정렬 알고리즘

신 제 용

ZEROBASE SCHOOL

01 정렬 알고리즘이란?

정렬 알고리즘이 무엇인지, 그리고 우리가 왜 정렬 알고리즘을 배우는지 알아봅니다.

학습 키워드 – 정렬, Sorting, Key, Comparator

정렬 (Sorting)

- 특정 값을 기준으로 데이터를 순서대로 배치하는 방법

- 정렬 알고리즘의 조건
  → 정렬 알고리즘의 출력은 **입력을 재배열하여 만든 순열**이다.
  → 정렬 알고리즘의 출력에서 **이전 원소는 다음 원소보다 작지 않다.**(오름차순)

정렬 알고리즘을 배우는 이유

- 다양한 방식으로 구현이 가능해, **알고리즘 핵심 개념 학습에 용이**하다.
  → 점근표기법 (Asymptotic notation)
  → 분할 정복 알고리즘 (Divide & Conquer)
  → 최악의 경우, 최선의 경우, 평균적인 경우

정렬 알고리즘의 주요 구분

- **제자리 알고리즘**(In-place algorithm) - 알고리즘 수행에 메모리가 $O(\log N)$ 이하로 사용되는 알고리즘

- **안정 알고리즘**(Stable algorithm) – 정렬 기준이 동일한 값이 정렬 전후에 순서가 유지되는 알고리즘

다양한 정렬 알고리즘

- 구현 난이도는 쉽지만, 속도는 느린 알고리즘
  - 버블 정렬, 삽입 정렬, 선택 정렬

- 구현 난이도는 조금 더 어렵지만, 속도는 빠른 알고리즘
  - 합병 정렬, 힙 정렬, 퀵 정렬, 트리 정렬

- 하이브리드 정렬
  - 팀소트, 블록 병합 정렬, 인트로 정렬

- 기타 정렬 알고리즘
  - 기수 정렬, 카운팅 정렬, 셸 정렬, 보고 정렬

02 기초 정렬 알고리즘

다음 챕터에서는 기초적인 정렬 알고리즘과
이를 구현하는 방법을 학습합니다.

Chapter 01
정렬 알고리즘이란?

ZEROBASE SCHOOL

02 기초 정렬 알고리즘

기초 정렬 알고리즘과 그 특성에 대해 배웁니다.

학습 키워드 – 버블 정렬, 삽입 정렬, 선택 정렬

Chapter 02
기초 정렬 알고리즘

ZEROBASE SCHOOL

In-place

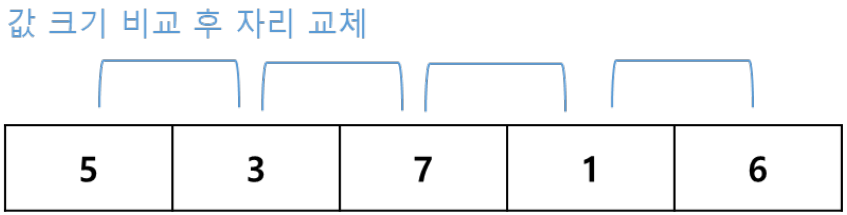
Stable

버블 정렬 (Bubble sort)

