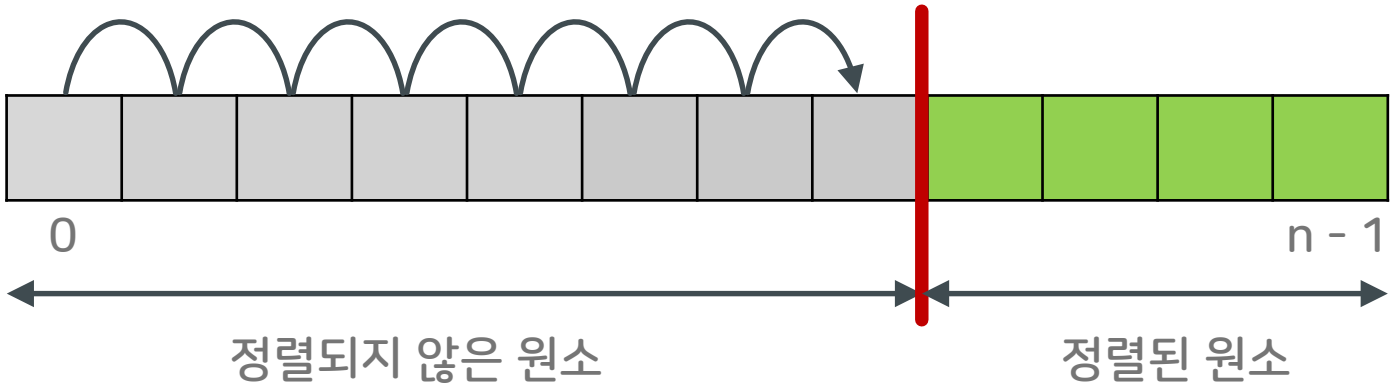
Lab10. Sort, Search and Recursion

CSED101 LAB

Sort

Sort

- 자료를 특정 기준에 따라 순서대로 나열하는 것
- 버블 정렬, 선택 정렬, 삽입 정렬 등



Bubble Sort

- 인접한 두 원소를 비교하여 순서에 맞지 않으면 서로 교환
- 이 과정을 반복하면서, 가장 큰(또는 작은) 원소가 리스트의 끝으로 이동

실습 1

- 1차원 리스트에 저장되어 있는 정수들을 오름차순으로 정렬하는 함수 bubble_sort() 를 아래 실행 예시를 참고하여 완성하시오.
  - 버블 정렬 알고리즘을 이용하여 작성할 것
  - 아래 실행 예시와 같이 각 줄에 각 스텝이 출력되도록 작성할 것
  - sorted(), list.sort() 사용 x

```
def bubble_sort(list):
    pass
```

```
# 실행 예시
>>> L = [14, 5, 20, 13, 9, -1, -22]
>>> bubble_sort(L)
[step 1]   5   14   13   9   -1  -22   20
[step 2]   5   13   9   -1  -22   14   20
[step 3]   5   9   -1  -22   13   14   20
[step 4]   5   -1  -22   9   13   14   20
[step 5]  -1  -22   5   9   13   14   20
[step 6] -22   -1   5   9   13   14   20
```

