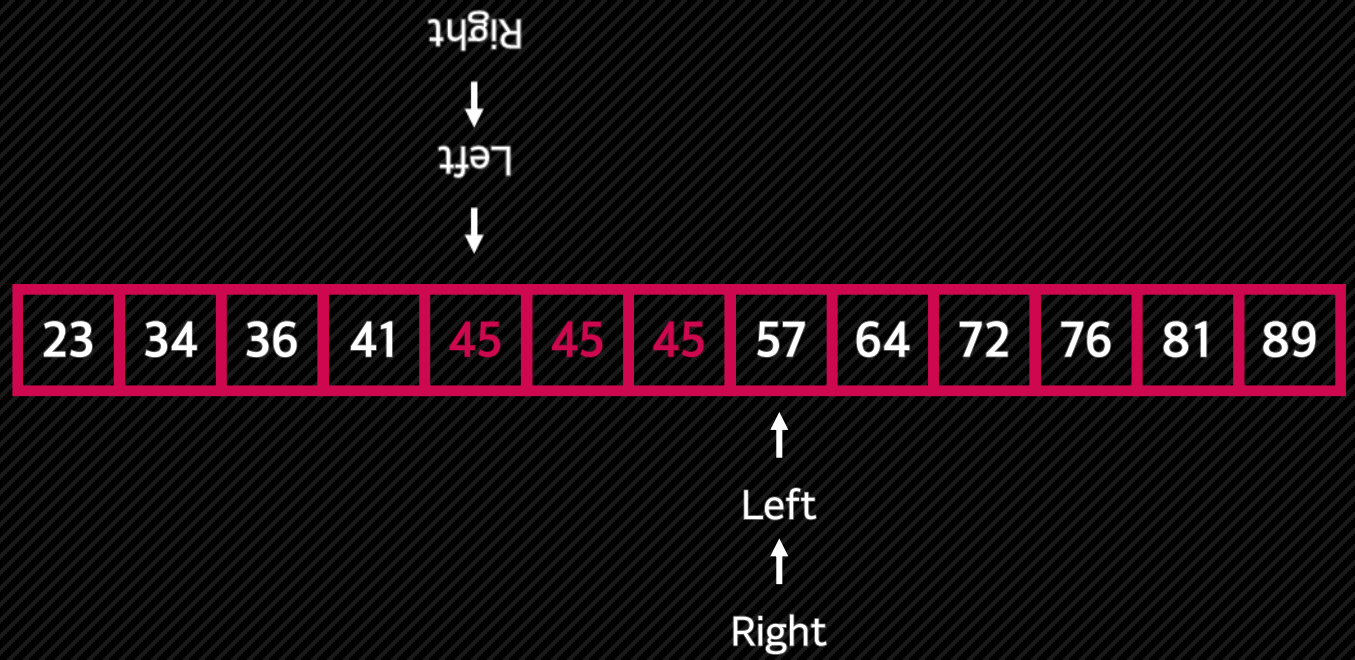
```
function BinarySearch(list, target) {
    set left = 0
    set right = last index of list

    while (left < right) do
        set mid = (left + right) / 2
        if (the mid-th value of list is smaller or same than target) do
            left = mid + 1
        end else do
            right = mid
        end
    end

    return (left + right) / 2
}
```



