이진 탐색

BABO 4기 이론반

23기 분석 심재혁

CONTENTS

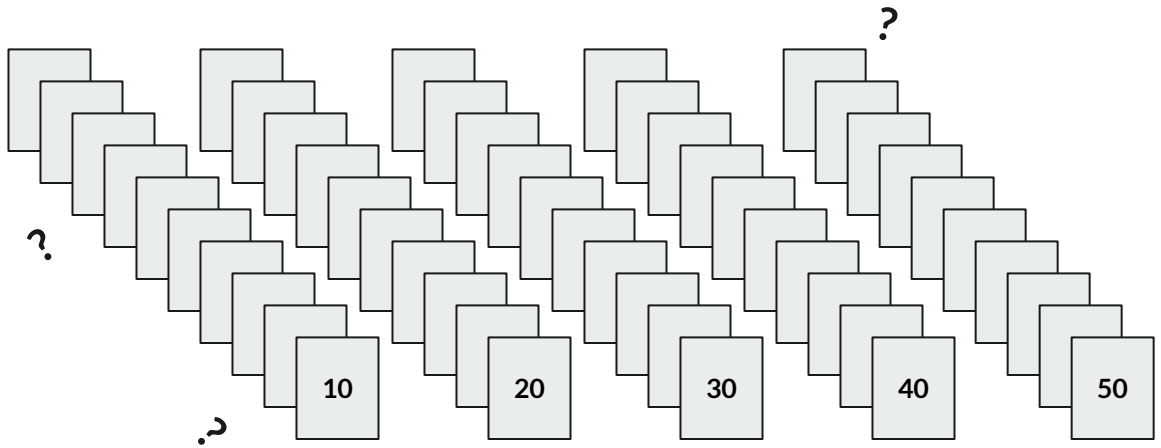
1. Up-Down Game

Up-Down Game

들어보신 분 계신가요..?

Up-Down Game

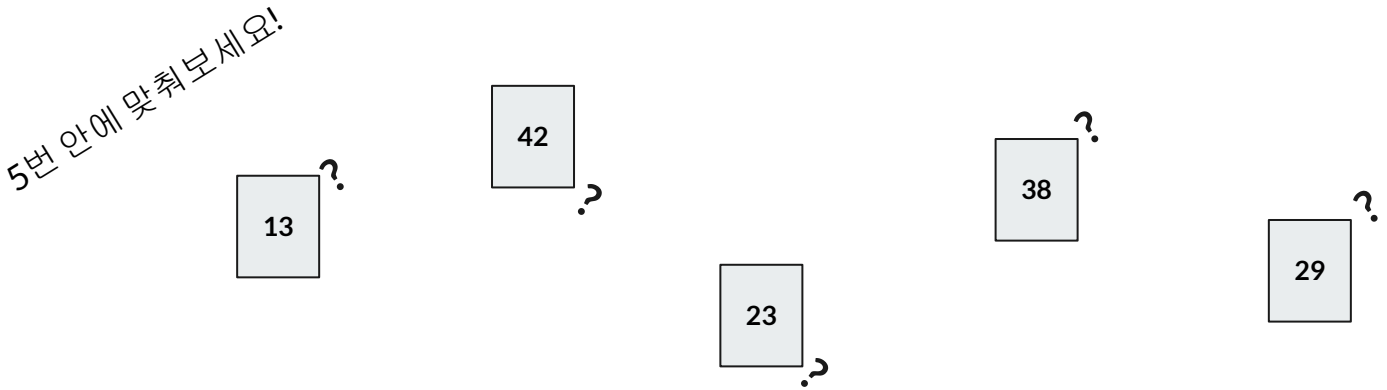
1부터 50까지 숫자 중, 술래가 원하는 숫자를 제한된 횟수안에 맞추는 게임



지금 제가 생각하고 있는 숫자는 몇일까요?

