알고리즘 특강 이분탐색

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

이분탐색

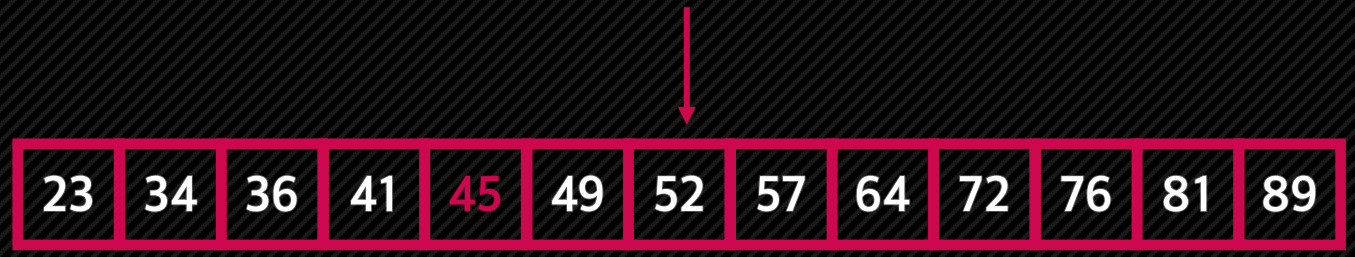
"1에서 100까지 숫자 중에 내가 생각한 수가 뭔지 맞춰봐!"

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



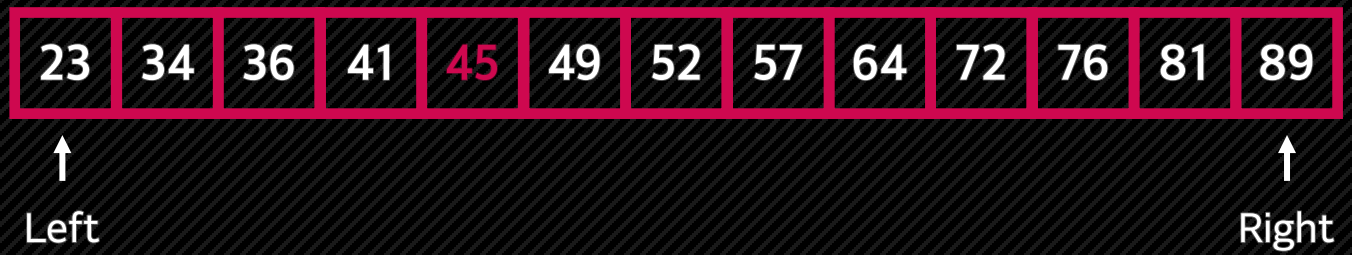
- 핵심은, 가운데다!

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
}
```

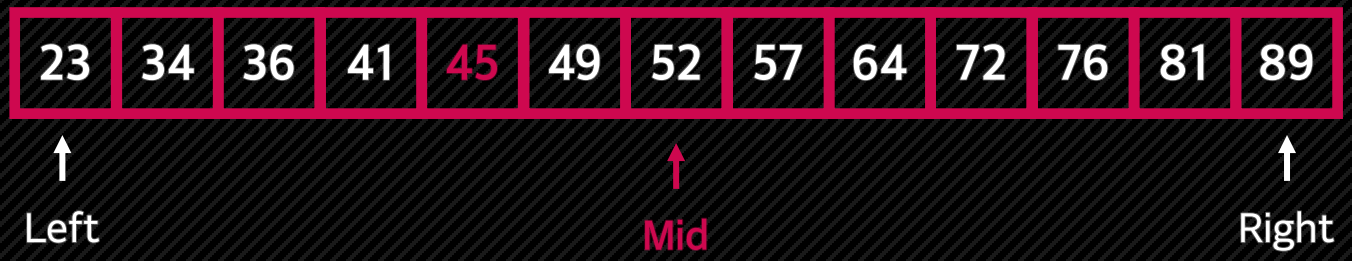


이분탐색

```
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
}
```

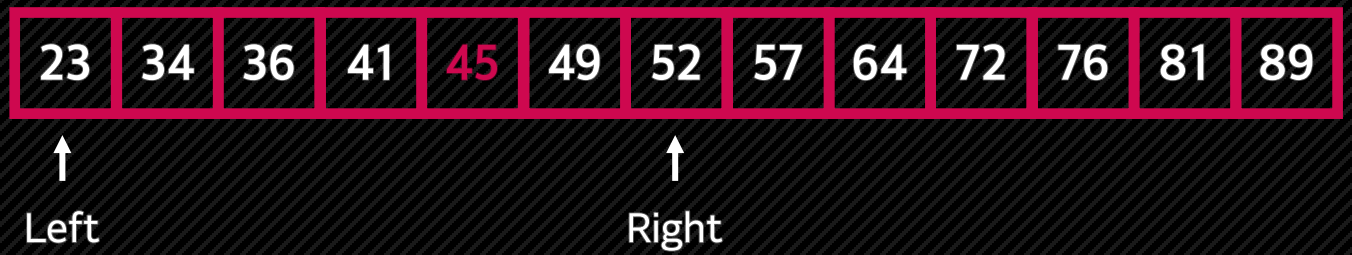


이분탐색

```
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
}
```

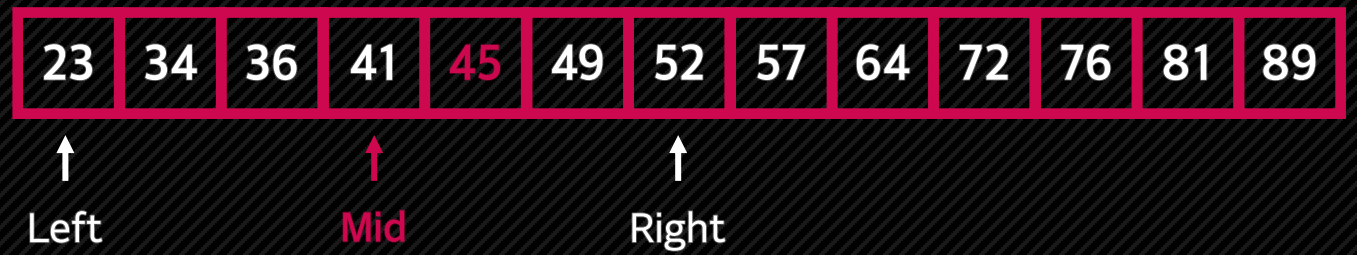


이분탐색

