정렬 알고리즘

유튜브 주소 : https://youtu.be/F8n9s9nGBOk

정렬 알고리즘 개요

기본 정렬 개념 및 구현

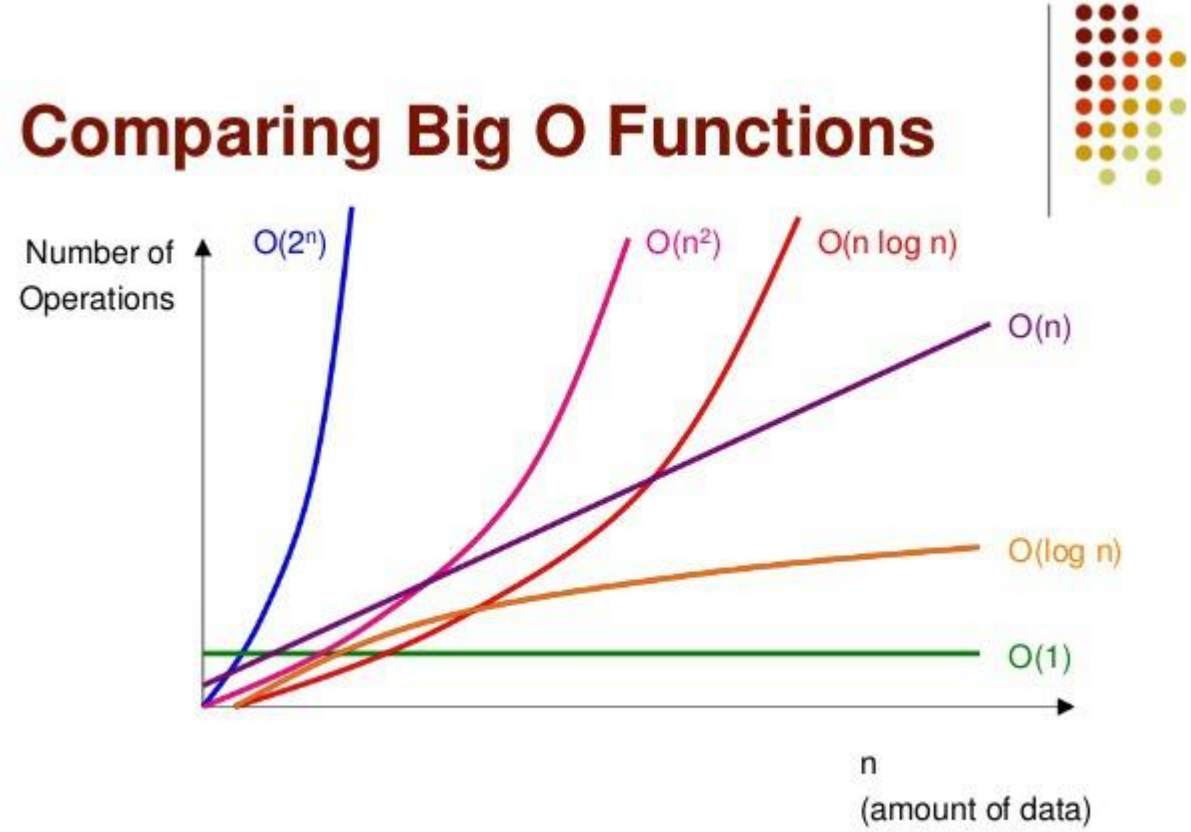
고급 정렬 개념 및 구현

정렬 알고리즘 개요

- 정렬 정의
  - 물건을 크기순으로 나열하는 것
- 정렬은 자료 탐색에 있어 필수적
- 정렬 알고리즘 조건
  - 출력은 비 내림차순
    (각 원소가 전 순서 원소에 비해 이전의 원소보다 작지 않은 순서)
  - 출력은 입력을 재배열하여 만든 순열

정렬 알고리즘 개요

- 시간 복잡도
  - 문제 해결 시 걸리는 시간과 입력의 함수 관계
- 시간 복잡도 표기법
  - 오메가 표기법
  - 세타 표기법
  - 빅 오 표기법