- 인접한 두 원소를 검사하여 자리를 교체하는 단순한 정렬 알고리즘

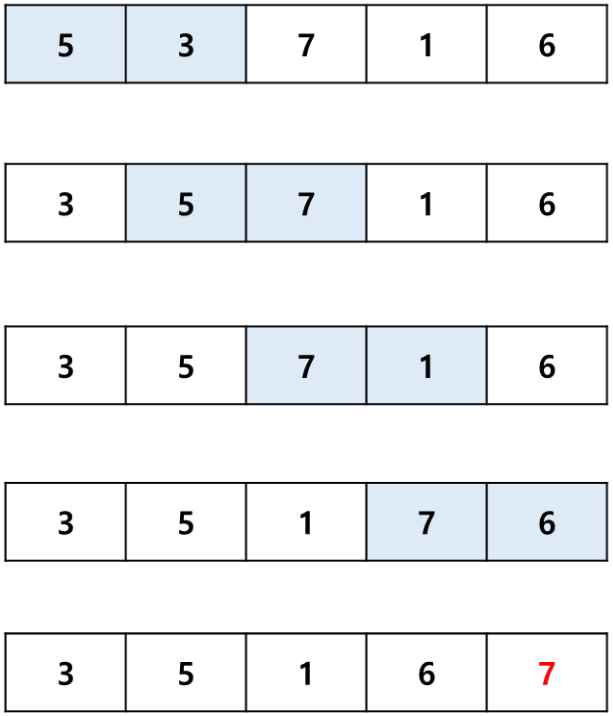
- 공간 복잡도: $O(1)$

- 시간 복잡도
  - 최악의 경우: $O(N^2)$
  - 최선의 경우: $O(N)$
  - 평균적인 경우: $O(N^2)$

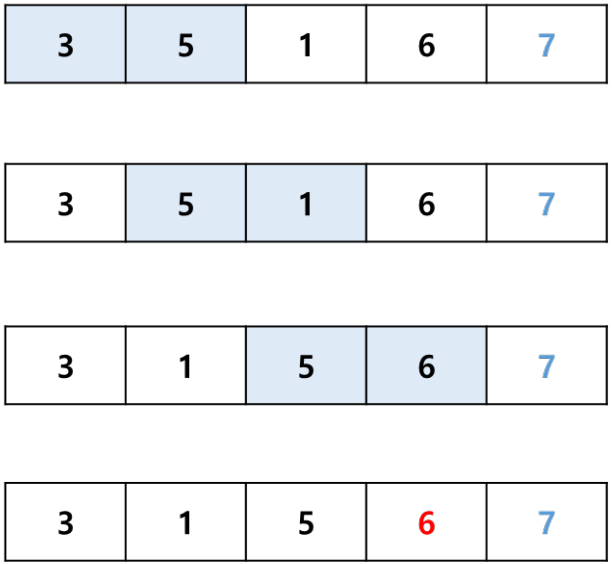


버블 정렬 과정 (1)

Step 1



Step 2

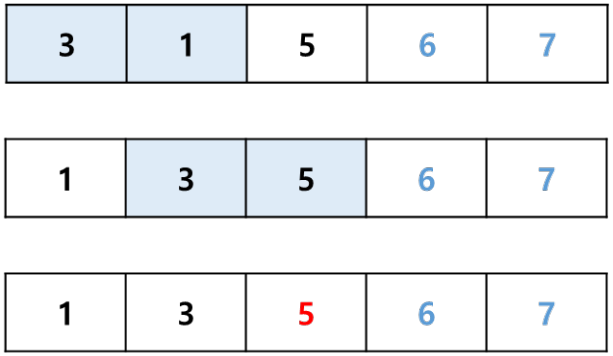


Chapter 02
기초 정렬 알고리즘

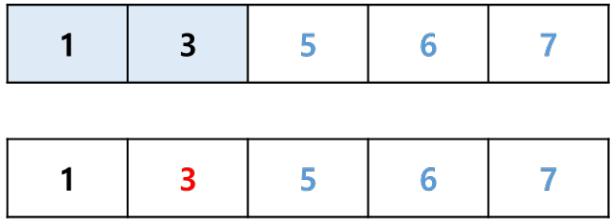
ZEROBASE SCHOOL

버블 정렬 과정 (2)

Step 3



Step 4



⇨ 1 3 5 6 7

O(n^2): (n-1) + (n-2)+...+2+1

n(n-1) / 2

버블 정렬의 구현

```
def bubble_sort(x):
    length = len(x)-1
    for i in range(length):
        swapped = False
        for j in range(length-i):
            if x[j] > x[j+1]:
                swapped = True
                x[j], x[j+1] = x[j+1], x[j]
        if swapped is False:  # 이 조건이 있어야 최선의 경우 O(n)으로 동작
            break
    return x
```

In-place

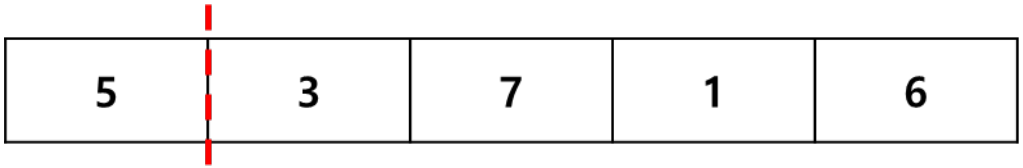
Stable

삽입 정렬 (Insertion sort)