Lower/Upper bound



- Lower Bound: 원하는 값 N 이상이 처음 나오는 위치
- Upper Bound: 원하는 값 N을 처음으로 초과하는 위치

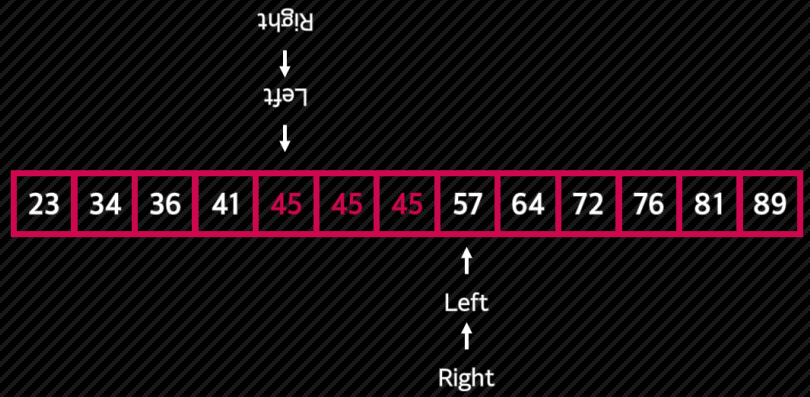
Lower/Upper bound

```python
import bisect

_list = [23, 34, 36, 41, 45, 45, 45, 57, 64, 72, 76, 81, 89]

print(bisect.bisect_left(_list, 45))
print(bisect.bisect_right(_list, 45))

print(bisect.bisect_right(_list, 45) - bisect.bisect_left(_list, 45))
```



도전!!!

Gold 2 - 합이 0인 네 정수

요약

- 정수로 이루어진 크기가 같은 배열 A, B, C, D가 있다.
- A[a], B[b], C[c], D[d]의 합이 0인 (a, b, c, d) 쌍의 개수를 구하는 프로그램을 작성하시오.

제약조건

- 배열의 크기 n은 $1 <= n <= 4{,}000$ 이다.
- 배열에 들어있는 정수의 절댓값은 2^{28} 이하 이다.

도전!!!

Gold 2 - 합이 0인 네 정수

요약

- 정수로 이루어진 크기가 같은 배열 A, B, C, D가 있다.
- A[a], B[b], C[c], D[d]의 합이 0인 (a, b, c, d) 쌍의 개수를 구하는 프로그램을 작성하시오.

제약조건

- 배열의 크기 n은 $1 <= n <= 4{,}000$ 이다.
- 배열에 들어있는 정수의 절댓값은 2^{28} 이하 이다.

접근

- 일단 완전탐색부터 생각해보자.

도전!!!

Gold 2 - 합이 0인 네 정수

요약

- 정수로 이루어진 크기가 같은 배열 A, B, C, D가 있다.
- A[a], B[b], C[c], D[d]의 합이 0인 (a, b, c, d) 쌍의 개수를 구하는 프로그램을 작성하시오.

제약조건

- 배열의 크기 n은 $1 <= n <= 4{,}000$ 이다.
- 배열에 들어있는 정수의 절댓값은 2^{28} 이하 이다.

접근

- 일단 완전탐색부터 생각해보자.

→ 4,000 * 4,000 * 4,000 * 4,000…

도전!!!

Gold 2 - 합이 0인 네 정수

요약

- 정수로 이루어진 크기가 같은 배열 A, B, C, D가 있다.
- A[a], B[b], C[c], D[d]의 합이 0인 (a, b, c, d) 쌍의 개수를 구하는 프로그램을 작성하시오.

제약조건

- 배열의 크기 n은 $1 <= n <= 4{,}000$ 이다.
- 배열에 들어있는 정수의 절댓값은 2^{28} 이하 이다.

접근

- 일단 완전탐색부터 생각해보자.
  → 4,000 * 4,000 * 4,000 * 4,000…
- A[a] + B[b] + C[c] + D[d] = 0 이래.

도전!!!

Gold 2 - 합이 0인 네 정수

요약

- 정수로 이루어진 크기가 같은 배열 A, B, C, D가 있다.
- A[a], B[b], C[c], D[d]의 합이 0인 (a, b, c, d) 쌍의 개수를 구하는 프로그램을 작성하시오.

제약조건

- 배열의 크기 n은 $1 <= n <= 4{,}000$ 이다.
- 배열에 들어있는 정수의 절댓값은 2^{28} 이하 이다.

접근

- 일단 완전탐색부터 생각해보자.
  - → 4,000 * 4,000 * 4,000 * 4,000…
- A[a] + B[b] + C[c] + D[d] = 0 이래.
  - → 식을 살짝 바꾸면 A[a] + B[b] = -(C[c] + D[d]) 겠네?
  - → 가능한 경우의 수는 각각 16,000,000개.

도전!!!

Gold 2 - 합이 0인 네 정수

요약

- 정수로 이루어진 크기가 같은 배열 A, B, C, D가 있다.
- A[a], B[b], C[c], D[d]의 합이 0인 (a, b, c, d) 쌍의 개수를 구하는 프로그램을 작성하시오.

