

# 정렬 알고리즘의 이해

---

송기태 ([kitae040522@gmail.com](mailto:kitae040522@gmail.com))  
Soongsil Univ. (Computer Science and Engineering)

# Content

선택 정렬

버블 정렬

삽입 정렬

파이썬으로 정렬 알고리즘 구현

## 선택 정렬 알고리즘의 개념

- 가장 작은 원소를 맨 앞에 있는 원소와 교환하면서 정렬하는 알고리즘
- $A[1...N]$  배열이 있을 때, 가장 큰 원소를 찾고 배열의 마지막 자리에 있는 원소와 자리를 바꾼다.
- 크기가 줄은  $A[1...N-1]$  배열에서 가장 큰 원소를 찾고  $N-1$ 번째 자리에 있는 원소와 자리를 바꾼다.
- $A[1...N-2]$  ... 이하 생략
- 루프를 돌 때마다 탐색해야 하는 배열의 크기가 하나씩 줄어들면서 원소들이 맞는 자리를 하나씩 찾는다.

# 선택 정렬 알고리즘의 작동 과정

정렬할 배열이 주어진다

8	31	48	73	3	11
---	----	----	----	---	----

## 선택 정렬 알고리즘의 작동 과정

가장 큰 수를 찾는다 (73)

8	31	48	73	3	11
---	----	----	----	---	----

# 선택 정렬 알고리즘의 작동 과정

73을 맨 오른쪽 수(11)와 자리 바꾼다

8	31	48	11	3	73
---	----	----	----	---	----

### 선택 정렬 알고리즘의 작동 과정

맨 오른쪽 수를 제외한 나머지에서 가장 큰 수를 찾는다 (48)

8	31	48	11	3	73
---	----	----	----	---	----

## 선택 정렬 알고리즘의 작동 과정

48을 맨 오른쪽 수(3)와 자리 바꾼다

8	31	3	11	48	73
---	----	---	----	----	----



### 선택 정렬 알고리즘의 작동 과정

맨 오른쪽 두 수를 제외한 나머지에서 가장 큰 수를 찾는다 (31)