- 앞의 데이터를 정렬 해가면서 뒤쪽의 데이터를 적절한 위치에 삽입

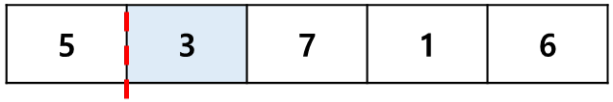
- 공간 복잡도: $O(1)$

- 시간 복잡도
  - 최악의 경우: $O(N^2)$
  - 최선의 경우: $O(N)$
  - 평균적인 경우: $O(N^2)$



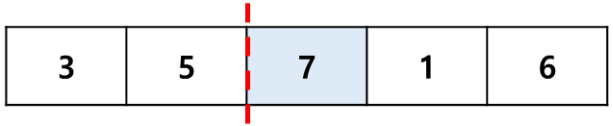
삽입 정렬 과정

Step 1

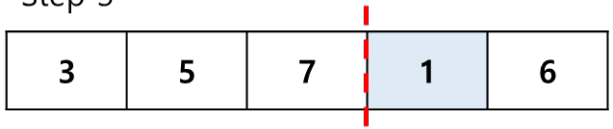


앞의 데이터와 비교 후 삽입

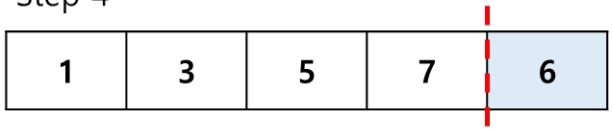
Step 2

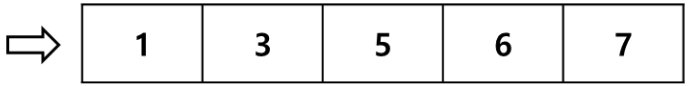


Step 3



Step 4





$O(n^2)$: (n-1) + (n-2)+...+2+1

n(n-1) / 2

Chapter 02
기초 정렬 알고리즘

ZEROBASE SCHOOL

삽입 정렬의 구현

```
def insertion_sort(x):
    for i in range(1, len(x)): # 정렬된 영역이 1씩 증가
        j = i - 1 # 0 ~ j가 정렬된 영역
        key = x[i] # x[i]를 왼쪽으로 한칸씩 옮기면서 제자리 찾기
        while x[j] > key and j >= 0:
            x[j+1] = x[j]
            j = j - 1
        x[j+1] = key
    return x
```

In-place

선택 정렬 (Selection sort)

- 최소값을 찾아서 맨 앞으로 정렬하는 방식

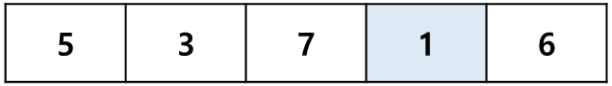
- 공간 복잡도: $O(1)$

- 시간 복잡도
  - 최악의 경우: $O(N^2)$
  - 최선의 경우: $O(N^2)$ → 매번 최소값을 찾아야 하므로 항상 일정하게 동작
  - 평균적인 경우: $O(N^2)$

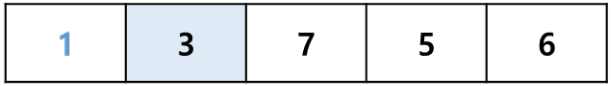
선택 정렬 과정

Step 1

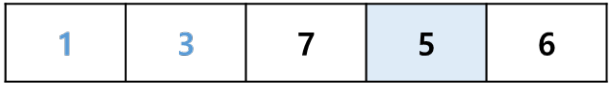
최소 값



Step 2

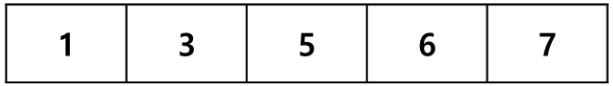


Step 3



Step 4

⇨



$O(n^2)$: (n-1) + (n-2)+...+2+1

n(n-1) / 2

Chapter 02

기초 정렬 알고리즘

ZEROBASE SCHOOL

선택 정렬의 구현

```
def selection_sort(x):
    length = len(x)
    for i in range(length-1):
        index_min = i
        for j in range(i+1, length):
            if x[index_min] > x[j]:
                index_min = j
        x[i], x[index_min] = x[index_min], x[i]
    return x
```

03 합병 정렬

다음 챕터에서는 분할 정복 방식을 사용하는 합병 정렬을 학습합니다.