정렬 알고리즘 개요

- 빅 오 표기법
  - 상수항 무시
  - 영향력 없는 항 무시

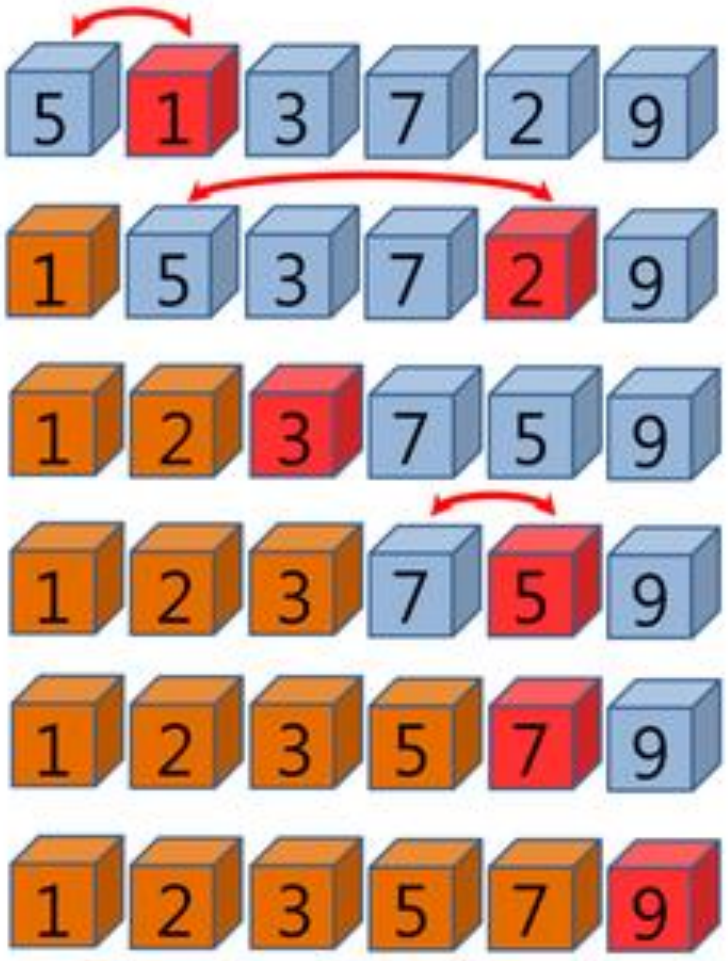
$O(2N)$ -> $O(N)$

$O(N^2+2N+1)$ -> $O(N^2)$

Complexity	1	10	100
O(1)	1	1	1
O(log N)	0	2	5
O(N)	1	10	100
O(N log N)	0	20	461
O(N^2)	1	100	10000
O(2^N)	1	1024	1267650600228229401496703205376
O(N!)	1	3628800	화면에 표현할 수 없음!

기본 정렬

- 선택 정렬
- 시간 복잡도 : O(N^2)