```
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
}
```

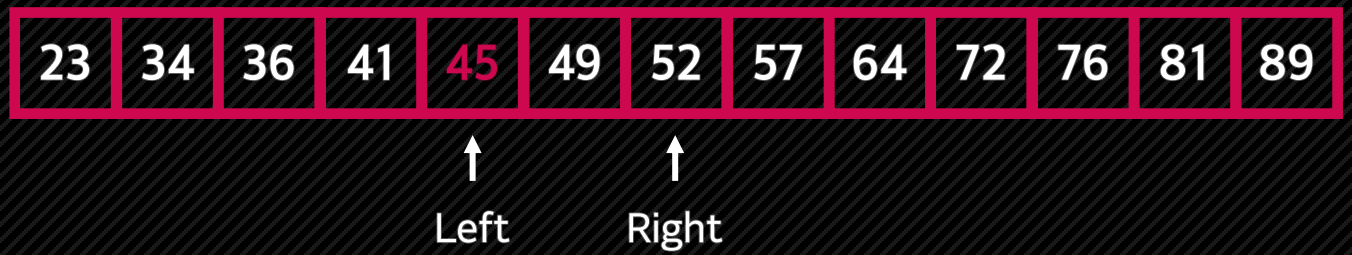


이분탐색

```
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
}
```

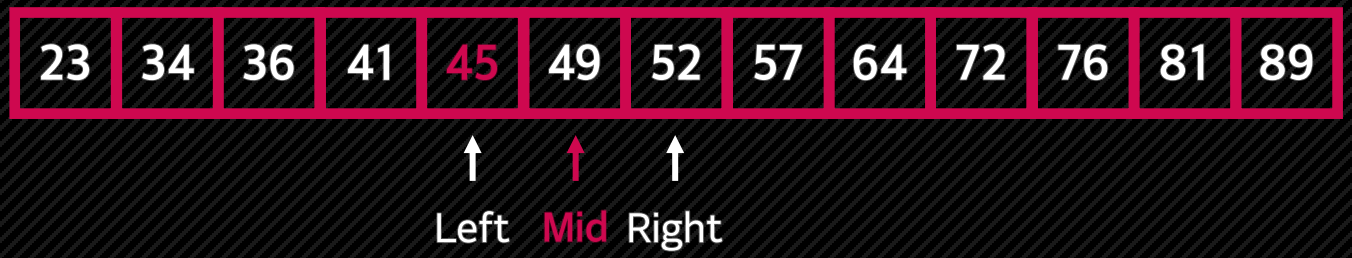


이분탐색

```
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
}
```

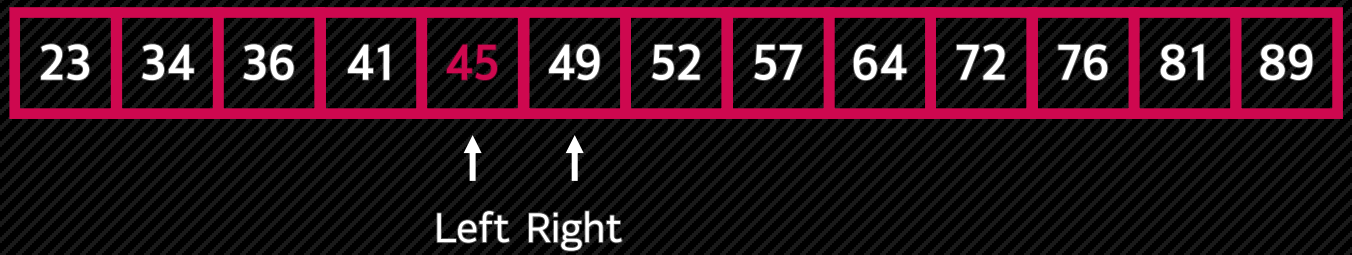


이분탐색

```
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
}
```

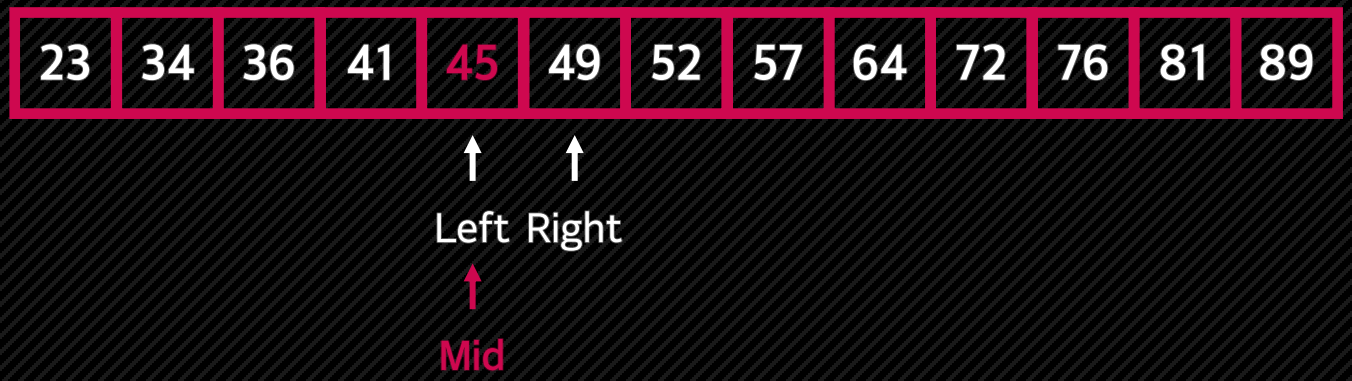


이분탐색

