탐색과 분할 정복

주요 내용

- 순차 탐색
- 이진 탐색
- 분할 정복
- 분할 정복과 재귀

탐색

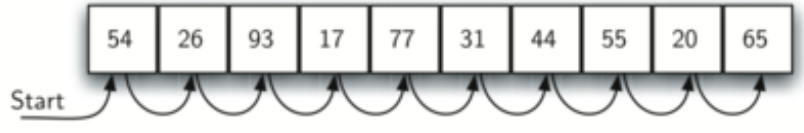
모음 객체에 특정 값이 포함되었는지 여부와 포함된 위치를 확인하는 과정

```
>>> 15 in [3, 5, 2, 4, 1]
False

>>> 3 in [3, 5, 2, 4, 1]
True
```

순차 탐색

- 인덱스 순서대로 항목 확인
- 리스트, 튜플, 넘파이 어레이 등 항목들의 순선 인덱스 지원 객체 대상



순차 탐색 알고리즘

```
def sequential_search(a_list, value):
    # 순차 탐색
    for item in a_list:
        if item == value:    # 찾은 경우 바로 True 반환
            return True

    # for 문을 다 돌아도 찾지 못한 경우 False 반환
    return False
```

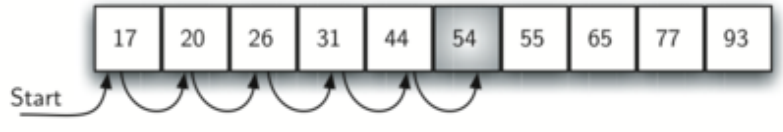
순차 탐색 시간복잡도

- 입력값의 크기: 리스트의 길이 n
- 항목이 리스트에 포함된 경우: 최선, 최악, 평균 시간복잡도가 달라짐.
- 항목이 리스트에 포함되지 않은 경우: 최선, 최악, 평균 시간복잡도가 n으로 동일함.

	최선	**최악**	**평균**
항목인 경우	1	n	$\frac{n}{2}$
항목 아닌 경우	n	n	n

정렬된 리스트 순차 탐색

- 리스트의 항목들이 오름차순으로 정렬된 경우의 순차 탐색
- 항목을 확인하다가 찾아야 하는 값보다 큰 값이 항목으로 확인되면 탐색 멈춤



```python
def ordered_sequential_search(a_list, value):
    for item in a_list:
        if item == value:
            return True
        elif item > value: # 보다 큰 값이 확인되면 바로 탐색 중단
            return False

    return False
```

	최선	최악	평균
항목인 경우	1	n	$\frac{n}{2}$
항목 아닌 경우	1	n	$\frac{n}{2}$

퀴즈

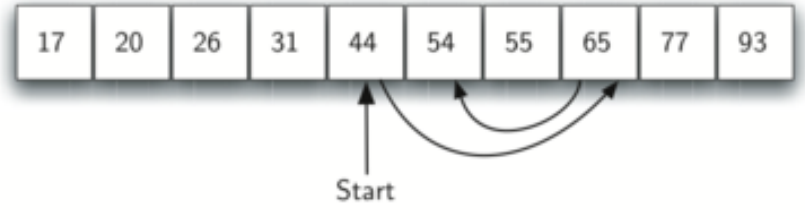
리스트 `[15, 18, 2, 19, 18, 0, 8, 14, 19, 14]`에 18이 포함되었는지 여부를 판단하는 데에 필요한 항목 비교 횟수는 얼마인가?

퀴즈

리스트 `[3, 5, 6, 8, 11, 12, 14, 15, 17, 18]`에 13이 포함되었는지 여부를 판단하는 데에 필요한 항목 비교 횟수는 얼마인가?

이진 탐색

- 정렬된 리스트 대상
- 순차 탐색보다 훨씬 빠름



이진 탐색 알고리즘

```
def binary_search(a_list, value):
    first = 0
    last = len(a_list) - 1

    while first <= last:
        midpoint = (first + last) // 2

        if a_list[midpoint] == value:
            return True

        elif value < a_list[midpoint]:
            last = midpoint - 1
        else:
            first = midpoint + 1

    return False
```

이진 탐색 시간복잡도

- 최악의 경우 시간복잡도

$$O(\log_2 n)$$

비교 횟수	탐색구간 크기
1	$\frac{n}{2}$
2	$\frac{n}{4}$
3	$\frac{n}{8}$
$\vdots$	$\vdots$
k	$\frac{n}{2^k}$

이진 탐색 알고리즘의 한계

- 리스트 정렬 필요
- 정렬 알고리즘의 최악 시간 복잡도: $O(n^2)$
- 정렬을 하고 이진 탐색을 사용할지, 아니면 그냥 순차탐색을 사용할지 판단 필요
- 한번 정렬한 다음 활용을 반복한다면 이진 탐색 선택

분할 정복

- 큰 입력사례의 해법을 보다 작은 크기의 입력사례에서 찾는 기법
- 예제: 이진 탐색

분할 정복과 재귀

- 분할 정복 기법으로 해결되는 문제 재귀 함수로 쉽게 구현 가능
- 아래 함수: 이진 탐색 재귀 함수

```
def binary_search_rec(a_list, value):
    # 종료 조건
    if len(a_list) == 0:
        return False
    # 리스트의 길이가 0보다 클 때
    else:
        midpoint = len(a_list) // 2
        if a_list[midpoint] == value:
            return True

        # 재귀호출: 탐색구간(입력값의 크기) 절반으로 줄이기
        elif value < a_list[midpoint]:
            return binary_search_rec(a_list[:midpoint], value)
        else:
            return binary_search_rec(a_list[midpoint + 1 :], value)
```

재귀 이진 탐색 알고리즘 시간 복잡도

- $O(\log n)$으로 보이지만 이는 정확하지 않음.
- 재귀호출될 때마다 새로운 리스트를 슬라이싱으로 생성하기 때문
- 슬라이싱 시간 복잡도: 슬라이싱 구간의 크기 k에 선형적으로 비례하는 $O(k)$

구간 지정 재귀 이진 탐색 알고리즘

```
def interval_binary_search_rec(a_list, value, first, last):
    if len(a_list) == 0 or last < first:
        return False
    else:
        midpoint = (first + last) // 2
        if a_list[midpoint] == value:
            return True

        # 재귀호출: 탐색구간 조정
        elif value < a_list[midpoint]:
            last = midpoint - 1
            return interval_binary_search_rec(a_list, value, first, last)
        else:
            first = midpoint + 1
            return interval_binary_search_rec(a_list, value, first, last)
```

퀴즈

리스트 `[3, 5, 6, 8, 11, 12, 14, 15, 17, 18]`가 주어졌을 때 재귀 이진 탐색 알고리즘을 이용하여 8을 탐색할 때 비교되는 값들은 무엇인가?

퀴즈

리스트 `[3, 5, 6, 8, 11, 12, 14, 15, 17, 18]`가 주어졌을 때 재귀 이진 탐색 알고리즘을 이용하여 16을 탐색할 때 비교되는 값들은 무엇인가?