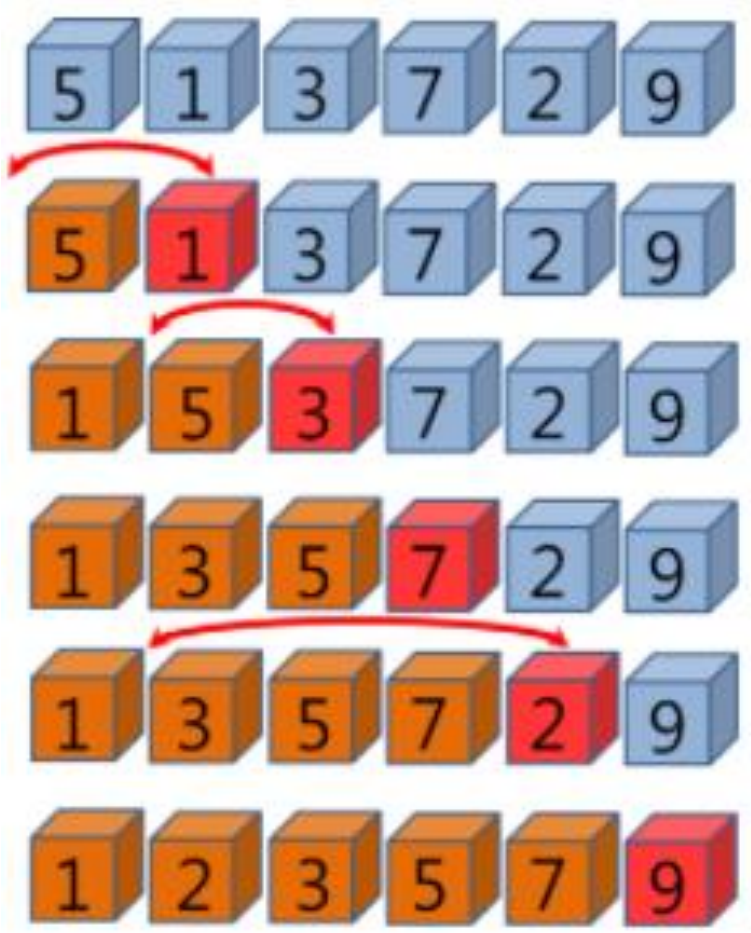
기본 정렬

- 선택 정렬 구현 코드

```
def selection_sort(data):
    for stand in range(len(data) - 1):
        lowest = stand
        for index in range(stand + 1, len(data)):
            if data[lowest] > data[index]:
                lowest = index
        data[lowest], data[stand] = data[stand], data[lowest]
    return data
```

기본 정렬

- 삽입 정렬
- 시간 복잡도 : O(N^2)



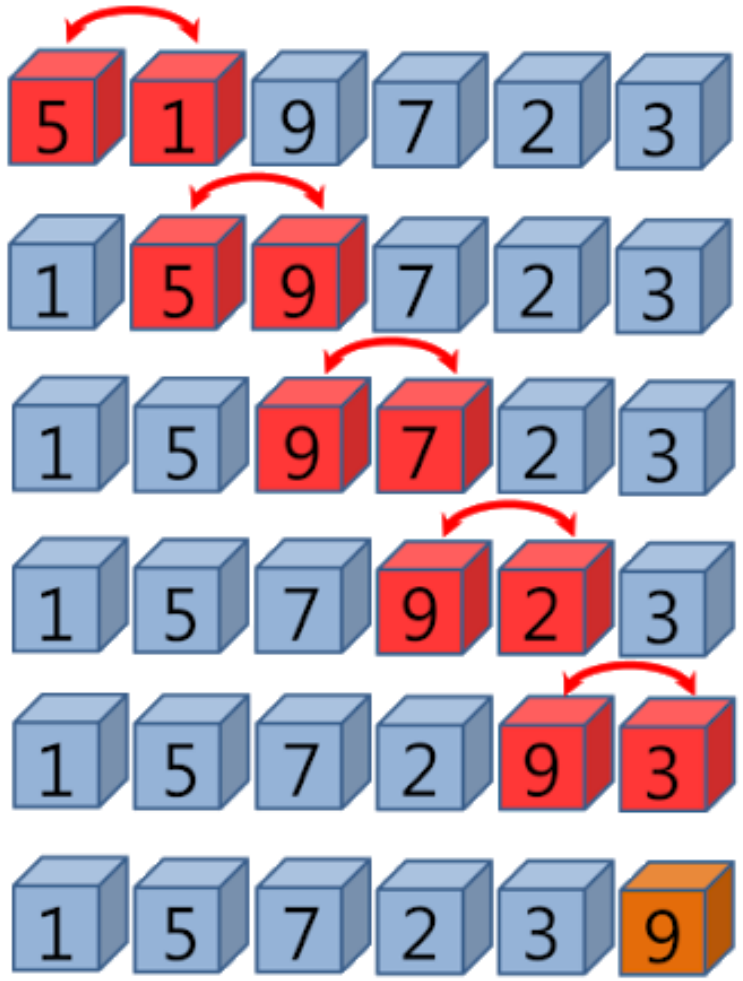
기본 정렬

- 삽입 정렬 구현 코드

```python
def insertion_sort(data):
    for index in range(len(data) - 1):
        for index2 in range(index + 1, 0, -1):
            if data[index2] < data[index2 - 1]:
                data[index2], data[index2 - 1] = data[index2 - 1], data[index2]
            else:
                break
    return data
```

기본 정렬

- 버블 정렬
- 시간 복잡도 : O(N^2)



