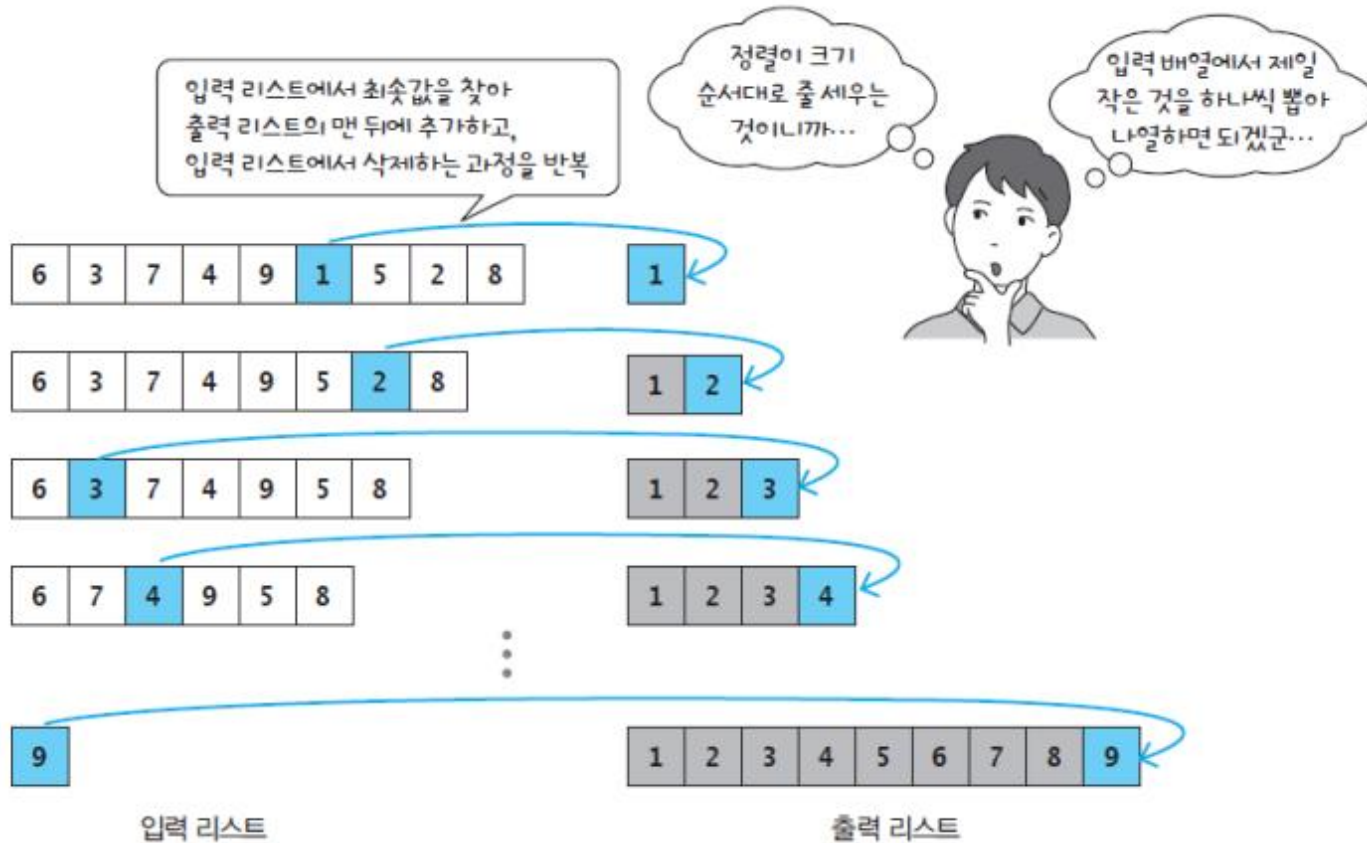


[Week 9] Practice

Sort

Problem 1

Selection sort



[Code] basic/main.py

```
def selection_sort(arr):  
    n = len(arr)  
    # i가 0부터 n-1까지 순회하도록 range 완성  
    for i in range('' block ''):  
        min_idx = i  
        for j in range(i + 1, n):  
            # 최소값 인덱스 갱신 로직 구현  
            #####  
            # block #  
            #####  
        # swap(swap) 구문 구현  
        '' block ''  
    return arr  
  
def insertion_sort(arr):  
    n = len(arr)  
    # i가 1부터 n-1까지 순회하도록 range 완성  
    for i in range('' block ''):  
        key = arr[i]  
        j = i - 1  
        # key보다 큰 요소를 뒤로 이동시키는 조건 구현  
        while j >= 0 and '' block '':  
            #####  
            # block #  
            #####  
        # key 삽입 구문 구현  
        '' block ''  
    return arr
```