```
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
}
```

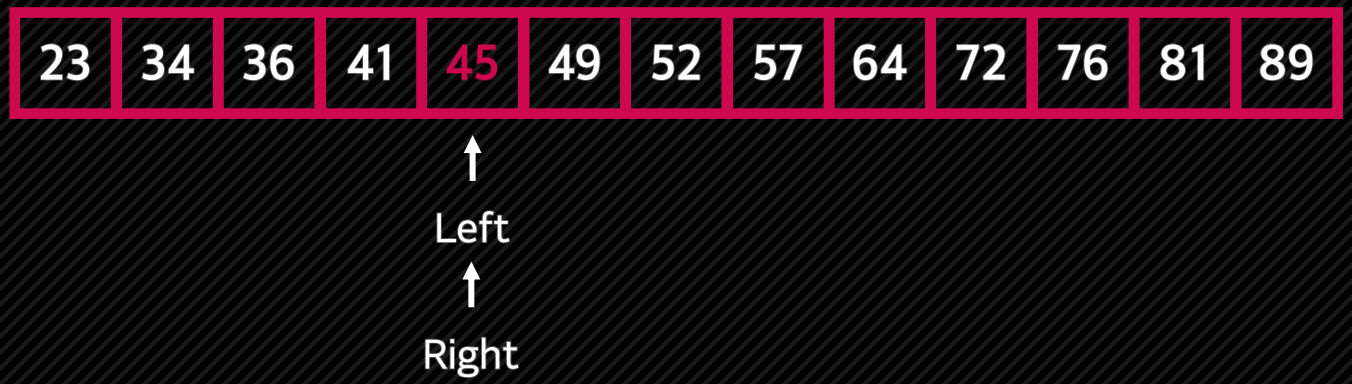


이분탐색

```
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 (#1920)

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.

두개 만들면 되겠네…

Silver 4 - 듣보잡 (#1764)

요약

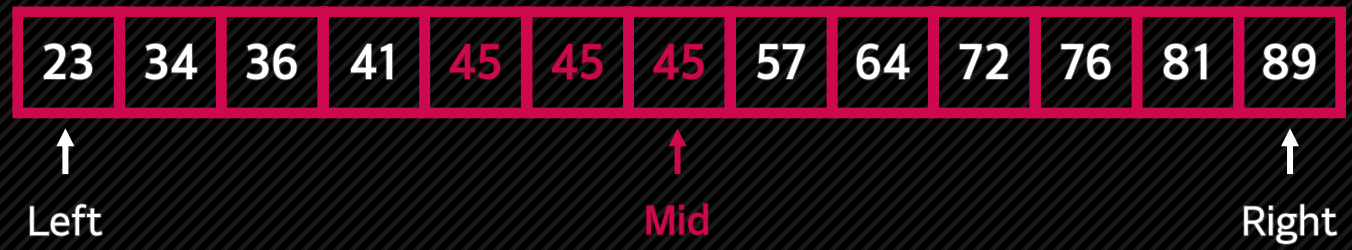
- 듣지도 못한 사람의 명단과, 보지도 못한 사람의 명단이 주어질 때, 듣도 보도 못한 사람의 명단을 구하는 프로그램을 작성하시오.

제약조건

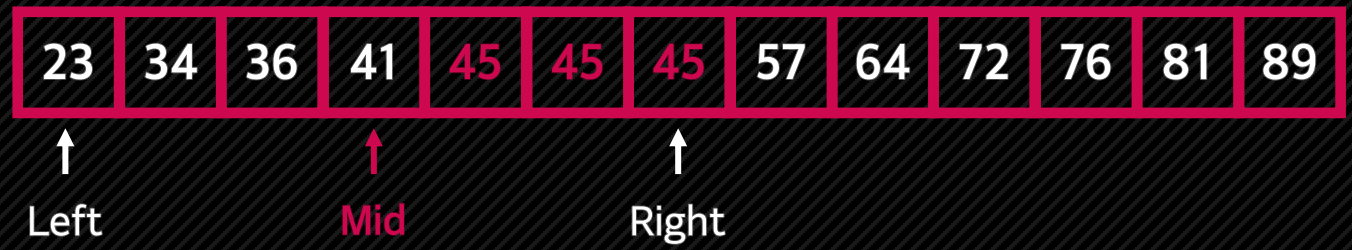
- 듣지도 못한 사람의 범위는 1 <= N <= 500,000 이다.
- 보지도 못한 사람의 범위는 1 <= M <= 500,000 이다.

이런 경우는 어떨까?

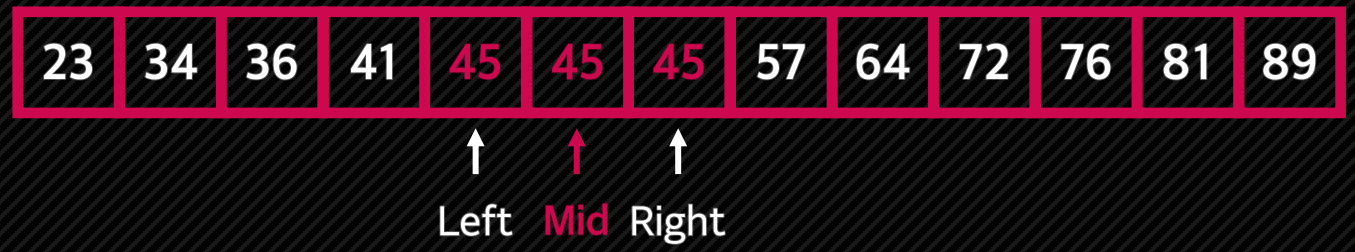
이런 경우는 어떨까?



