알고리즘 멘토링

- 탐색 알고리즘 -

- 주민찬 -

Sequential Search

- 순차 탐색

0	1	2	3	4	5	6	7	8	9	…
3	4	6	7	8	1	2	9	5	11	…

이 리스트에서 내가 찾고자 하는 값을 지정 ex) 9

그러면 9가 어느 위치에 있는지 알려면?

0번째 위치부터 하나하나 다 확인해야 한다.

시간복잡도는 O(N)

리스트에 어떤 성질이 있다면 더 효율적으로 할 수 있을까?

Binary Search

- 이분탐색
- 주의 : 정렬된 리스트에서만 사용 가능

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
2	6	8	13	15	19	25	26	29	33	35	39	40	46	49

위 정렬된 리스트에서 33을 찾고자 한다면

처음 s = 0, e = 14로 저장한다.

mid = (s+e)//2 로 계산 -> 7

리스트의 mid 위치에 있는 값 26과 찾고자 하는 값 33을 비교

찾고자 하는 값이 더 크기 때문에 s를 mid+1로 변경 -> 8

다시 mid = (s+e)//2 = 11

리스트의 mid 위치에 있는 값 39와 찾고자 하는 값 33을 비교

찾고자 하는 값이 더 작기 때문에 e를 mid-1로 변경 -> 10

다시 mid = (s+e)//2 = 9

리스트의 mid 위치에 있는 값 33과 찾고자 하는 값 33을 비교

찾고자 하는 값과 같다면 mid 반환

Binary Search

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
2	6	8	13	15	19	25	26	29	33	35	39	40	46	49

33 > 26

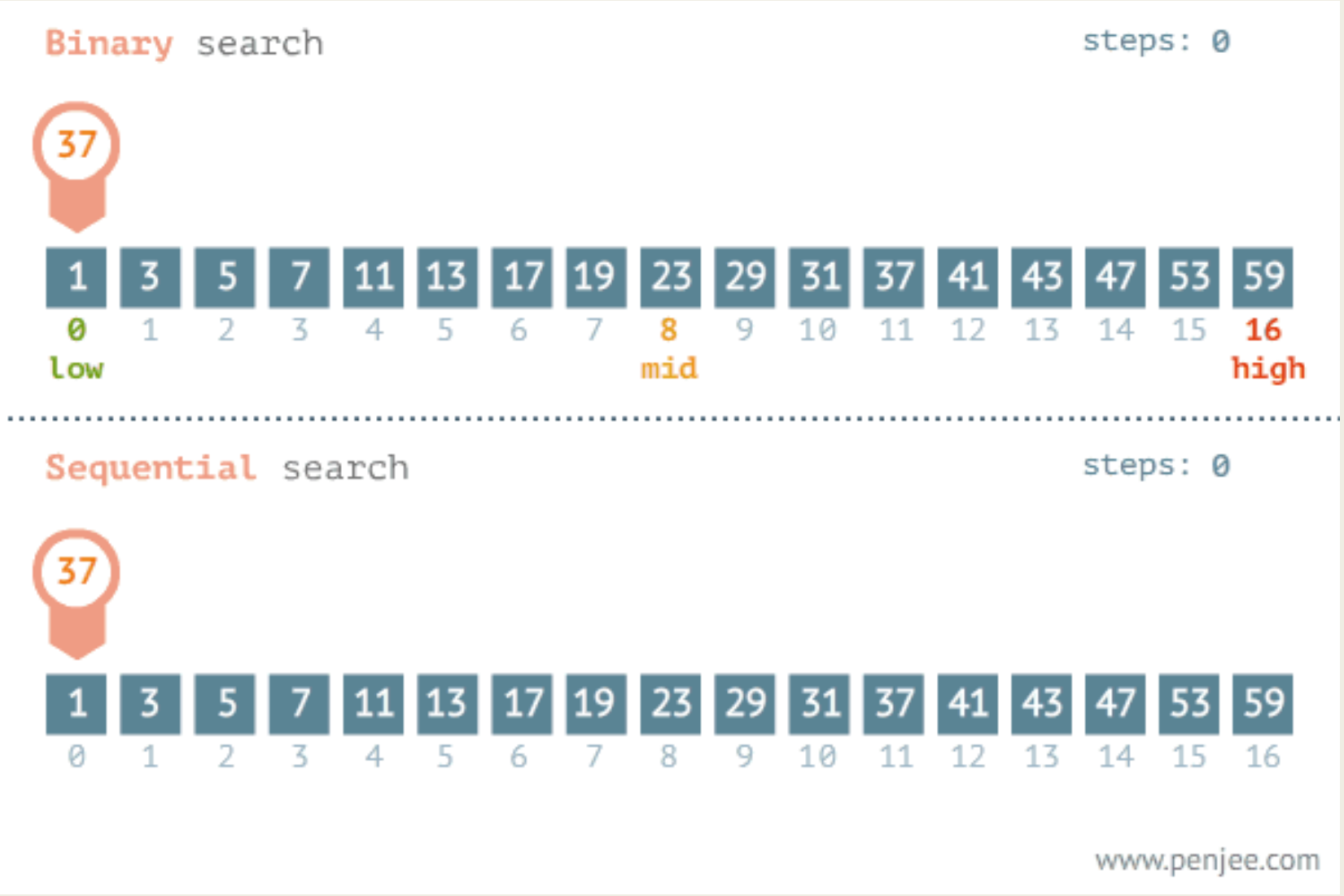
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
2	6	8	13	15	19	25	26	29	33	35	39	40	46	49