Chapter 02
기초 정렬 알고리즘

03 합병 정렬

뛰어난 성능을 보이는 분할 정복 알고리즘인 합병 정렬을
알아봅니다.

학습 키워드 – 합병 정렬, Merge sort, 분할 정복, Divide & Conquer

Chapter 03
합병 정렬

ZEROBASE SCHOOL

Stable

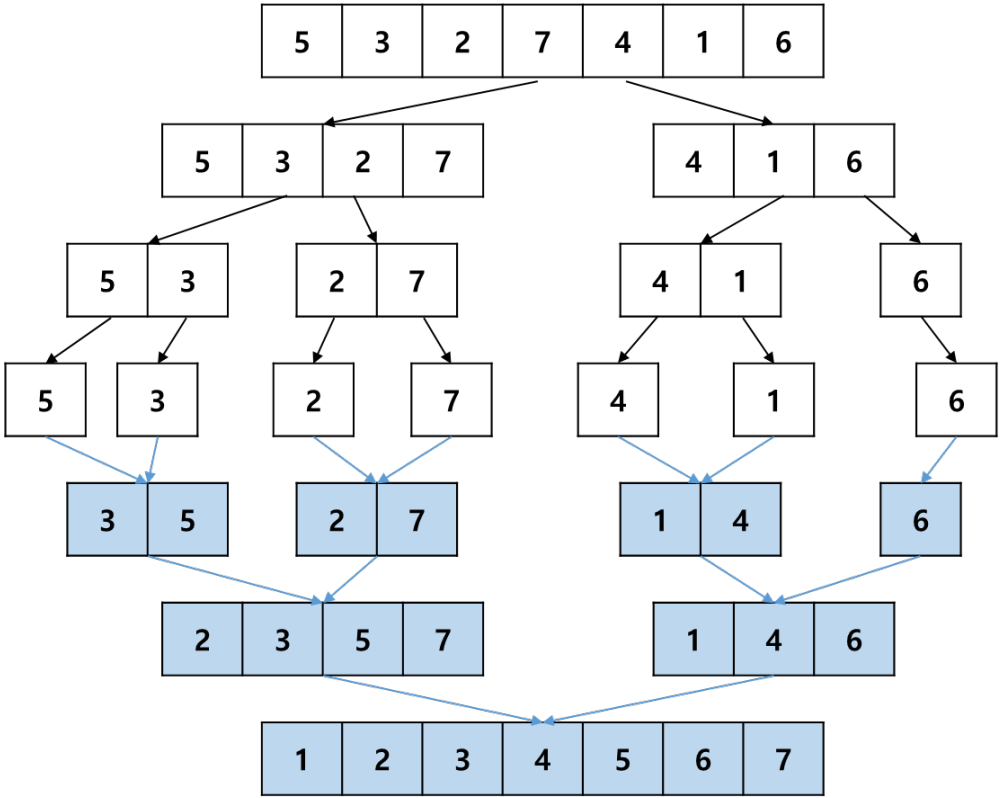
합병 정렬 (Merge sort)

- 배열의 길이가 1이 될 때 까지 이분할 하고, 인접한 배열끼리 합병하면서 정렬하는 알고리즘

- 공간 복잡도: $O(N)$

- 시간 복잡도
  - 최악의 경우: $O(N \log N)$
  - 최선의 경우:$O(N \log N)$
  - 평균적인 경우:$O(N \log N)$

합병 정렬 과정



Chapter 02

기초 정렬 알고리즘

ZEROBASE SCHOOL

합병 정렬의 구현

```python
def merge_sort(x, low=0, high=-1, arr=None):
    if arr is None:
        arr = [0] * len(x)

    if high == -1:
        high = len(x) - 1

    if low >= high:
        return

    mid = (low + high) // 2
    merge_sort(x, low, mid, arr)
    merge_sort(x, mid + 1, high, arr)
```

```python
    i, j = low, mid + 1
    for k in range(low, high + 1):
        if j > high:
            arr[k] = x[i]
            i += 1
        elif i > mid:
            arr[k] = x[j]
            j += 1
        elif x[i] <= x[j]:
            arr[k] = x[i]
            i += 1
        else:
            arr[k] = x[j]
            j += 1

    x[low:high + 1] = arr[low:high + 1]
```

04 퀵 정렬

다음 챕터에서는 빠른 것으로 유명한 퀵 정렬을 학습합니다.