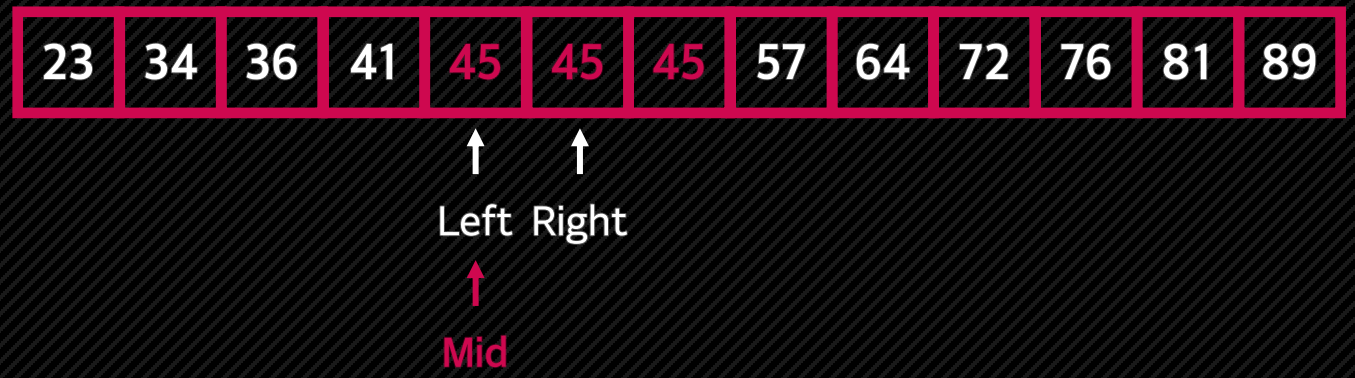
이런 경우는 어떨까?



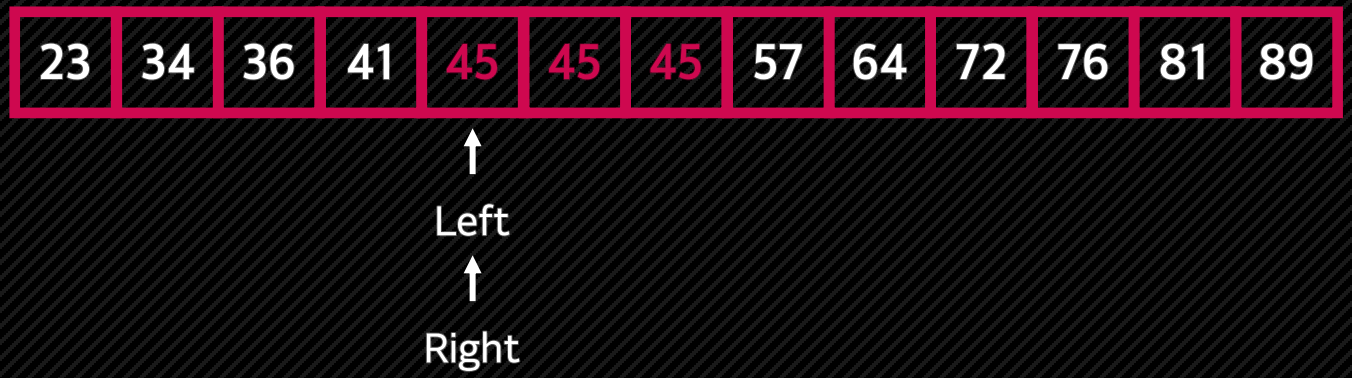
이런 경우는 어떨까?



이런 경우는 어떨까?



이런 경우는 어떨까?

