이진탐색

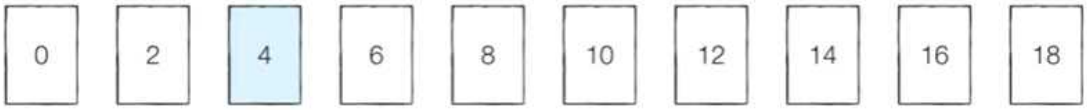
순차 탐색: 리스트 안에 있는 특정한 **데이터를 찾기 위해 앞에서부터 데이터를 하나씩 확인**하는 방법

이진 탐색: 정렬되어 있는 리스트에서 **탐색 범위를 절반씩 좁혀가며 데이터를 탐색**하는 방법

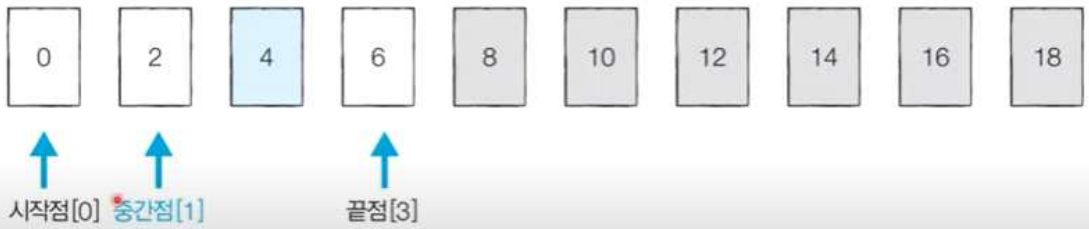
- 이진 탐색은 시작점, 끝점, 중간점을 이용하여 탐색 범위를 설정합니다.

이진 탐색 동작 예시

- 이미 정렬된 10개의 데이터 중에서 값이 4인 원소를 찾는 예시를 살펴봅시다.



- **[Step 2]** 시작점: 0, 끝점: 3, 중간점: 1 (소수점 이하 제거)



이진 탐색의 시간 복잡도

- 단계마다 탐색 범위를 2로 나누는 것과 동일하므로 **연산 횟수는** $\log_2 N$**에 비례**합니다.
- 예를 들어 초기 데이터 개수가 32개일 때, 이상적으로 1단계를 거치면 16개가량의 데이터만 남습니다.
  - 2단계를 거치면 8개가량의 데이터만 남습니다.
  - 3단계를 거치면 4개가량의 데이터만 남습니다.
- 다시 말해 이진 탐색은 탐색 범위를 절반씩 줄이며, 시간 복잡도는 $\boldsymbol{O(\log N)}$을 보장합니다.

값이 특정 범위에 속하는 데이터 개수 구하기

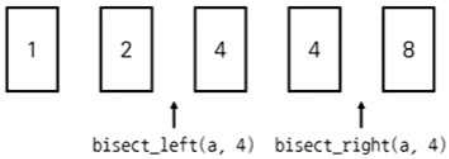
```python
from bisect import bisect_left, bisect_right

# 값이 [left_value, right_value]인 데이터의 개수를 반환하는 함수
def count_by_range(a, left_value, right_value):
    right_index = bisect_right(a, right_value)
    left_index = bisect_left(a, left_value)
    return right_index - left_index

# 배열 선언
a = [1, 2, 3, 3, 3, 3, 4, 4, 8, 9]

# 값이 4인 데이터 개수 출력
print(count_by_range(a, 4, 4))

# 값이 [-1, 3] 범위에 있는 데이터 개수 출력
print(count_by_range(a, -1, 3))
```

실행 결과

```
2
6
```

파이썬 이진 탐색 라이브러리

- bisect_left(a, x): 정렬된 순서를 유지하면서 배열 a에 x를 삽입할 가장 왼쪽 인덱스를 반
- bisect_right(a, x): 정렬된 순서를 유지하면서 배열 a에 x를 삽입할 가장 오른쪽 인덱스를




```python
from bisect import bisect_left, bisect_right

a = [1, 2, 4, 4, 8]
x = 4

print(bisect_left(a, x))
print(bisect_right(a, x))
```

실행 결과

```
2
4
```

파라메트릭 서치 (Parametric Search)

- **파라메트릭 서치**란 최적화 문제를 결정 문제('예' 혹은 '아니오')로 바꾸어 해결하는 기법입니다.
  - 예시: 특정한 조건을 만족하는 가장 알맞은 값을 빠르게 찾는 최적화 문제
- 일반적으로 코딩 테스트에서 파라메트릭 서치 문제는 **이진 탐색을 이용하여 해결**할 수 있습니다.