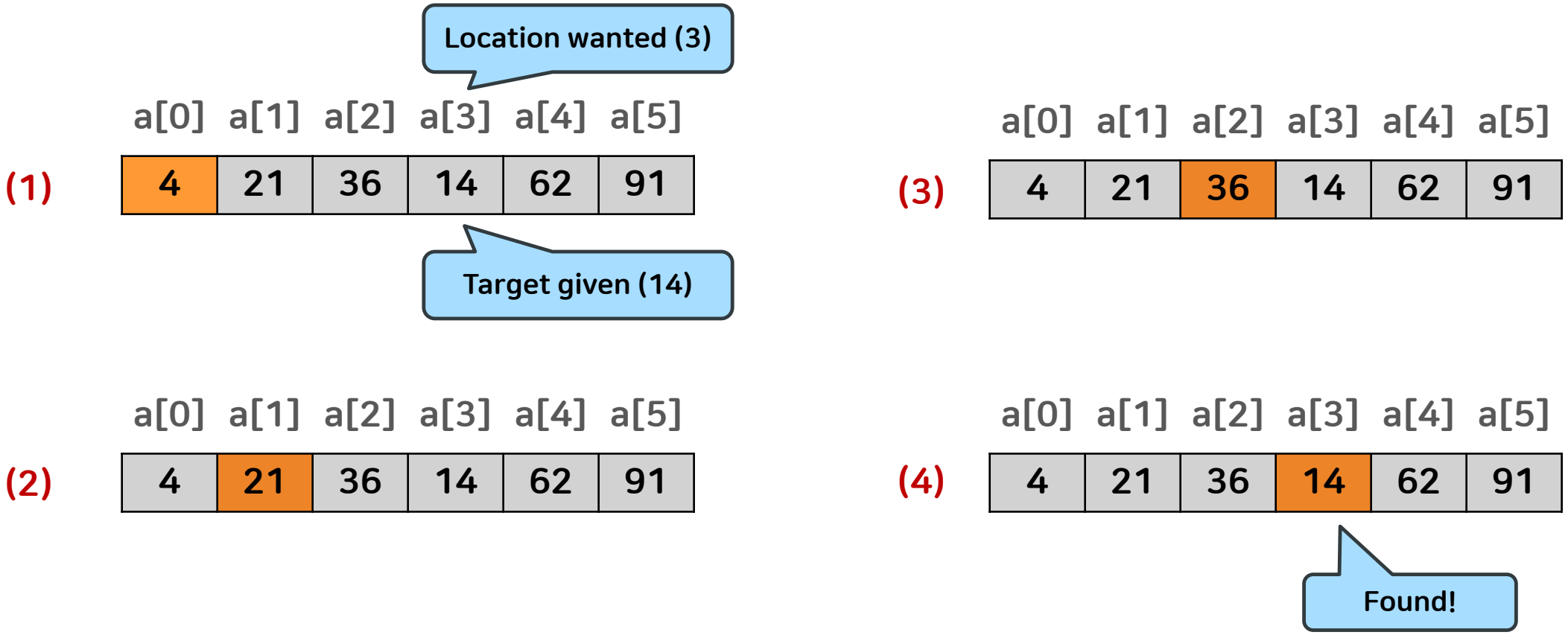
Search

탐색(Search)

- 여러 개의 자료 중에서 원하는 자료를 찾는 작업
- 순차탐색, 이진탐색

순차 탐색(Sequential Search)

- 정렬되지 않은 배열의 원소들을 처음부터 마지막까지 하나씩 검사하여 원하는 값을 찾아가는 방법



실습 2

- 순차 탐색을 수행하는 함수 seq_search(list, target)를 아래 예시를 참고하여 완성하시오.
  - 찾고자 하는 값(target)이 리스트(list) 내에 존재하면 해당 인덱스를 반환하고, 없으면 -1을 반환할 것
  - in 키워드 사용 x
  - list.index() 사용 x

```
def seq_search(list, target):
    pass
```

```
# 실행 예시
L1 = [76, 53, 80, 25, 54, 48, 79, 83, 9, 15]
target1 = 83
target2 = 200
print(seq_search(L1, target1))   # 결과: 7
print(seq_search(L1, target2))   # 결과: -1

L2 = [4, 6, 9, 17, 3]
target3 = 4
target4 = 21
print(seq_search(L2, target3))   # 결과: 0
print(seq_search(L2, target4))   # 결과: -1
```

Recursive Function

재귀 함수

- 자기 자신을 호출하는 함수
- 예시) Count down

10, 9, 8, 7, 6, 5, 4, 3, 2, 1, Done!

```
# 반복문 사용하여 구현
def countdown_for(n):
    pass
```

```
# 재귀 함수 사용하여 구현
def countdown_rec(n):
    pass
```

```
>>> countdown_for(5)
5, 4, 3, 2, 1, Done!
```

```
>>> countdown_rec(10)
10, 9, 8, 7, 6, 5, 4, 3, 2, 1, Done!
```