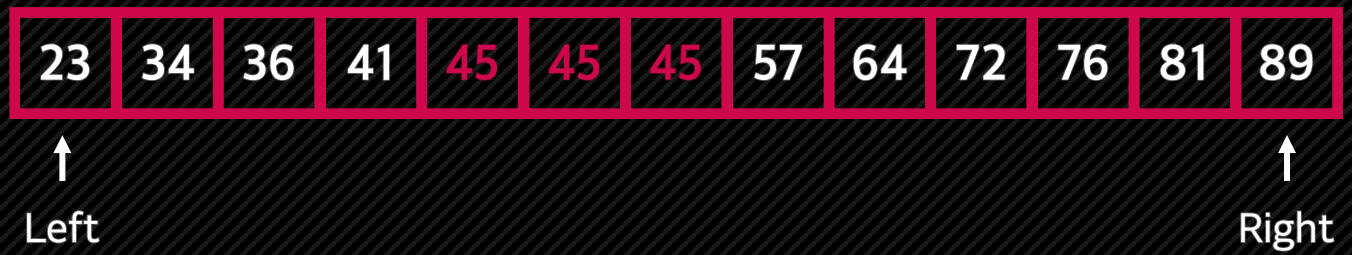


살짝만 알고리즘을 바꿔보자!

```
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
}
```

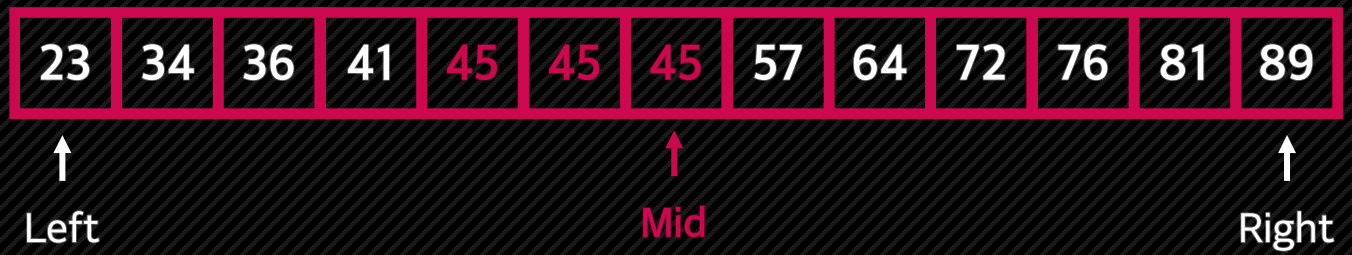


살짝만 알고리즘을 바꿔보자!

```
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
}
```

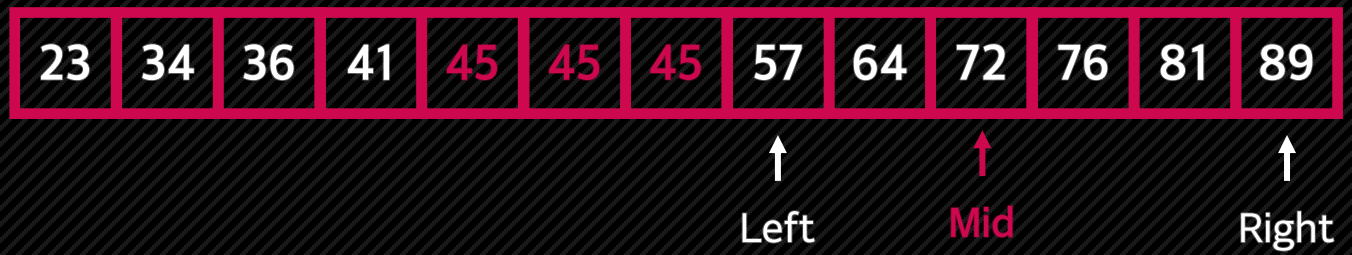


살짝만 알고리즘을 바꿔보자!

```
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
}
```

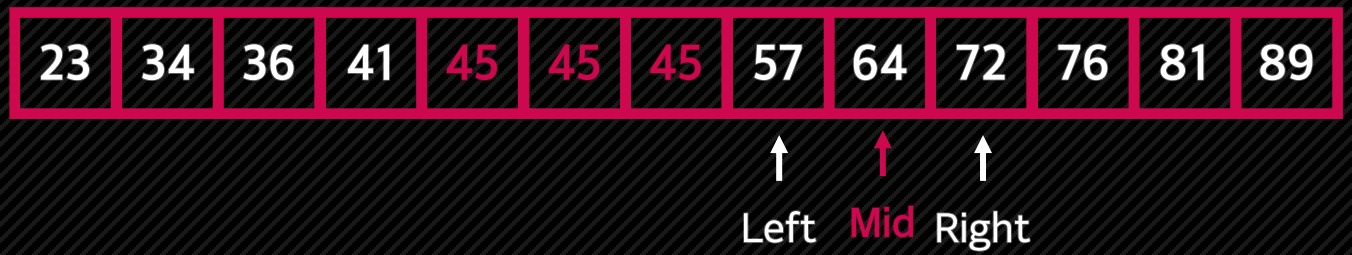


살짝만 알고리즘을 바꿔보자!

```
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
}
```

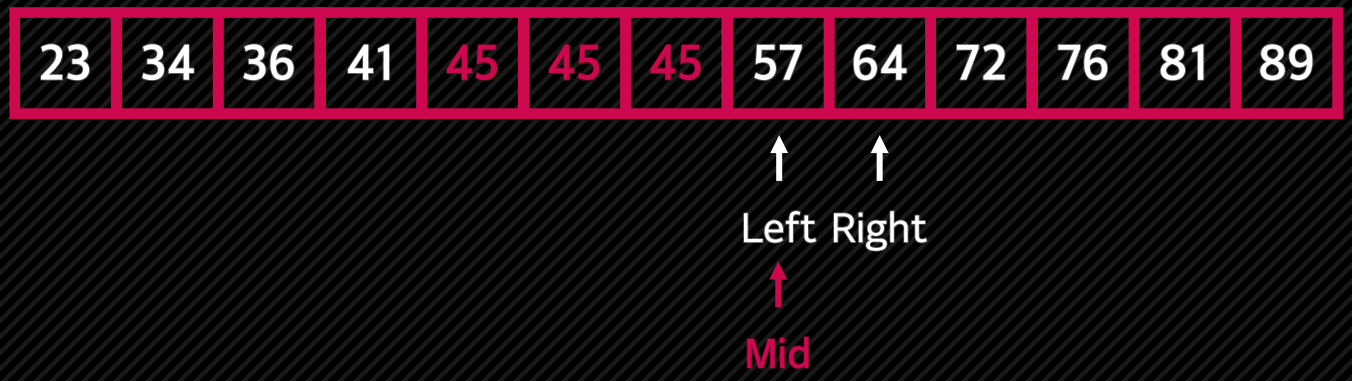


살짝만 알고리즘을 바꿔보자!

```
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
}
```

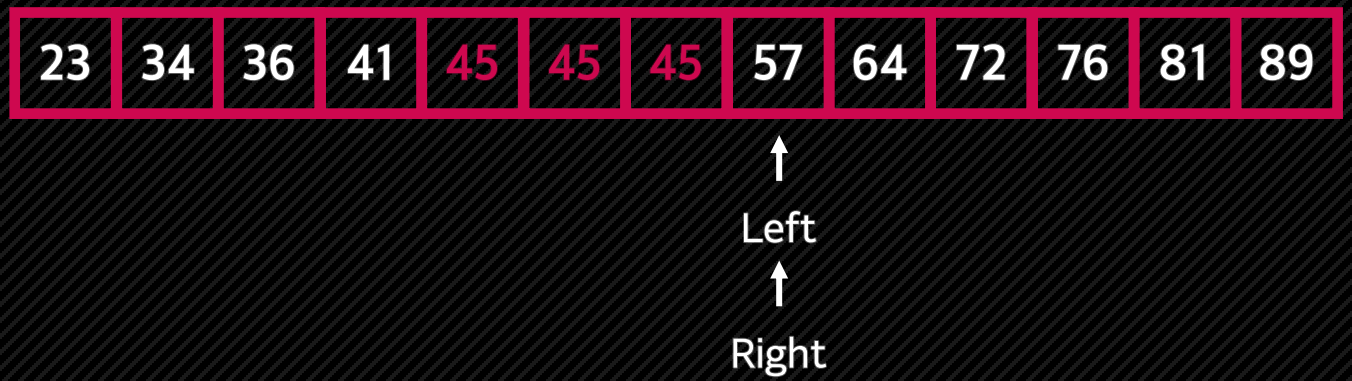


살짝만 알고리즘을 바꿔보자!