Problem 1

Insertion sort

6	3	7	4	9	1	5	2	8
---	---	---	---	---	---	---	---	---

초기 상태. 6은 정렬된 리스트

6	3	7	4	9	1	5	2	8
---	---	---	---	---	---	---	---	---

3을 정렬된 리스트에 삽입

3	6	7	4	9	1	5	2	8
---	---	---	---	---	---	---	---	---

7은 이미 제자리에 있음

3	6	7	4	9	1	5	2	8
---	---	---	---	---	---	---	---	---

4를 정렬된 리스트에 삽입

3	4	6	7	9	1	5	2	8
---	---	---	---	---	---	---	---	---

9는 제자리에 있음

1	2	3	4	5	6	7	8	9
---	---	---	---	---	---	---	---	---

정렬 완료

카드처럼 숫자를
하나씩 끼워 넣어서
정렬할 수 있겠군...

6은 정렬된
리스트이고,
거기에서 3을 제
위치에 끼워 넣고,
7을 끼워 넣고...

[Code] basic/main.py

```
def selection_sort(arr):
    n = len(arr)
    # i가 0부터 n-1까지 순회하도록 range 완성
    for i in range('' block ''):
        min_idx = i
        for j in range(i + 1, n):
            # 최소값 인덱스 갱신 로직 구현
            #####
            # block #
            #####
        # 스왑(swap) 구문 구현
        '' block ''
    return arr

def insertion_sort(arr):
    n = len(arr)
    # i가 1부터 n-1까지 순회하도록 range 완성
    for i in range('' block ''):
        key = arr[i]
        j = i - 1
        # key보다 큰 요소를 뒤로 이동시키는 조건 구현
        while j >= 0 and '' block '':
            #####
            # block #
            #####
        # key 삽입 구문 구현
        '' block ''
    return arr
```

Problem 2

- Selection sort:
오름차순 선택정렬
- Insertion sort:
오름차순 삽입정렬
- Frequency based selection sort:
빈도 내림차순 우선, 값 오름차순 선택정렬
- Frequency based insertion sort:
빈도 내림차순 우선, 값 오름차순 삽입정렬

```
def frequency_calculation(arr):  
    # 빈도 계산  
    freq = {}  
    for x in arr:  
        freq[x] = freq.get(x, 0) + 1  
    return freq
```

- 빈도 계산 함수

[Code] advanced/main.py

```
def frequency_based_selection_sort(arr):  
    freq = frequency_calculation(arr)  
    n = len(arr)  
    # i가 0부터 n-1까지 순회하도록 range 완성  
    for i in range('' block ''):  
        best = i  
        # j가 i+1부터 n-1까지 순회하도록 range 완성  
        for j in range('' block ''):  
            # 빈도 내림차순 우선 · 값 오름차순 조건 기준으로 best 갱신 로직 구현  
            #####  
            #                block                #  
            #####  
        # swap 구문 구현  
        '' block ''  
    return arr  
  
def frequency_based_insertion_sort(arr):  
    freq = frequency_calculation(arr)  
    n = len(arr)  
    # i가 1부터 n-1까지 순회하도록 range 완성  
    for i in range('' block ''):  
        key = arr[i]  
        j = i - 1  
        # 빈도 내림차순 우선 · 값 오름차순 조건 기준으로 삽입 위치 찾기  
        while j >= 0 and ('' block ''):  
            #####  
            #                block                #  
            #####  
        # key 삽입 구문 구현  
        '' block ''  
    return arr
```

Problem Output

Problem 1 (basic)

```
selection_sort: [1, 2, 3, 4, 5, 6, 7, 8, 9]  
insertion_sort: [1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Problem 2 (advanced)

```
selection_sort: [1, 1, 1, 2, 2, 2, 2, 3, 3, 5, 5, 5, 6]  
insertion_sort: [1, 1, 1, 2, 2, 2, 2, 3, 3, 5, 5, 5, 6]  
frequency_based_selection_sort: [2, 2, 2, 2, 1, 1, 1, 5, 5, 5, 3, 3, 6]  
frequency_based_insertion_sort: [2, 2, 2, 2, 1, 1, 1, 5, 5, 5, 3, 3, 6]
```