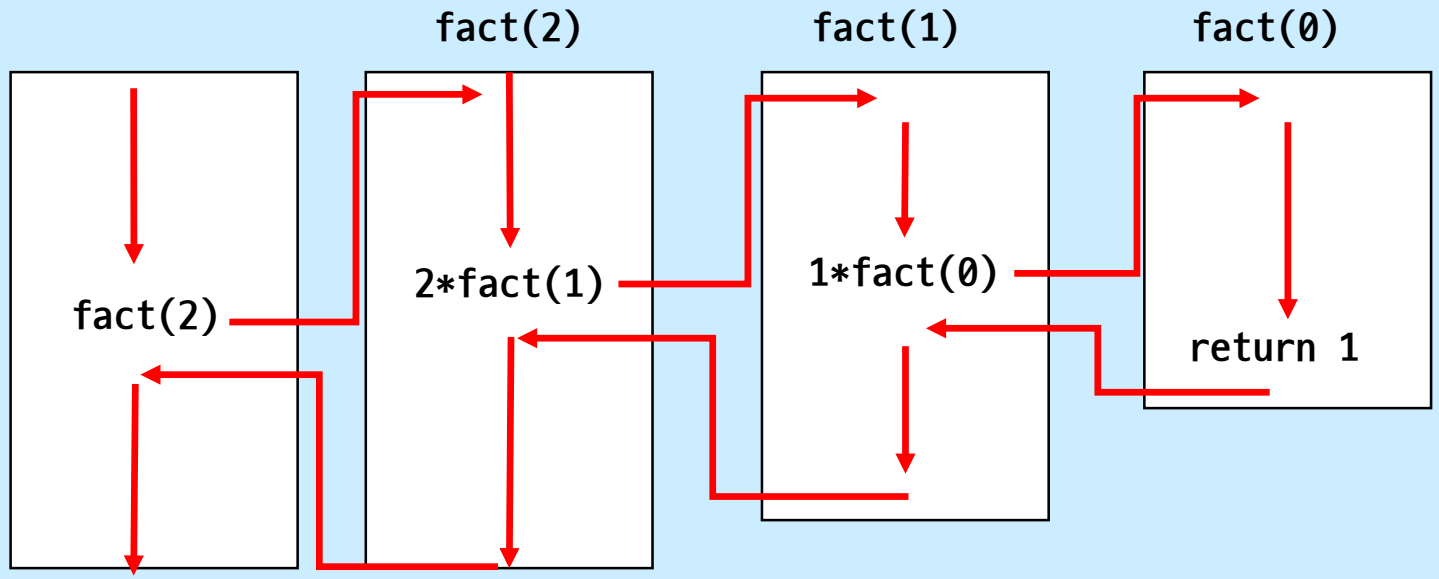
재귀 함수

- 팩토리얼(factorial)

$$n! = \begin{cases} 1 & if\ n = 0 \\ n \cdot (n-1)! & if\ n > 0 \end{cases}$$

```
def fact(n):
    if n == 0:
        return 1
    else:
        return n * fact(n-1)
```

※ 실행 흐름



실습 3

- 정수 a부터 b까지의 합을 구하여 반환하는 함수를 작성하시오.
  - (가정) a ≥ 0, a ≤ b
  - 반복문을 사용하는 방법(sigma 함수)과 재귀적 방법(sigma_rec 함수)을 사용하는 2가지 버전으로 구현할 것!

```
# 반복문 사용하여 구현
def sigma(a, b):
    pass

# 재귀적 방법 사용하여 구현
# 아래 함수 구현 시, sigma()함수 호출할 수 없음
def sigma_rec(a, b):
    pass
```

```
# 실행 예시
print( sigma(1, 10) )      # 55
print( sigma_rec(1, 10) )  # 55
```