Up-Down Game

1부터 50까지 숫자 중, 술래가 원하는 숫자를 제한된 횟수안에 맞추는 게임



Up-Down Game

1부터 50까지 숫자 중, 술래가 원하는 숫자를 제한된 횟수안에 맞추는 게임

50은 2^5보다 크고 2^6보다 작으므로,

이진탐색을 이용하여 이론상 6번의 기회안에 무조건 정답을 맞출 수 있음

Up-Down Game

1부터 50까지 숫자 중, 술래가 원하는 숫자를 제한된 횟수안에 맞추는 게임

술래가 생각한 숫자 : 14

현재 선택할 수 있는 수의 범위 : [1, 50]

내가 선택 해야 할 수 : 25

내 대답 횟수 : 1

술래의 대답 : Down!

Up-Down Game

1부터 50까지 숫자 중, 술래가 원하는 숫자를 제한된 횟수안에 맞추는 게임

술래가 생각한 숫자 : 14

현재 선택할 수 있는 수의 범위 : [1, 24]

내가 선택 해야 할 수 : 12

내 대답 횟수 : 2

술래의 대답 : Up!

Up-Down Game

1부터 50까지 숫자 중, 술래가 원하는 숫자를 제한된 횟수안에 맞추는 게임

술래가 생각한 숫자 : 14

현재 선택할 수 있는 수의 범위 : [13, 24]

내가 선택 해야 할 수 : 18

내 대답 횟수 : 3

술래의 대답 : Down!

Up-Down Game

1부터 50까지 숫자 중, 술래가 원하는 숫자를 제한된 횟수안에 맞추는 게임

술래가 생각한 숫자 : 14

현재 선택할 수 있는 수의 범위 : [13, 17]

내가 선택 해야 할 수 : 15

내 대답 횟수 : 4

술래의 대답 : Down!

Up-Down Game

1부터 50까지 숫자 중, 술래가 원하는 숫자를 제한된 횟수안에 맞추는 게임

술래가 생각한 숫자 : 14

현재 선택할 수 있는 수의 범위 : [13, 14]

내가 선택 해야 할 수 : 13

내 대답 횟수 : 5

술래의 대답 : Up!

Up-Down Game

1부터 50까지 숫자 중, 술래가 원하는 숫자를 제한된 횟수안에 맞추는 게임

술래가 생각한 숫자 : 14

현재 선택할 수 있는 수의 범위 : [14, 14]

내가 선택 해야 할 수 : 14

내 대답 횟수 : 6

술래의 대답 : Bingo!

2. 이진탐색 (Binary Search)