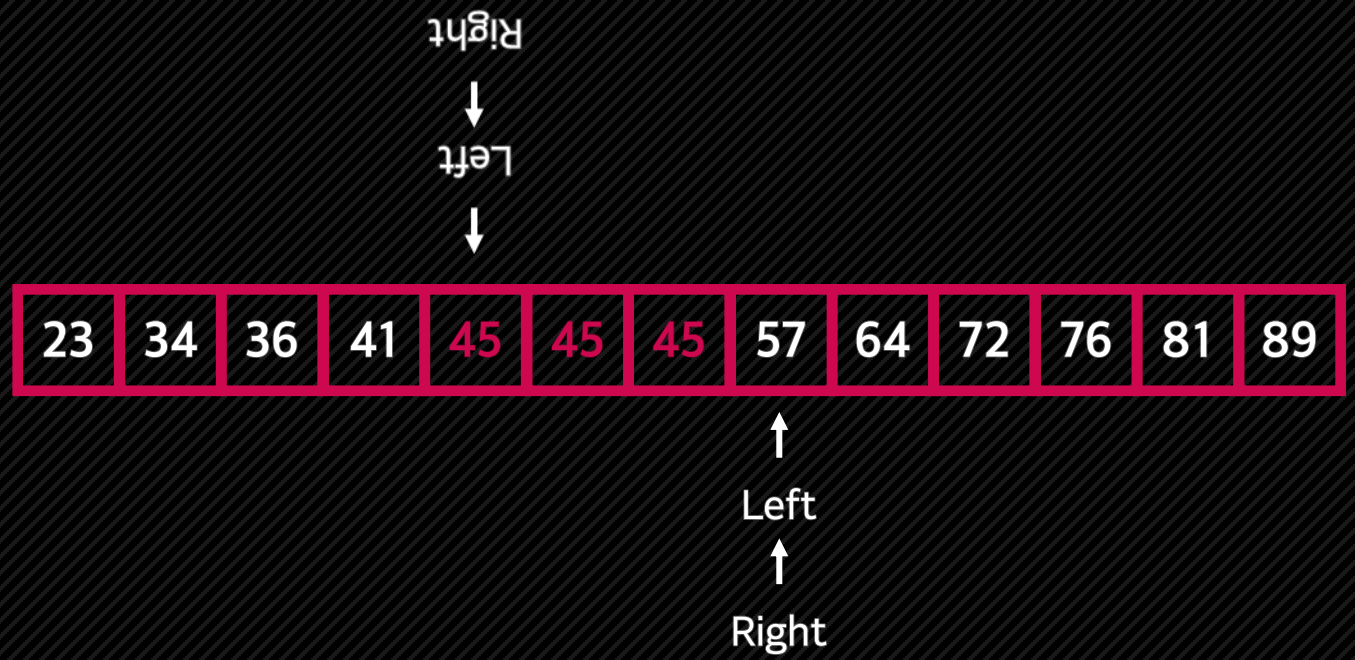
```
function BinarySearch(list, target) {
    set left = 0
    set right = last index of list

    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

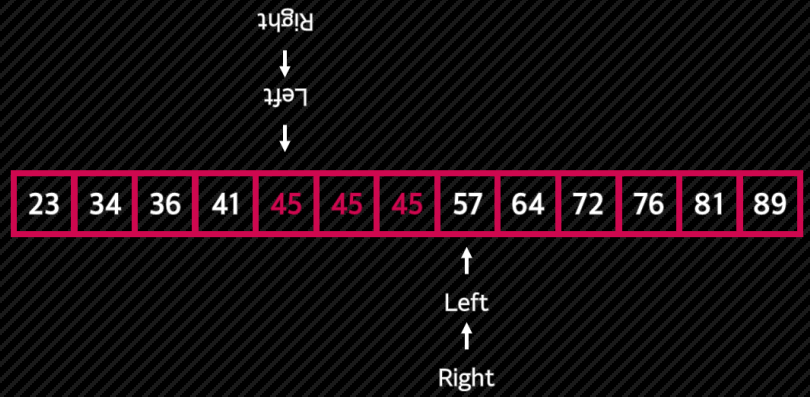
Lower/Upper bound

```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))
```



연습문제

Silver 4 - 숫자 카드 2 (#10816)

요약

- 각각의 카드에 정수가 하나 적혀있고, N장의 카드가 있다.
- 정수 M개가 주어졌을 때, 이 수가 적혀있는 카드를 몇 개 가지고 있는지 구하는 프로그램을 작성하시오.

제약조건

- N의 범위는 $1 <= N <= 500,000$ 이다.
- M의 범위는 $1 <= M <= 500,000$ 이다.
- 모든 수의 범위는 $-10,000,000 < N_i, M_i < 10,000,000$ 이다.

혹시…

사실...

Gold 2 - 합이 0인 네 정수 (#7453)

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.

사실…

Gold 2 - 합이 0인 네 정수 (#7453)

★ Python Naive한 풀이 기준. 컷팅 제대로 한다면…

이진탐색 풀이

해시 풀이

투포인터 풀이

~~이진탐색 풀이~~

해시 풀이

투포인터 풀이

아니 그럼, 왜 쓰는거에요?

```
_list = [17, 22, 25, 29, 35, 42, 46, 49, 53, 65]
```

“30 이상 50 이하인 데이터의 수는?”

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!

문제로 배워보자.

Silver 3 - 랜선 자르기 (#1654)

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.

직접적으로 구하는게 아니라

Silver 3 - 랜선 자르기 (#1654)

```
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
}
```

- 이 문제에서의 target의 값은?

직접적으로 구하는게 아니라

Silver 3 - 랜선 자르기 (#1654)

```
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
}
```

- 이 문제에서의 target의 값은? - **필요한 랜선의 수!**
- 그렇다면… “the mid-th value of list is smaller than target”에서 mid와 target의 비교는 가능한가?

직접적으로 구하는게 아니라

Silver 3 - 랜선 자르기 (#1654)