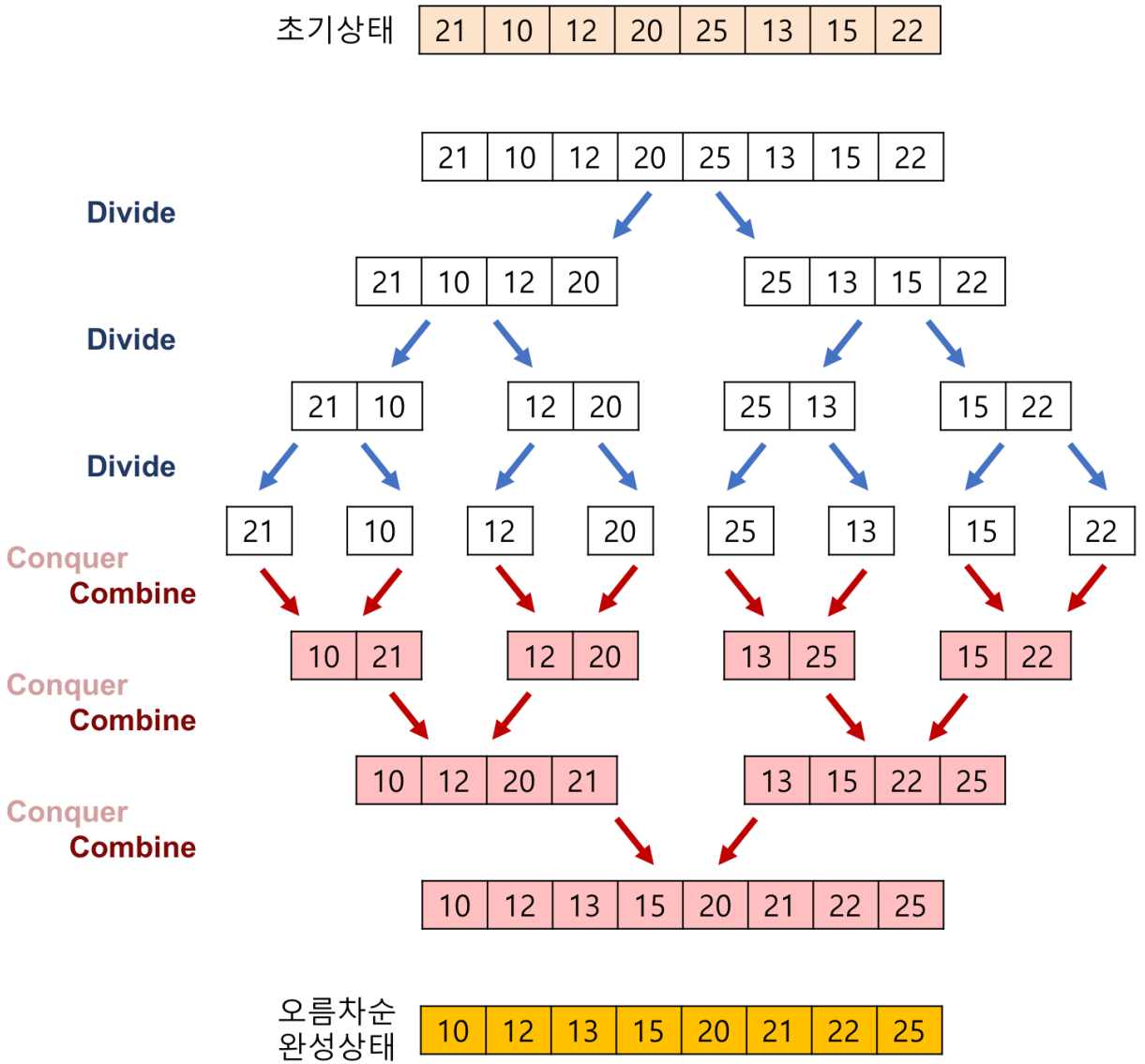
기본 정렬

- 버블 정렬 구현 코드

```python
def bubble_sort(data):
    for index in range(len(data) - 1):
        swap = False
        for index2 in range(len(data) - index - 1):
            if data[index2] > data[index2 + 1]:
                data[index2], data[index2 + 1] = data[index2 + 1], data[index2]
                swap = True
        if swap == False:
            break
    return data
```

고급 정렬

- 병합 정렬
- 시간 복잡도 : O(N logN)