이진탐색

정렬된 리스트에서 검색 범위를 반으로 줄여나가며 원하는 데이터를 검색하는 알고리즘

이진탐색

정렬된 리스트에서 검색 범위를 반으로 줄여나가며 원하는 데이터를 검색하는 알고리즘

Q. 왜 조건에 **정렬된 리스트**가 있을까요?

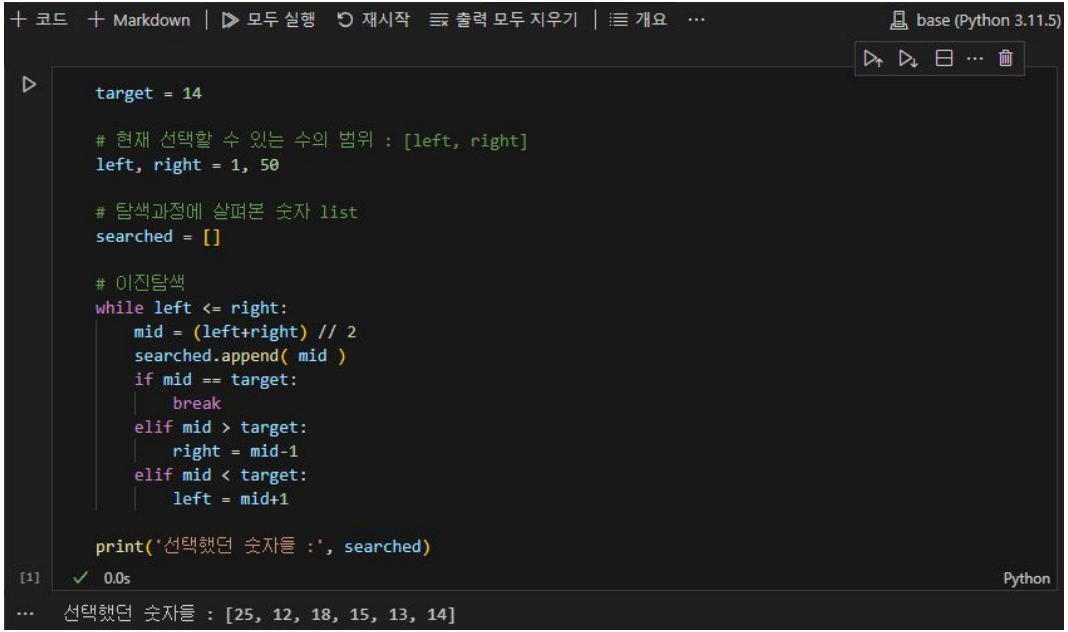
이진탐색

정렬된 리스트에서 검색 범위를 반으로 줄여나가며 원하는 데이터를 검색하는 알고리즘

Q. 왜 조건에 **정렬된 리스트**가 있을까요?

A. 검색 분기 조건이 **더 큰지 or 작은지**이기 때문!

이진탐색


+ 코드
+ Markdown
모두 실행
재시작
출력 모두 지우기
개요
base (Python 3.11.5)
target = 14
# 현재 선택할 수 있는 수의 범위 : [left, right]
left, right = 1, 50
# 탐색과정에 살펴본 숫자 list
searched = []
# 이진탐색
while left <= right:
mid = (left+right) // 2
searched.append(mid)
if mid == target:
break
elif mid > target:
right = mid-1
elif mid < target:
left = mid+1
print('선택했던 숫자들 :', searched)
[1]
0.0s
Python
선택했던 숫자들 : [25, 12, 18, 15, 13, 14]

이진탐색

전체 리스트를 차례대로 살펴본다면 O(n)이지만,

이진탐색이 사용가능한 경우 O(log n)으로 시간복잡도를 줄여줄 수 있다.