```
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
}
```

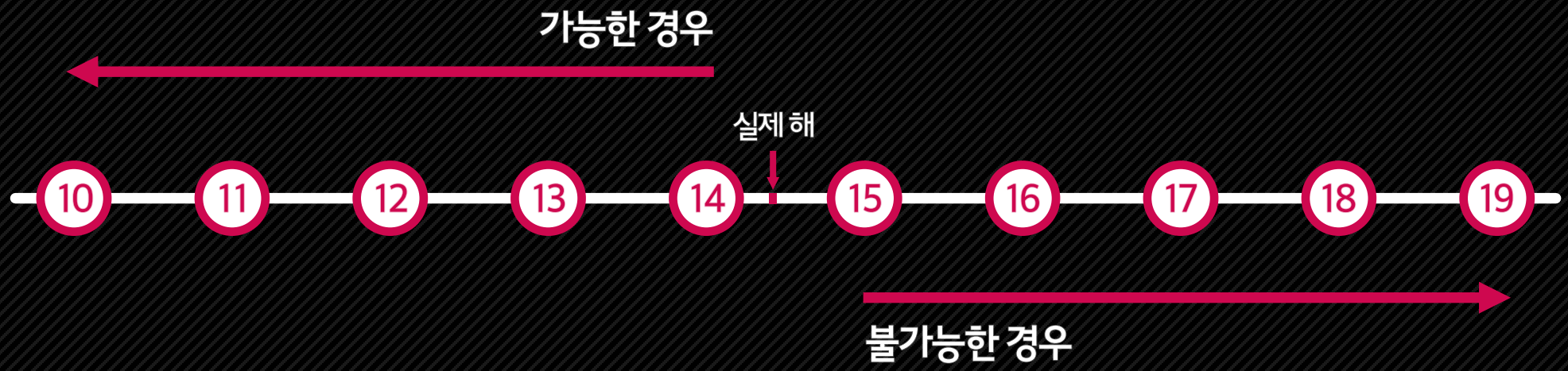
- 이 문제에서의 target의 값은? - **필요한 랜선의 수!**
- 그렇다면… “the mid-th value of list is smaller than target”에서 mid와 target의 비교는 가능한가?
- target과의 비교를 위해, mid 값을 기반으로 해당 길이에서 얻어지는 랜선의 수를 구해 줄 필요가 있음.

범위 문제

Silver 3 - 랜선 자르기 (#1654)

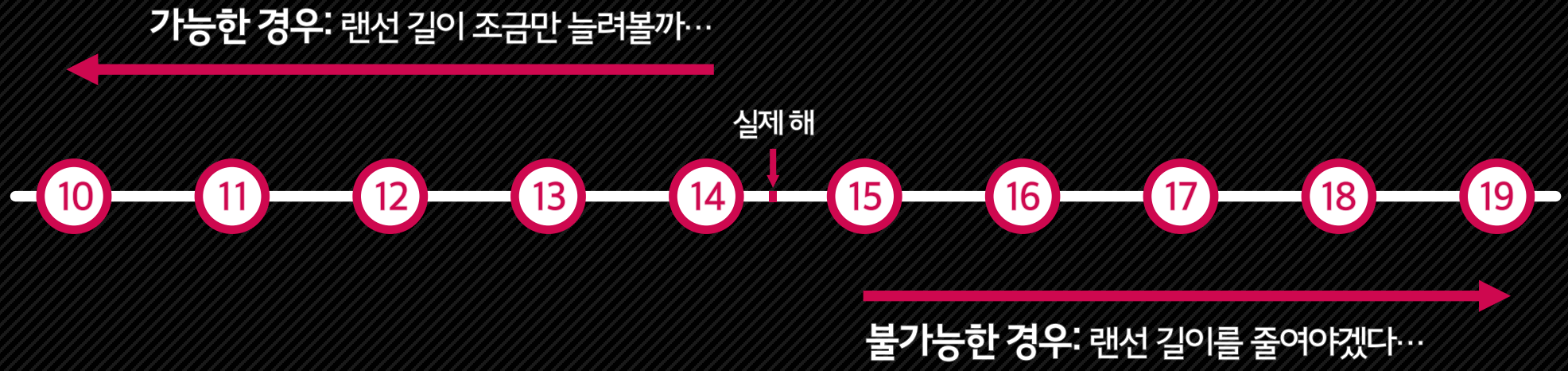
경우를 나눠보자.

Silver 3 - 랜선 자르기 (#1654)



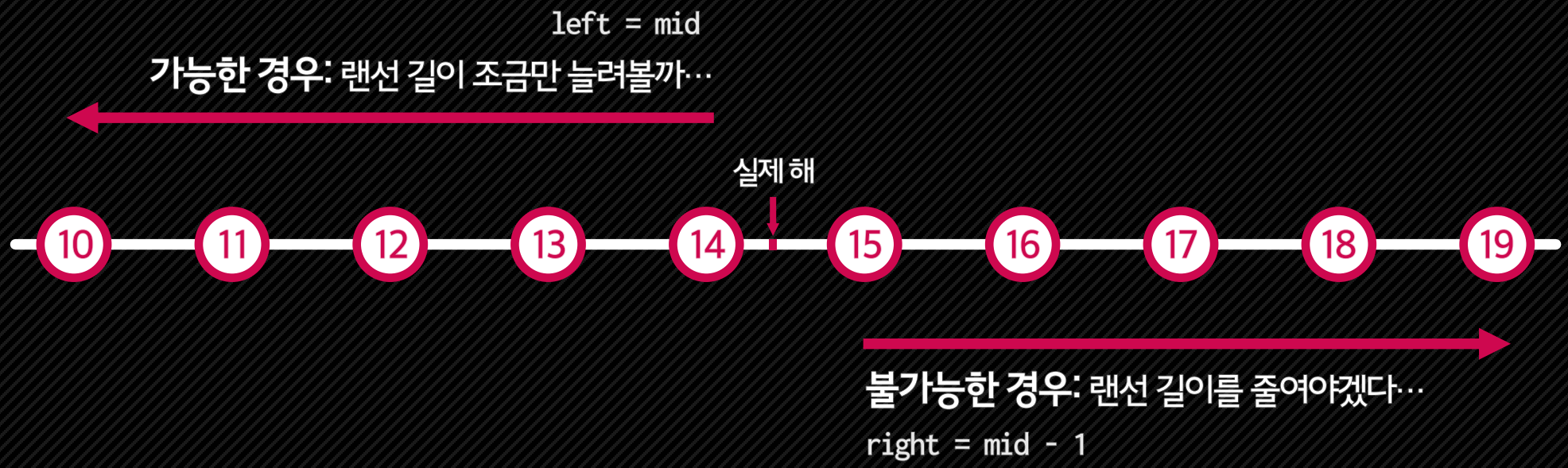
경우를 나눠보자.

Silver 3 - 랜선 자르기 (#1654)



경우를 나눠보자.

Silver 3 - 랜선 자르기 (#1654)



근데 이러면…

Silver 3 - 랜선 자르기 (#1654)

left = mid

가능한 경우: 랜선 길이 조금만 늘려볼까…