33 < 39

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
2	6	8	13	15	19	25	26	29	33	35	39	40	46	49

33 = 33

Binary Search



Binary Search

- 시간복잡도 O(logN)

- 증명 -

최악의 경우 남은 데이터가 하나가 될 때 끝난다.

첫번째 탐색 후 남은 데이터는 $N \times \frac{1}{2}$

두번째 탐색 후 남은 데이터는 $N \times \frac{1}{2} \times \frac{1}{2}$

세번째 탐색 후 남은 데이터는 $N \times \frac{1}{2} \times \frac{1}{2} \times \frac{1}{2}$

...

k번째 탐색 후 남은 데이터는 $N \times \frac{1}{2^k}$

최악의 경우를 생각해 본다면 $N \times \frac{1}{2^k} = 1$ -> $N = 2^k$ -> $k = \log_2 N$

따라서 총 $\log_2 N$ 번 탐색

Binary Search

```python
def binary_search(n_list,target):
    s=0
    e=len(n_list)-1
    while s<=e:
        mid=(s+e)//2
        if target==n_list[mid]:
            return mid
        elif target>n_list[mid]:
            s=mid+1
        else:
            e=mid-1
    return None
```

Binary Search

- 그러면 정렬이 되어 있지 않은 경우에는 어떻게 해야 하나?

➔ Sequential Search

- 정렬이 되어 있지 않고 찾고자 하는 값들이 N개가 있다면?

➔ 정렬 후 Binary Search를 이용

- 리스트의 index() 함수는 무슨 탐색일까?

➔ Sequential Search

Parametric Search

- 매개변수 탐색

기본적인 아이디어는 Binary Search와 비슷하지만

다른 점은 리스트의 원소가 중복이 될 수도 있다는 점

Ex)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
2	6	8	13	15	19	25	25	25	25	35	39	40	46	49

이 리스트에서 탐색 가능

찾고자 하는 값의 처음 위치를 반환 또는 마지막 위치를 반환 (둘 중 택1)

예를 들면 25를 찾고자 하고 처음 위치를 반환한다면 6을 반환

마지막 위치를 반환한다면 9를 반환

Parametric Search

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
2	6	8	13	15	19	25	25	25	25	35	39	40	46	49

25 <= 25

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
2	6	8	13	15	19	25	25	25	25	35	39	40	46	49

25 > 13

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
2	6	8	13	15	19	25	25	25	25	35	39	40	46	49

25 > 19

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
2	6	8	13	15	19	25	25	25	25	35	39	40	46	49

25 <= 25

Parametric Search

```python
def parametric_search(n_list,target):          # 중복된 target중 최대값
    s=0
    e=len(n_list)-1
    while s<=e:
        mid=(s+e)//2
        if target>=n_list[mid]:
            s=mid+1
        else:
            e=mid-1
    return e
```

```python
def parametric_search(n_list,target):          # 중복된 target중 최소값
    s=0
    e=len(n_list)-1
    while s<=e:
        mid=(s+e)//2
        if target>n_list[mid]:
            s=mid+1
        else:
            e=mid-1
    return s
```


실습!!