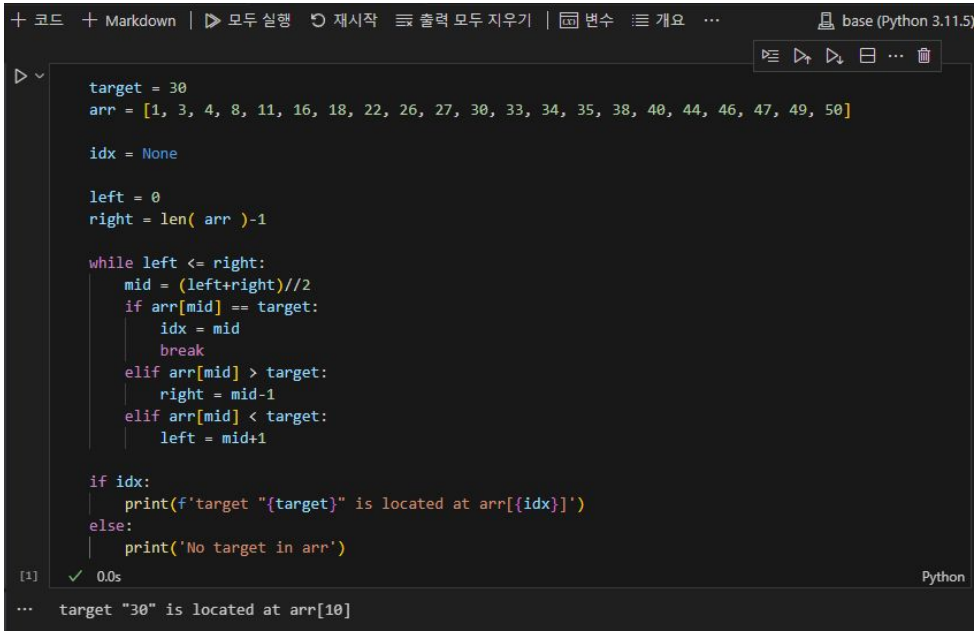
3. 이진탐색 대표유형

1. 특정 숫자 빠르게 찾기

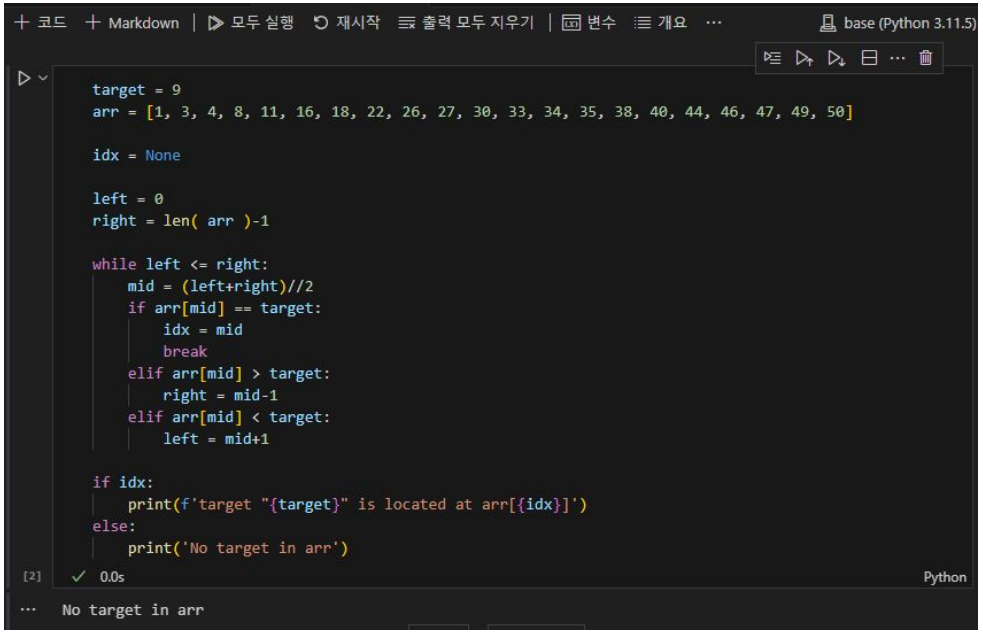
[주어지는 조건]
찾고자하는 숫자(target), 숫자들의 배열(arr)

[로직]
1. 배열의 인덱스로 대소비교를 하며 이진탐색 진행
2. 이진탐색 중 target을 만나면 탐색 종료 후 해당 숫자가 있는 인덱스 반환 -> target의 위치
3. 만약 left==right조건으로 탐색종료 시, '찾고자하는 숫자 없음'으로 판단

1. 특정 숫자 빠르게 찾기


+ 코드
+ Markdown
모두 실행
재시작
출력 모두 지우기
변수
개요
base (Python 3.11.5)
target = 30
arr = [1, 3, 4, 8, 11, 16, 18, 22, 26, 27, 30, 33, 34, 35, 38, 40, 44, 46, 47, 49, 50]
idx = None
left = 0
right = len(arr)-1
while left <= right:
mid = (left+right)//2
if arr[mid] == target:
idx = mid
break
elif arr[mid] > target:
right = mid-1
elif arr[mid] < target:
left = mid+1
if idx:
print(f'target "{target}" is located at arr[{idx}]')
else:
print('No target in arr')
[1]
0.0s
Python
target "30" is located at arr[10]

1. 특정 숫자 빠르게 찾기


+ 코드
+ Markdown
모두 실행
재시작
출력 모두 지우기
변수
개요
base (Python 3.11.5)
target = 9
arr = [1, 3, 4, 8, 11, 16, 18, 22, 26, 27, 30, 33, 34, 35, 38, 40, 44, 46, 47, 49, 50]
idx = None
left = 0
right = len(arr)-1
while left <= right:
mid = (left+right)//2
if arr[mid] == target:
idx = mid
break
elif arr[mid] > target:
right = mid-1
elif arr[mid] < target:
left = mid+1
if idx:
print(f'target "{target}" is located at arr[{idx}]')
else:
print('No target in arr')
[2]
0.0s
Python
No target in arr

2. 특정 숫자 개수 세기

[주어지는 조건]
찾고자하는 숫자(target), 숫자들의 배열(arr)

[로직]

1. target **이상의** 값이 최초로 나오는 위치를 찾음. target이 있다면 target들의 가장 앞 idx를 찾고 아닌경우 target보다 크지만 target과 가장 적게 차이나는 숫자의 idx를 찾음만약 left==right조건으로 탐색종료 시, '찾고자하는 숫자 없음'으로 판단
2. target **초과의** 값이 최초로 나오는 위치를 찾음.
3. 따라서 1에서 target이 arr에 없다면 1과 2의 결과가 같으므로 target이 0개 있다고 판단.
4. 1에서 target이 arr에 있다면 2 - 1의 연산으로 target의 개수를 판단.