제약조건

- 배열의 크기 n은 $1 <= n <= 4,000$ 이다.
- 배열에 들어있는 정수의 절댓값은 2^{28} 이하 이다.

접근

- 일단 완전탐색부터 생각해보자.
  - → 4,000 * 4,000 * 4,000 * 4,000…
- A[a] + B[b] + C[c] + D[d] = 0 이래.
  - → 식을 살짝 바꾸면 A[a] + B[b] = -(C[c] + D[d]) 겠네?
  - → 가능한 경우의 수는 각각 16,000,000개.
- 식의 앞 부분을 A, 뒷 부분을 B라고 하자.

도전!!!

Gold 2 - 합이 0인 네 정수

요약

- 정수로 이루어진 크기가 같은 배열 A, B, C, D가 있다.
- A[a], B[b], C[c], D[d]의 합이 0인 (a, b, c, d) 쌍의 개수를 구하는 프로그램을 작성하시오.

제약조건

- 배열의 크기 n은 $1 <= n <= 4,000$ 이다.
- 배열에 들어있는 정수의 절댓값은 2^{28} 이하 이다.

접근

- 일단 완전탐색부터 생각해보자.
  - → 4,000 * 4,000 * 4,000 * 4,000…
- A[a] + B[b] + C[c] + D[d] = 0 이래.
  - → 식을 살짝 바꾸면 A[a] + B[b] = -(C[c] + D[d]) 겠네?
  - → 가능한 경우의 수는 각각 16,000,000개.
- 식의 앞 부분을 A, 뒷 부분을 B라고 하자.
  - → 그럼 A = -B의 개수를 찾으면 되겠네?

도전!!!

Gold 2 - 합이 0인 네 정수

요약

- 정수로 이루어진 크기가 같은 배열 A, B, C, D가 있다.
- A[a], B[b], C[c], D[d]의 합이 0인 (a, b, c, d) 쌍의 개수를 구하는 프로그램을 작성하시오.

제약조건

- 배열의 크기 n은 $1 <= n <= 4,000$ 이다.
- 배열에 들어있는 정수의 절댓값은 2^{28} 이하 이다.

접근

- 일단 완전탐색부터 생각해보자.
  - → 4,000 * 4,000 * 4,000 * 4,000…
- A[a] + B[b] + C[c] + D[d] = 0 이래.
  - → 식을 살짝 바꾸면 A[a] + B[b] = -(C[c] + D[d]) 겠네?
  - → 가능한 경우의 수는 각각 16,000,000개.
- 식의 앞 부분을 A, 뒷 부분을 B라고 하자.
  - → 그럼 A = -B의 개수를 찾으면 되겠네?
  - → 이분탐색!

도전!!!

Gold 2 - 합이 0인 네 정수

요약

- 정수로 이루어진 크기가 같은 배열 A, B, C, D가 있다.
- A[a], B[b], C[c], D[d]의 합이 0인 (a, b, c, d) 쌍의 개수를 구하는 프로그램을 작성하시오.

