알고리즘 특강
정렬

자료구조에서 정렬을 공부하셨다면 알겠지만, 정말 중요한 내용입니다.
이번 시간엔 언어에서의 정렬을 주로 배우며, 활용은 다른 부분에서 볼 수 있습니다.

빠르게 복습하자.

Sorting

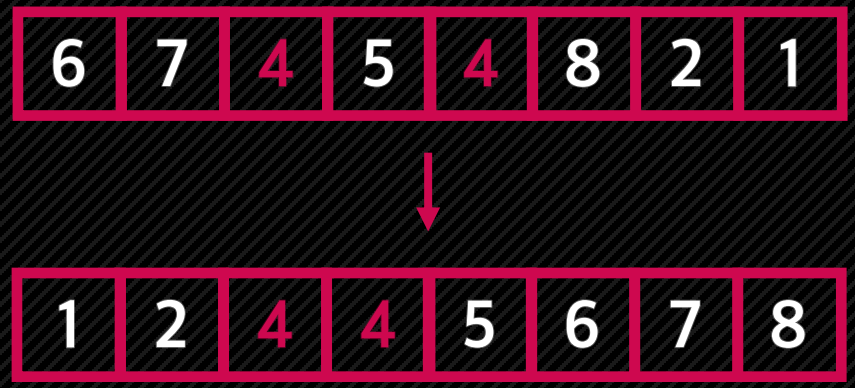
- FIFO (First In, First Out) 구조를 띄고 있는 자료구조로, 삽입과 삭제 연산이 서로 다른 한군데에서 발생함.
- 느린 알고리즘의 경우 시간복잡도가 $O(N^2)$, 빠른 경우 $O(logN)$ 정도 된다.

Python에서의 정렬

```
_list = [4, 2, 1, 3, 1]

sorted_list = sorted(_list)
print(' '.join(map(str, sorted_list)))

_list.sort()
print(' '.join(map(str, sorted_list)))

wanna_to_eat = [
    ('Chicken', 17900, 'Puradak'),
    ('Pizza', 21000, 'Domino'),
    ('Spagetti', 12000, 'Mola')
]

wanna_to_eat = sorted(wanna_to_eat, key=lambda price: wanna_to_eat[1])

wanna_to_eat = sorted(wanna_to_eat, key=lambda price: wanna_to_eat[1], reverse = True)

_list1 = [21, 61, 4, 31, 65, 98]
_list2 = [66, 12, 34, 58, 91, 3]
_dict = dict(zip(_list1, _list2))
sorted_dict = sorted(_dict.items())

for key, item in sorted_dict:
    print("dictionary[{}] = {}".format(key, item))
```

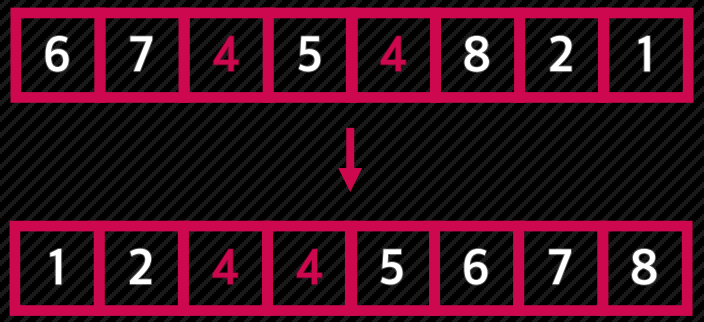
- 자료형에 대해 **오름차순**으로 정렬을 하고, 결과값을 리턴함.
- 리스트의 메소드로, 내부 정렬을 함. (리스트를 제외한 **다른 자료형에선 불가!**)
- **Key** 옵션을 통해 정렬 기준을 지정함.
- 내림차순 정렬을 하기 위해 **reverse** 사용.
- Sorted를 딕셔너리에 사용 시 **키만 리턴 됨.** 따라서, 전체 조합을 유지하려면 items 사용.

Stable Sort



- 동일한 값의 데이터가 있을 때, 순서가 바뀌지 않음이 보장되는가?

Python은…?



- 동일한 값의 데이터가 있을 때, 순서가 바뀌지 않음이 보장되는가?

- Python의 정렬 알고리즘은 Timsort로, Stable함이 보장됨!

실습 (1)

Silver 5 - 수 정렬하기 (2)

요약

- N개의 수가 주어졌을 때, 이를 오름차순으로 정렬하는 프로그램을 작성하시오.

제약조건

- N의 범위는 1 <= N <= 1,000,000 이다.
- 각각의 수의 절댓값의 범위는 1 <= N_i <= 1,000,000 이다.
- 수는 중복되지 않는다.

실습 (2)

Silver 5 - 나이순 정렬

요약

- 온라인 저지에 가입한 사람들의 나이와 이름이 가입한 순서대로 주어진다.
- 나이가 증가하는 순으로, 나이가 같으면 먼저 가입한 사람이 앞에 오는 순서로 정렬하는 프로그램을 작성하시오.

제약조건

- 총 인원의 범위는 1 <= N <= 100,000 이다.

“Any Question?”