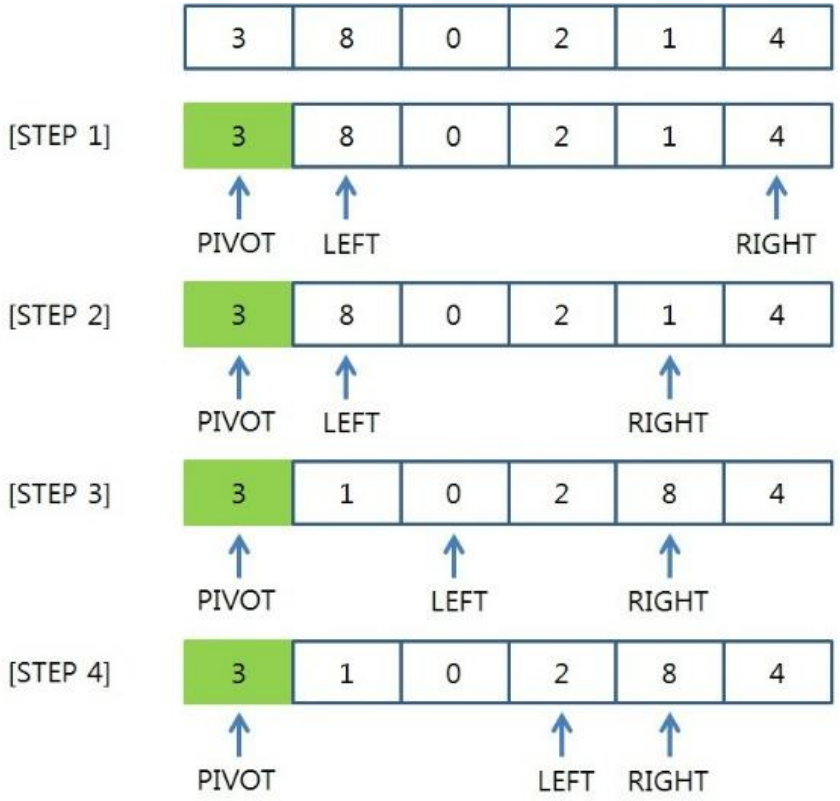
고급 정렬

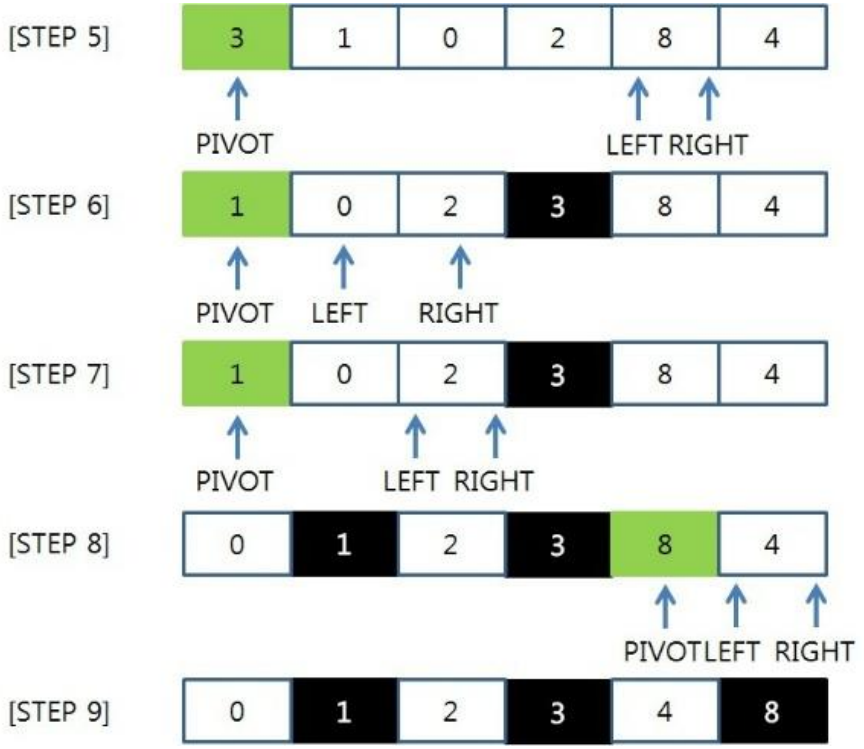
- 병합 정렬 구현 코드

```python
def merge_sort(a):
    n = len(a)
    if n <= 1:
        return a
    mid = n // 2
    g1 = merge_sort(a[:mid])
    g2 = merge_sort(a[mid:])
    result = list()
    while g1 and g2:
        if g1[0] < g2[0]:
            result.append(g1.pop(0))
        else:
            result.append(g2.pop(0))
    while g1:
        result.append(g1.pop(0))
    while g2:
        result.append(g2.pop(0))
    return result
```

고급 정렬

- 퀵 정렬
- 시간 복잡도 : O(N logN)





고급 정렬

- 퀵 정렬 구현 코드

```python
def quick_sort(data):
    if len(data) <= 1:
        return data

    pivot = data[0]

    left = [item for item in data[1:] if pivot > item]
    right = [item for item in data[1:] if pivot <= item]

    return quick_sort(left) + [pivot] + quick_sort(right)
```

Q & A