Chapter 03
합병 정렬

ZEROBASE SCHOOL

04 퀵 정렬

이름부터 심상치 않은 퀵 정렬, 얼마나 빠른지 한번 배워봅시다.

학습 키워드 – 퀵 정렬, Quick sort, pivot, Top-Down

Chapter 04
퀵 정렬

ZEROBASE SCHOOL

In-place

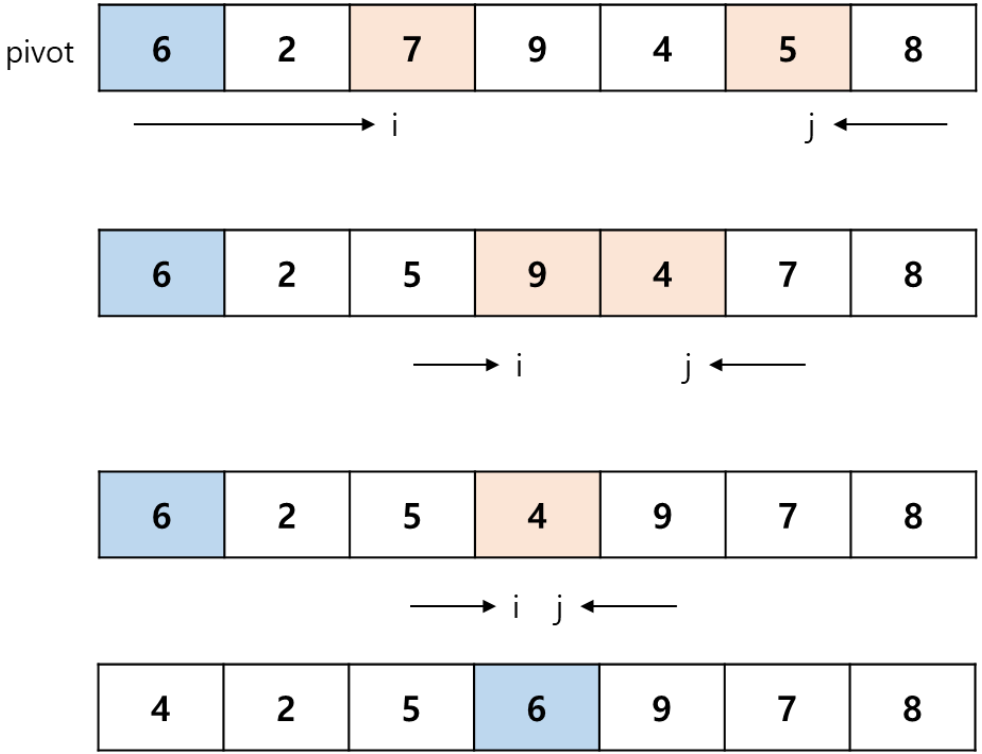
퀵 정렬 (Quick sort)

- 임의의 값(pivot)을 정해, 그 값을 기준으로 좌우로 나누어 정렬하는 알고리즘. Pivot의 선택에 의해 알고리즘 성능이 크게 달라질 수 있다.

- 공간 복잡도: $O(\log N)$

- 시간 복잡도
  - 최악의 경우: $O(N^2)$
  - 최선의 경우: $O(N \log N)$
  - 평균적인 경우: $O(N \log N)$