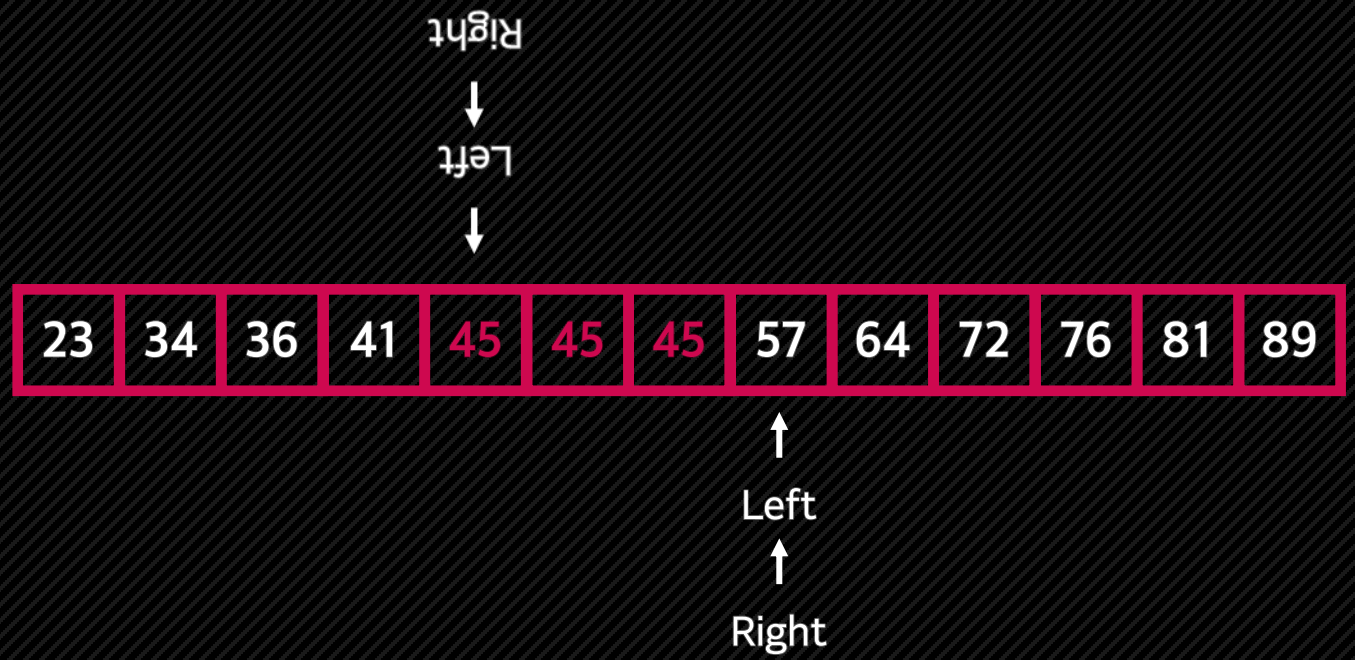
제약조건

- 배열의 크기 n은 $1 <= n <= 4,000$ 이다.
- 배열에 들어있는 정수의 절댓값은 2^{28} 이하 이다.

접근

- 일단 완전탐색부터 생각해보자.
  - → 4,000 * 4,000 * 4,000 * 4,000…
- A[a] + B[b] + C[c] + D[d] = 0 이래.
  - → 식을 살짝 바꾸면 A[a] + B[b] = -(C[c] + D[d]) 겠네?
  - → 가능한 경우의 수는 각각 16,000,000개.
- 식의 앞 부분을 A, 뒷 부분을 B라고 하자.
  - → 그럼 A = -B의 개수를 찾으면 되겠네?
  - → 이분탐색!
  - → 그렇지만 같은 값이 여러 개 일 수 있겠네…

Lower/Upper bound



- Lower Bound: 원하는 값 N 이상이 처음 나오는 위치
- Upper Bound: 원하는 값 N을 처음으로 초과하는 위치

Parametric Search

$$f(x) = x(x - 48)$$

"이 식의 해를 알 수 없을까?"

Parametric Search

$f(x) = x(x - 48)$

Parametric Search

- 이분탐색을 활용하되, 정확한 값이 아닌 가장 근사한 값을 찾는 기법.
- 방정식 f(x)가 있다면, 값에 최대한 근사한 해를 찾는 기법이라고 생각하면 이해가 쉬움!
- 즉, 찾는 값 자체를 비교하는 것이 아닌, 값을 식에 대입하여 나온 결과물을 비교해야 함!

Parametric Search

```
function ParametricSearch(list, target) {
    set left = 0
    set right = last index of list

    while (left < right) do
        set mid = (left + right) / 2
        if (f(mid) is less than target) do
            left = mid + 1
        end else do
            right = mid
        end
    end

    return (left + right) / 2
}
```

23	34	36	41	45	49	52	57	64	72	76	81	89

$f(x) = x(x - 48)$

Silver 3 - 랜선 자르기

요약

- K개의 랜선을 잘라서 서로 같은 길이의 랜선 N개를 만들려고 한다. 이때, N개보다 많이 만들어도 된다.
- 이때, 만들 수 있는 랜선의 최대 길이를 구하는 프로그램을 작성하시오.