2. 특정 숫자 개수 세기

```
target = 7
arr = [1, 1, 2, 2, 2, 3, 4, 5, 5, 6, 7, 7, 7, 7, 8, 9, 10, 10]
```

```python
def lower_bound(target, arr):
    left = 0
    right = len(arr)-1
    min_idx = len(arr)

    while left <= right:
        mid = (left+right) // 2
        if arr[mid] >= target:
            right = mid-1
            min_idx = min([mid, min_idx])
        else:
            left = mid+1

    return min_idx
```

✓ 0.0s Python

2. 특정 숫자 개수 세기

```
target = 7
arr = [1, 1, 2, 2, 2, 3, 4, 5, 5, 6, 7, 7, 7, 7, 8, 9, 10, 10]
```

```python
def lower_bound(target, arr):
    left = 0
    right = len(arr)-1
    min_idx = len(arr)

    while left <= right:
        mid = (left+right) // 2
        if arr[mid] >= target:
            right = mid-1
            min_idx = min([mid, min_idx])
        else:
            left = mid+1

    return min_idx
```

✓ 0.0s Python

2. 특정 숫자 개수 세기

```
target = 7
arr = [1, 1, 2, 2, 2, 3, 4, 5, 5, 6, 7, 7, 7, 7, 8, 9, 10, 10]
```

```python
def lower_bound(target, arr):
    left = 0
    right = len(arr)-1
    min_idx = len(arr)

    while left <= right:
        mid = (left+right) // 2
        if arr[mid] >= target:
            right = mid-1
            min_idx = min([mid, min_idx])
        else:
            left = mid+1

    return min_idx
```

✓ 0.0s Python

2. 특정 숫자 개수 세기

```
target = 7
arr = [1, 1, 2, 2, 2, 3, 4, 5, 5, 6, 7, 7, 7, 7, 8, 9, 10, 10]
```

```python
def lower_bound(target, arr):
    left = 0
    right = len(arr)-1
    min_idx = len(arr)

    while left <= right:
        mid = (left+right) // 2
        if arr[mid] >= target:
            right = mid-1
            min_idx = min([mid, min_idx])
        else:
            left = mid+1

    return min_idx
```

✓ 0.0s Python

2. 특정 숫자 개수 세기

```
target = 7
arr = [1, 1, 2, 2, 2, 3, 4, 5, 5, 6, 7, 7, 7, 7, 8, 9, 10, 10]
```

```python
def lower_bound(target, arr):
    left = 0
    right = len(arr)-1
    min_idx = len(arr)

    while left <= right:
        mid = (left+right) // 2
        if arr[mid] >= target:
            right = mid-1
            min_idx = min([mid, min_idx])
        else:
            left = mid+1

    return min_idx
```

✓ 0.0s Python

2. 특정 숫자 개수 세기

```
target = 7
arr = [1, 1, 2, 2, 2, 3, 4, 5, 5, 6, 7, 7, 7, 7, 8, 9, 10, 10]
```

```python
def lower_bound(target, arr):
    left = 0
    right = len(arr)-1
    min_idx = len(arr)

    while left <= right:
        mid = (left+right) // 2
        if arr[mid] >= target:
            right = mid-1
            min_idx = min([mid, min_idx])
        else:
            left = mid+1

    return min_idx
```