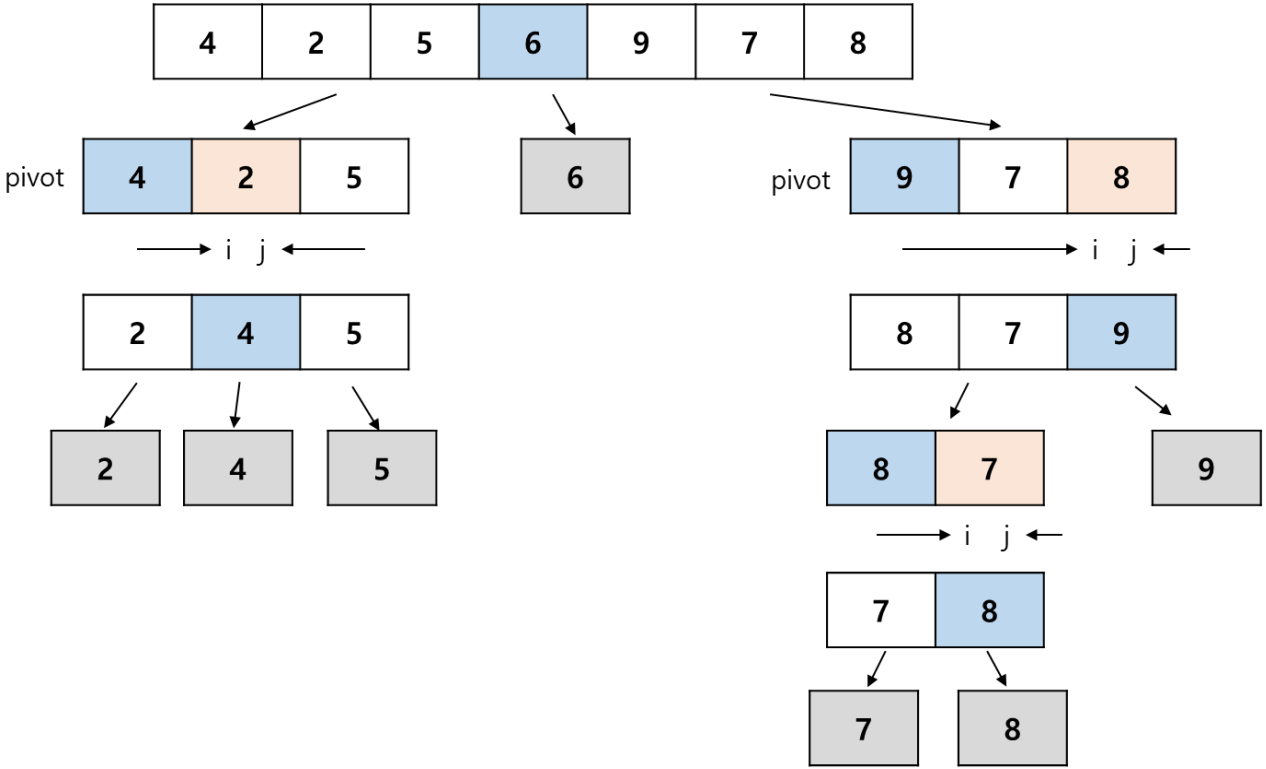
퀵 정렬 과정 (1)



Chapter 02
기초 정렬 알고리즘

ZEROBASE SCHOOL

퀵 정렬 과정 (2)



Chapter 02

기초 정렬 알고리즘

ZEROBASE SCHOOL

퀵 정렬의 구현 (Easy)

```python
def quick_sort(x):
    if len(x) < 2:
        return x

    left = []
    right = []
    pivot = x[len(x) // 2]

    for el in x:
        if el < pivot:
            left.append(el)
        else:
            right.append(el)

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

퀵 정렬의 구현 (Hard)

```
def quick_sort(arr):
    def sort(low, high):
        if high <= low:
            return

        mid = partition(low, high)
        sort(low, mid - 1)
        sort(mid, high)

    def partition(low, high):
        pivot = arr[(low + high) // 2]
        while low <= high:
            while arr[low] < pivot:
                low += 1
            while arr[high] > pivot:
                high -= 1
            if low <= high:
                arr[low], arr[high] = arr[high], arr[low]
                low, high = low + 1, high - 1
        return low

