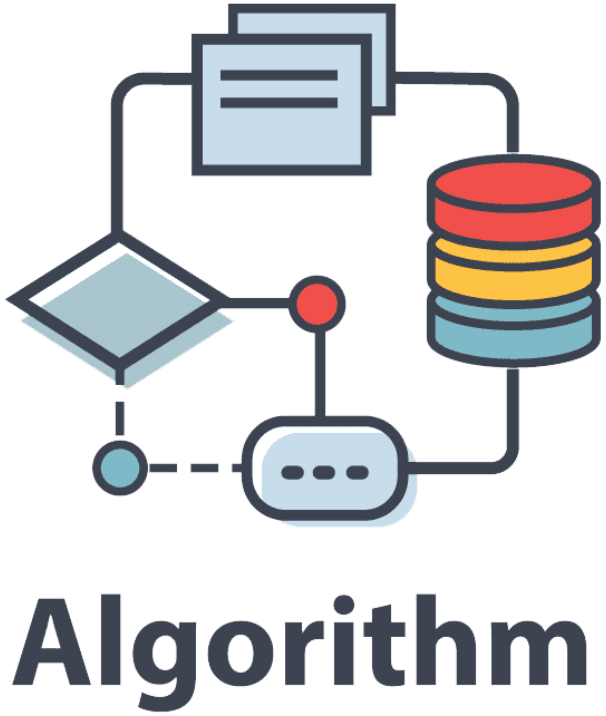
이분 탐색 알고리즘



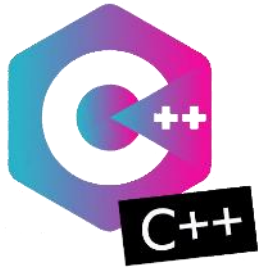
오늘 할 것

01 이분 탐색

02 lower-bound

03 upper-bound

04 문제 예시



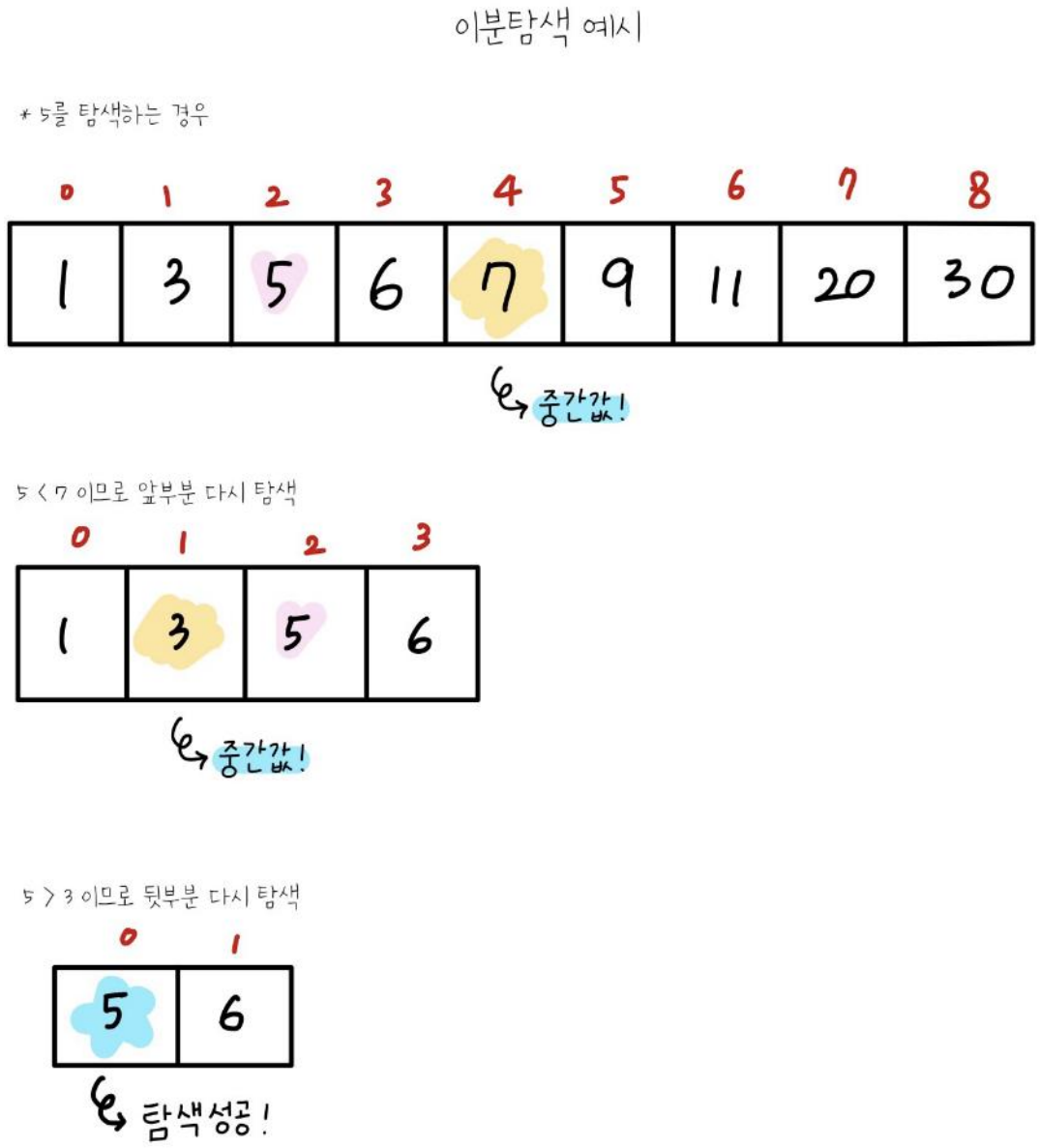
이분 탐색

정렬된 데이터 리스트에서 데이터를 찾을 때,

구간을 절반씩 나누면서 탐색

➔ 데이터가 N 개면 최대 $log_2 N$ 번의 탐색 수행

! 데이터가 정렬되어 있어야 함



이분 탐색

중복 데이터가 있는 경우

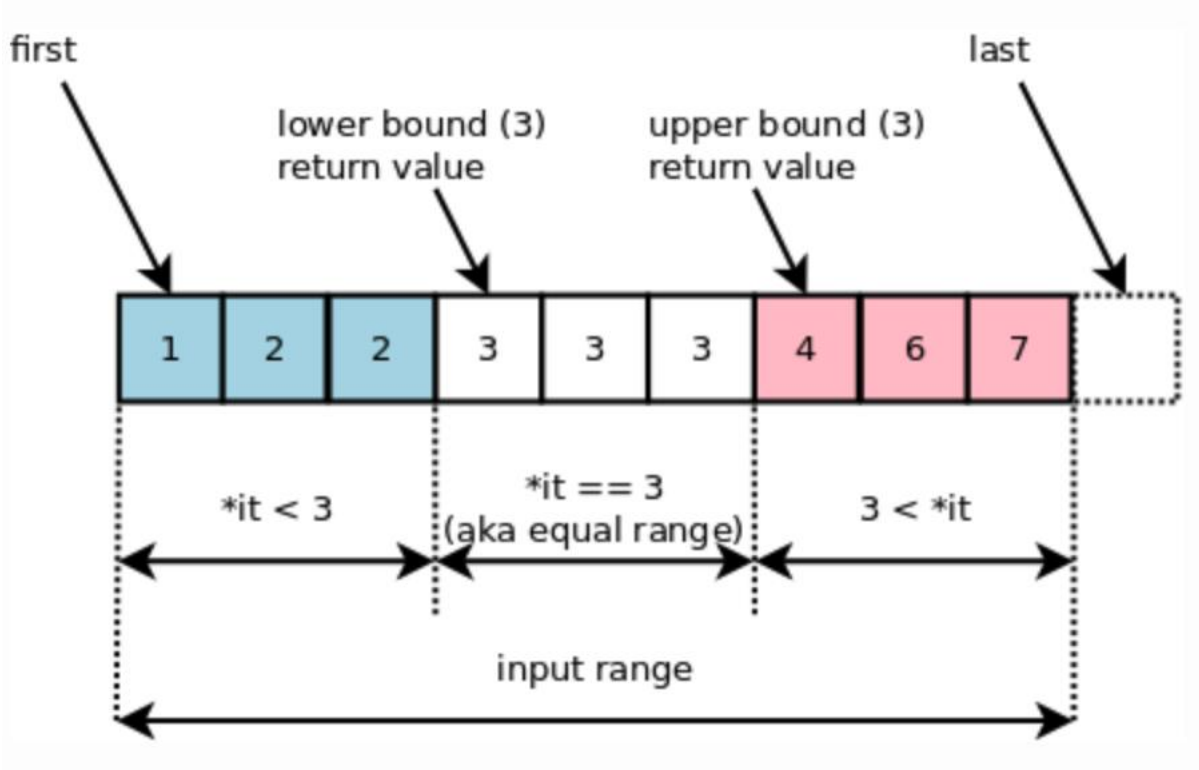
찾는 두 가지 방식

lower-bound:
1. 데이터가 처음 나오는 위치.

2. 데이터 정렬을 유지하면서,
특정 값 **k** 를 삽입할 수 있는 가장 앞 쪽 위치.

upper-bound:
1. 데이터를 초과한 값이 처음 나오는 위치.

2. 데이터 정렬을 유지하면서,
특정 값 **k** 를 삽입할 수 있는 가장 뒤 쪽 위치.



lower-bound

- 초기 **high** 값은 마지막 인덱스 **+ 1**
- 현재 값이 **target** 보다 작거나 같으면 아래쪽 구간을 탐색하므로 **lower-bound**
- 반복문이 끝났을 때, **high** 에 남은 값이 결과

```python
n_list = [1, 2, 5, 5, 5, 5, 6, 7, 9, 11, 20, 30]
target = 5

low, high = 0, len(n_list)

while low < high:
    mid = (low + high) // 2

    if n_list[mid] < target:
        low = mid + 1
    else:
        high = mid

print(high)
```

upper-bound

- 초기 **high** 값은 마지막 인덱스 **+ 1**
- 현재 값이 **target** 보다 작거나 같으면 위쪽 구간을 탐색하므로 **upper-bound**
- 반복문이 끝났을 때, **high** 에 남은 값이 결과

```python
n_list = [1, 2, 5, 5, 5, 5, 6, 7, 9, 11, 20, 30]
target = 5

low, high = 0, len(n_list)

while low < high:
    mid = (low + high) // 2

    if n_list[mid] <= target:
        low = mid + 1
    else:
        high = mid

print(high)
```

bisect

Python 의 이분탐색 모듈

- **bisect.bisect_left()** ➔ **lower-bound**
- **bisect.bisect_right()** ➔ **upper-bound**

문제 예시

프로그래머스 / level 3 / 입국 심사

https://programmers.co.kr/learn/courses/30/lessons/43238

“

끝.