✓ 0.0s Python

2. 특정 숫자 개수 세기

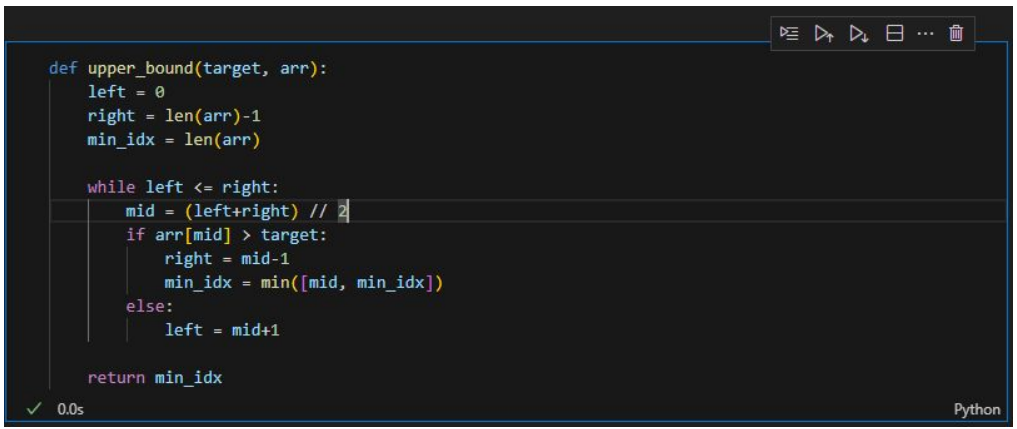
```
target = 7
arr = [1, 1, 2, 2, 2, 3, 4, 5, 5, 6, 7, 7, 7, 7, 8, 9, 10, 10]
```

```python
def upper_bound(target, arr):
    left = 0
    right = len(arr)-1
    min_idx = len(arr)

    while left <= right:
        mid = (left+right) // 2
        if arr[mid] > target:
            right = mid-1
            min_idx = min([mid, min_idx])
        else:
            left = mid+1

    return min_idx
```

0.0s Python

2. 특정 숫자 개수 세기

```
target = 7
arr = [1, 1, 2, 2, 2, 3, 4, 5, 5, 6, 7, 7, 7, 7, 8, 9, 10, 10]
```

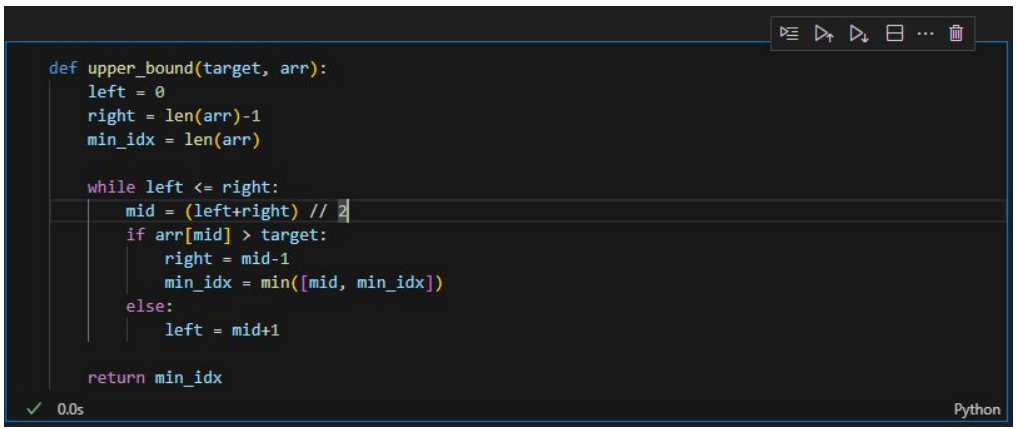


```python
def upper_bound(target, arr):
    left = 0
    right = len(arr)-1
    min_idx = len(arr)

    while left <= right:
        mid = (left+right) // 2
        if arr[mid] > target:
            right = mid-1
            min_idx = min([mid, min_idx])
        else:
            left = mid+1

    return min_idx
```

0.0s Python

2. 특정 숫자 개수 세기

```
target = 7
arr = [1, 1, 2, 2, 2, 3, 4, 5, 5, 6, 7, 7, 7, 7, 8, 9, 10, 10]
```



```python
def upper_bound(target, arr):
    left = 0
    right = len(arr)-1
    min_idx = len(arr)

    while left <= right:
        mid = (left+right) // 2
        if arr[mid] > target:
            right = mid-1
            min_idx = min([mid, min_idx])
        else:
            left = mid+1

    return min_idx
```