    return sort(0, len(arr) - 1)
```

Chapter 02
기초 정렬 알고리즘

ZEROBASE SCHOOL

05 팀소트

다음 챕터에서는 현대적인 정렬 알고리즘인 팀소트를 학습합니다.

Chapter 04
퀵 정렬

05 팀소트

팀소트는 다양한 구현체에서 사용하는 주요 정렬 알고리즘입니다.

학습 키워드 – 팀소트, Timsort

Chapter 05
팀소트

ZEROBASE SCHOOL

Stable

팀소트 (Timsort)

- Java SE 7, Android, GNU Octave, Chrome V8, Swift, Rust, Python 등 다양한 인터프리터/컴파일러에 구현된 정렬 알고리즘

- 개선된 버전의 **삽입 정렬**과 **합병 정렬**을 결합하여 구현된 **하이브리드 정렬 알고리즘**

- 현실의 데이터에서 매우 **성능이 뛰어나며**, **안정 알고리즘**이므로 활용성이 매우 뛰어나다.

- 공간 복잡도: $O(N)$

- 시간 복잡도
  - 최악의 경우: $O(N \log N)$
  - 최선의 경우: $O(N)$
  - 평균적인 경우: $O(N \log N)$

정렬 알고리즘 복잡도 비교

정렬 방법	시간 복잡도 O(.)			보조 메모리	안정성
	최선	평균	최악		
버블 정렬	n	n^2	n^2	1	o
삽입 정렬	n	n^2	n^2	1	o
선택 정렬	n^2	n^2	n^2	1	x
합병 정렬	$n \log n$	$n \log n$	$n \log n$	n	o
퀵 정렬	$n \log n$	$n \log n$	n^2	$\log n$	x
팀소트	n	$n \log n$	$n \log n$	n	o

06 정렬 예시 문제 풀이

다음 챕터에서는 정렬을 응용하는 문제를 해결하는 방법을 학습합니다.

Chapter 05
팀소트

ZEROBASE SCHOOL

06 정렬 예시 문제 풀이

정렬 알고리즘은 많은 알고리즘의 일부를 담당합니다. 다양한 상황에 맞게 정렬하는 방법을 예시 문제를 통해 학습합니다.

학습 키워드 – 정렬, Sort, 구현

Chapter 06
정렬 예시 문제 풀이

ZEROBASE SCHOOL

Problem1

문제 설명

당신은 학교에서 선생님으로 일하고 있다. 어느날 학생의 성적를 관리하는 프로그램을 제작해 달라는 의뢰를 받았다.

총 `N`명의 학생의 성적을 관리하고자 한다. 이 때 `i`번째 학생(`0 <= i < N`)의 학년이 `grade[i]`, 반이 `class_name[i]`, 점수가 `score[i]`에 기록되어 있다.

이 때, 학생의 점수를 아래와 같은 조건에 맞게 정렬하여 출력하시오.

- 정렬의 우선순위는 '학년이 낮을 수록 앞에', '반 숫자가 낮을 수록 앞에', '점수가 높을 수록 앞에' 순으로 정렬한다.

매개변수 형식

```
grade = [3, 2, 1, 2, 1, 3, 2]
```

```
class_name = [1, 3, 2, 3, 1, 3, 3]
```

```
score = [50, 40, 66, 80, 100, 42, 99]
```

반환값 형식

```
[100, 66, 99, 80, 40, 50, 42]
```

Chapter 06
정렬 예시 문제 풀이

ZEROBASE SCHOOL

Problem2

문제 설명

문자열 `s`가 있을 때, 이 문자열을 재배치하여 만든 문자열을 '애너그램'이라고 한다.

예를 들어, `"fine"`은 `"infe"`의 애너그램이라고 할 수 있다.

`s`가 영문 소문자로만 이루어져 있다고 할 때, 문자열 `t`가 문자열 `s`의 애너그램인지 판단하는 프로그램을 작성하시오.

매개변수 형식

`s = "imfinethankyou"` `t = "atfhinemnoyuki"`

반환값 형식

`True`

Chapter 06
정렬 예시 문제 풀이

ZEROBASE SCHOOL

Problem3

문제 설명

당신은 천재적인 두뇌를 가진 개발자끼리 겨루는 '제로 지니어스' 프로그램에 참가하게 되었다.

'제로 지니어스' 프로그램에서는 주어진 숫자를 이어붙여 가장 큰 수를 만드는 프로그램을 작성하는 미션이 주어졌다.

문제의 조건은 아래와 같다.

- `0` 또는 양의 정수가 `numbers` 배열로 주어진다.
- `numbers` 배열에 주어진 정수를 이어붙여 만들 수 있는 가장 큰 수를 출력한다.

예를 들어, 주어진 정수가 `[6, 10, 2]` 라면 `[6102, 6210, 1062, 1026, 2610, 2106]` 를 만들 수 있고, 이중 가장 큰 수는 `6210` 이다.

위 미션을 수행하여 프로그램을 작성하시오. 단, 출력 정수 값이 너무 클 것을 대비하여 문자열로 출력하시오.

매개변수 형식

```
numbers = [3, 30, 34, 5, 9]
```

반환값 형식

```
"9534330"
```

Chapter 06
정렬 예시 문제 풀이

ZEROBASE SCHOOL