```
mid = (left + right) // 2 # 14
if func(mid) >= N:
    left = mid
else:
    right = mid - 1
```

불가능한 경우: 랜선 길이를 줄여야겠다…

```
right = mid - 1
```

근데 이러면…

Silver 3 - 랜선 자르기 (#1654)

left = mid = 14

가능한 경우: 랜선 길이 조금만 늘려볼까…


```
mid = (left + right) // 2 # 14
if func(mid) >= N:
    left = mid
else:
    right = mid - 1
```

불가능한 경우: 랜선 길이를 줄여야겠다…

```
right = mid - 1
```

왜 무한루프죠…

Silver 3 - 랜선 자르기 (#1654)

```
mid = (left + right) // 2 # 14
if func(mid) >= N:
    left = mid
else:
    right = mid - 1
```

- left와 right의 차이가 1인 경우, mid는 무조건 left를 가리키게 됨.
- 이 경우, left = mid 라는 연산을 사용하면 무조건 무한루프에 빠짐!

1차 해결책

Silver 3 - 랜선 자르기 (#1654)

left = mid + 1

가능한 경우: 랜선 길이 조금만 늘려볼까…


```
mid = (left + right) // 2 # 14
if func(mid) >= N:
    left = mid + 1
else:
    right = mid - 1
```

불가능한 경우: 랜선 길이를 줄여야겠다…

```
right = mid - 1
```

얘도 문제인게…

Silver 3 - 랜선 자르기 (#1654)

`left = mid + 1`

가능한 경우: 랜선 길이 조금만 늘려볼까…


```
mid = (left + right) // 2 # 14
if func(mid) >= N:
    left = mid + 1
else:
    right = mid - 1
```

불가능한 경우: 랜선 길이를 줄여야겠다…

`right = mid - 1`

얘도 문제인게…

Silver 3 - 랜선 자르기 (#1654)

left = mid + 1 = 15

가능한 경우: 랜선 길이 조금만 늘려볼까…


```
mid = (left + right) // 2 # 14
if func(mid) >= N:
    left = mid + 1
else:
    right = mid - 1
```

불가능한 경우: 랜선 길이를 줄여야겠다…

```
right = mid - 1
```

이 부분도 보완하면…

Silver 3 - 랜선 자르기 (#1654)

left = mid + 1

가능한 경우: 랜선 길이 조금만 늘려볼까…


```
mid = (left + right) // 2 # 14
if func(mid) >= N:
    left = mid + 1
    answer = mid # 가능한 경우 저장
else:
    right = mid - 1
```

불가능한 경우: 랜선 길이를 줄여야겠다…

```
right = mid - 1
```

결과는?

Silver 3 - 랜선 자르기 (#1654)

left = mid + 1 = 15

가능한 경우: 랜선 길이 조금만 늘려볼까…

answer = 14


```
mid = (left + right) // 2 # 14
if func(mid) >= N:
    left = mid + 1
    answer = mid # 가능한 경우 저장
else:
    right = mid - 1
```

불가능한 경우: 랜선 길이를 줄여야겠다…

```
right = mid - 1
```

원리를 정확히 이해하면 난이도 올라가도 할만해요.

Silver 1 - 공유기 설치 (#2110)

요약

- 집 N개가 수직선 위에 있다. 집 여러 개가 같은 좌표를 갖진 않는다.
- 공유기 C개를 설치하려고 하는데, 가장 인접한 공유기 사이의 거리를 가능한 한 크게 하려고 한다.
- 가장 인접한 공유기 사이의 거리를 최대로 하는 프로그램을 작성하시오.

제약조건

- N의 범위는 $1 <= N <= 200,000$ 이다.
- C의 범위는 $2 <= N <= N$ 이다.
- 각 집의 좌표는 $0 <= X_i <= 1,000,000,000$ 인 자연수이다.

원리를 정확히 이해하면 난이도 올라가도 할만해요.

Silver 1 - 공유기 설치 (#2110)

요약

- 집 N개가 수직선 위에 있다. 집 여러 개가 같은 좌표를 갖진 않는다.
- 공유기 C개를 설치하려고 하는데, 가장 인접한 공유기 사이의 거리를 가능한 한 크게 하려고 한다.
- 가장 인접한 공유기 사이의 거리를 최대로 하는 프로그램을 작성하시오.

- 인접한 공유기 사이의 거리를 Parametric Search의 대상으로 두고, "해당 거리를 최소로 잡았을 때 설치한 공유기의 수" 를 계산.
- 공유기 수가 C개 이상이 나오는지 확인해서 left와 right를 이동시키자!

Parametric…
언제 사용하는게 좋을까?

Silver 3 - 랜선 자르기 (#1654)

- 각각의 랜선의 길이는 $K_i < 2^{31}$ 인 자연수이다.

Silver 1 - 공유기 설치 (#2110)

- 각 집의 좌표는 $0 <= X_i <= 1{,}000{,}000{,}000$ 인 자연수이다.

- 구해야 할 값의 범위가 매우 크다면, (즉, 완전탐색으로 계산할 수 없다면) 고려해보자!
- 일부 문제의 경우 저런 식으로 값을 대놓고 크게 주지 않으니, 유의할 것.

많이 어렵지만,
확실히 짚고 갑시다!

Gold 3 - K번째 수 (#1300)

요약

- 크기가 N * N인 배열 A가 있고, A[i][j] = i * j 이다.
- 이 배열에 있는 수들을 일차원 배열 B에 넣고 오름차순 할 때, B[k]의 값을 구하는 프로그램을 작성하시오.

제약조건

- N의 범위는 1 <= N <= 100,000 이다.

많이 어렵지만,
확실히 짚고 갑시다!

Gold 3 - K번째 수 (#1300)

Q. 수의 범위는?

$1 \sim 10^{10}$

많이 어렵지만,
확실히 짚고 갑시다!

Gold 3 - K번째 수 (#1300)

Q. 수의 범위는?

$1 \sim 10^{10}$

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.

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

도전!

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

문제에선 한 명 씩 건너가지만, 우리는 모두 동시에 건너가게 할거다!

“Any Question?”