0.0s Python

2. 특정 숫자 개수 세기

```
target = 7
arr = [1, 1, 2, 2, 2, 3, 4, 5, 5, 6, 7, 7, 7, 7, 8, 9, 10, 10]
```



```python
def upper_bound(target, arr):
    left = 0
    right = len(arr)-1
    min_idx = len(arr)

    while left <= right:
        mid = (left+right) // 2
        if arr[mid] > target:
            right = mid-1
            min_idx = min([mid, min_idx])
        else:
            left = mid+1

    return min_idx
```

0.0s Python

2. 특정 숫자 개수 세기

```
target = 7
arr = [1, 1, 2, 2, 2, 3, 4, 5, 5, 6, 7, 7, 7, 7, 8, 9, 10, 10]
```

```python
def upper_bound(target, arr):
    left = 0
    right = len(arr)-1
    min_idx = len(arr)

    while left <= right:
        mid = (left+right) // 2
        if arr[mid] > target:
            right = mid-1
            min_idx = min([mid, min_idx])
        else:
            left = mid+1

    return min_idx
```

0.0s Python

2. 특정 숫자 개수 세기

```
target = 7
arr = [1, 1, 2, 2, 2, 3, 4, 5, 5, 6, 7, 7, 7, 7, 8, 9, 10, 10]
```



```python
def upper_bound(target, arr):
    left = 0
    right = len(arr)-1
    min_idx = len(arr)

    while left <= right:
        mid = (left+right) // 2
        if arr[mid] > target:
            right = mid-1
            min_idx = min([mid, min_idx])
        else:
            left = mid+1

    return min_idx
```

0.0s Python

2. 특정 숫자 개수 세기

```
target = 7
arr = [1, 1, 2, 2, 2, 3, 4, 5, 5, 6, 7, 7, 7, 7, 8, 9, 10, 10]
```



```python
def upper_bound(target, arr):
    left = 0
    right = len(arr)-1
    min_idx = len(arr)

    while left <= right:
        mid = (left+right) // 2
        if arr[mid] > target:
            right = mid-1
            min_idx = min([mid, min_idx])
        else:
            left = mid+1

    return min_idx
```

✓ 0.0s Python

2. 특정 숫자 개수 세기

```python
target = 7
arr = [1, 1, 2, 2, 2, 3, 4, 5, 5, 6, 7, 7, 7, 7, 8, 9, 10, 10]

lower = lower_bound(target, arr) # 1
upper = upper_bound(target, arr) # 2

print(upper-lower)
```

✓ 0.0s Python

```
4
```

3. 정수 분배

[주어지는 조건]
나누어지는 그룹의 수(groups), 숫자들의 배열(arr)

[목적]
나누어지는 모든 그룹들의 요소의 합이 최소가 되도록 하기

[로직]
1. 기준 요소의 합(max_sum)을 arr의 합으로 구해서 이를 갖고 이진탐색 진행
2. max_sum을 기준으로 이를 넘기면 그룹을 분할함
3. 최종적으로 그룹의 수가 groups보다 크면 left=mid+1, 작으면 right=mid-1로 진행

3. 정수 분배

```python
groups = 5
arr = [2, 3, 5, 5, 3, 1, 6, 5, 7, 3]
```

```python
def is_possible(max_sum, arr, target_groups):
    curr_sum = 0
    groups_cnt = 1

    for i in arr:
        if i > max_sum:
            return False

        curr_sum += i

        if curr_sum > max_sum:
            groups_cnt += 1
            curr_sum = i

    return groups_cnt <= target_groups
```

0.0s Python

3. 정수 분배

```
groups = 5
arr = [2, 3, 5, 5, 3, 1, 6, 5, 7, 3]
```

```
left = 1
right = sum( arr )
ans = sum( arr )

while left <= right:
    mid = (left+right) // 2
    if is_possible(mid, arr, groups):
        right = mid-1
        ans = min(ans, mid)
    else:
        left = mid+1

print(ans)
```

✓ 0.0s Python

```
10
```

과제

BOJ 1920 수 찾기(S4)
BOJ 16401 과자 나눠주기(S2)
BOJ 2110 공유기 설치(G4)

7주동안 수고 많으셨습니다!