제약조건

- 갖고 있는 랜선의 수 K의 범위는 $1 <= K <= 10,000$ 이다.
- 필요한 랜선의 수 N의 범위는 $1 <= M <= 1,000,000$ 이다.
- 각각의 랜선의 길이는 $K_i < 2^{31}$ 인 자연수이다.

Silver 3 - 게임

요약

- 현재 게임을 X판 해서 Y번 승리하였다. (즉, 승률은 Z%라고 할 수 있다.)
- 이후 모든 경기를 승리한다고 할 때, Z의 정수부분이 바뀌기 위해선 최소 몇 번의 게임을 해야하는지 구하는 프로그램을 작성하시오.

제약조건

- X의 범위는 1 <= X <= 1,000,000,000 이다.
- Y의 범위는 1 <= Y <= X 이다.

난이도를 올려보자.

Silver 1 - 기타 레슨

요약

- N개의 레슨 영상을 블루레이에 담으려고 한다. 이때, 반드시 레슨은 순서대로 영상에 담겨야 한다.
- 모든 영상을 담기 위해 M개의 블루레이를 사용하려고 한다. 이때, 블루레이의 길이는 최소로 하고, M개의 블루레이는 길이가 같아야 한다.
- 블루레이 길이의 최솟값을 구하는 프로그램을 작성하시오.

제약조건

- N의 범위는 $1 <= N <= 100,000$ 이다.
- M의 범위는 $1 <= M <= N$ 이다.
- 각 레슨의 길이는 $N_i < 10,000$ 인 자연수이다.

난이도를 올려보자.

Silver 1 - 기타 레슨

요약

- N개의 레슨 영상을 블루레이에 담으려고 한다. 이때, 반드시 레슨은 순서대로 영상에 담겨야 한다.
- 모든 영상을 담기 위해 M개의 블루레이를 사용하려고 한다. 이때, 블루레이의 길이는 최소로 하고, M개의 블루레이는 길이가 같아야 한다.
- 블루레이 길이의 최솟값을 구하는 프로그램을 작성하시오.

제약조건

- N의 범위는 $1 \leq N \leq 100,000$ 이다.
- M의 범위는 $1 \leq M \leq N$ 이다.
- 각 레슨의 길이는 $N_i < 10,000$ 인 자연수이다.

접근

- 블루레이 길이의 범위는?
  - → 최솟값: 1 * 1 = 1, 최댓값: 100,000 * 10,000 = 10,000,000,000
  - → 따라서, 완전 탐색으로는 불가능함.
- 그렇다면 어떻게 블루레이 범위를 찾지?
  - → Parametric Search의 대상을 블루레이의 길이로 하자!
  - → 블루레이 길이의 범위가 int 범위를 초과하니 유의할 것!

도전!

Level 3 - 징검다리 건너기 (카카오 2019 개발자 겨울 인턴십)

요약

- 디딤돌을 밟으면서 징검다리를 건너는데, 한 번 디딤돌을 지나갈 때 마다 적힌 숫자가 1씩 감소한다.
- 밟아야 할 디딤돌에 적힌 숫자가 0일 땐, 건너뛸 수 있는데, 한 번에 최대 K칸 까지 뛰어넘을 수 있다.
- 최대 몇 명까지 징검다리를 건널 수 있는지 구하는 프로그램을 작성하시오.

제약조건

- 징검다리의 길이는 1 <= N <= 10,000 이다.
- 디딤돌에 적힌 숫자의 크기는 1 <= N_i <= 200,000,000 이다.
- K의 범위는 1 <= K <= N 이다.

추가 추천 문제

Silver 1 - 공유기 설치

- 살짝 꼬여있는 이분탐색 문제입니다.

Gold 4 - K번째 수

- N은 100,000 이하입니다. 뭔가 시도할 만한 게 있지 않을까요?

Gold 3 - 드래곤 앤 던전

- 문제의 설명이 길지만, 차근 차근 구현하면 쉽게 해결할 수 있습니다.

Level 3 - 입국심사

- 앞에서 해결한 징검다리 건너기 문제와 유사합니다.

“Any Question?”