8	31	3	11	48	73
---	----	---	----	----	----

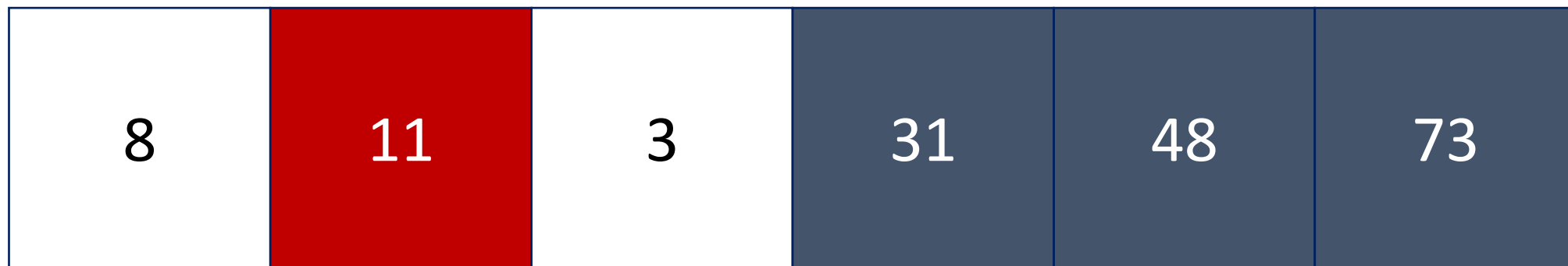
# 선택 정렬 알고리즘의 작동 과정

앞에서 했던 걸 계속 반복한다

8	11	3	31	48	73
---	----	---	----	----	----

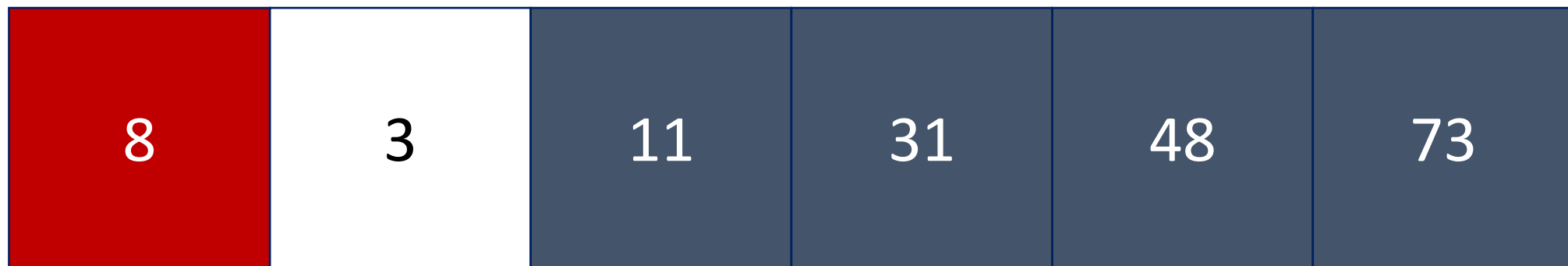
## 선택 정렬 알고리즘의 작동 과정

앞에서 했던 걸 계속 반복한다



## 선택 정렬 알고리즘의 작동 과정

앞에서 했던 걸 계속 반복한다



# 선택 정렬 알고리즘의 작동 과정

앞에서 했던 걸 계속 반복한다

3	8	11	31	48	73
---	---	----	----	----	----

# 선택 정렬 알고리즘의 작동 과정

앞에서 했던 걸 계속 반복한다

3	8	11	31	48	73
---	---	----	----	----	----

# 선택 정렬 알고리즘은 몇 번 연산을 해야할까?

- 가장 큰 값을 찾는 연산
  - 크기가 N인 배열에서 가장 큰 값을 찾으려면 N만큼 연산해야함
  - 크기가 3인 배열에서 가장 큰 값을 찾으려면 3번 연산하면 됨
- 위치를 교환하는 연산

# 선택 정렬 알고리즘은 몇 번 연산을 해야할까?

- 가장 큰 값을 찾는 연산
  - 크기가 N인 배열에서 가장 큰 값을 찾으려면 N만큼 연산해야함
  - 크기가 3인 배열에서 가장 큰 값을 찾으려면 3번 연산하면 됨
- 위치를 교환하는 연산
- 위에 연산들을 N번 반복해야 하니까 대충  $N^2$  만큼 연산하면 정렬이 됨  
(시간복잡도의 개념을 설명하지 않았기 때문에 대충 설명함.)



## 버블 정렬 알고리즘의 개념

- 서로 인접한 두 원소를 비교해서 필요에 따라 위치를 교환하여 정렬하는 알고리즘
- 선택 정렬이랑 똑같이 제일 큰 원소를 끝자리로 옮기는 작업을 반복한다.
- $A[1...N]$  배열이 있을 때,  $A[1]$ 와  $A[2]$ 를 비교해서  $A[1]$ 이 더 크면  $A[2]$ 랑 위치를 바꾼다.
- $A[2]$ 와  $A[3]$ 을 비교해서  $A[2]$ 가 작으면 위치를 바꾸지 않는다.
- $A[3]$ 과  $A[4]$ 를 비교해서 ... 이하 생략

## 버블 정렬 알고리즘의 작동 과정

정렬할 배열이 주어진다

8	31	48	73	3	11
---	----	----	----	---	----

## 버블 정렬 알고리즘의 작동 과정

왼쪽부터 시작해 이웃한 쌍들을 비교한다

8	31	48	73	3	11
---	----	----	----	---	----

## 버블 정렬 알고리즘의 작동 과정

왼쪽부터 시작해 이웃한 쌍들을 비교한다

8	31	48	73	3	11
---	----	----	----	---	----

## 버블 정렬 알고리즘의 작동 과정

왼쪽부터 시작해 이웃한 쌍들을 비교한다

8	31	48	73	3	11
---	----	----	----	---	----

## 버블 정렬 알고리즘의 작동 과정

비교했을 때, 왼쪽 값이 오른쪽보다 크다면 서로 자리를 바꾼다

8	31	48	73	3	11
---	----	----	----	---	----

## 버블 정렬 알고리즘의 작동 과정

비교했을 때, 왼쪽 값이 오른쪽보다 크다면 서로 자리를 바꾼다

8	31	48	3	73	11
---	----	----	---	----	----

## 버블 정렬 알고리즘의 작동 과정

왼쪽부터 시작해 이웃한 쌍들을 비교한다

8	31	48	3	73	11
---	----	----	---	----	----



## 버블 정렬 알고리즘의 작동 과정

비교했을 때, 왼쪽 값이 오른쪽보다 크다면 서로 자리를 바꾼다

8	31	48	3	11	73
---	----	----	---	----	----

## 버블 정렬 알고리즘의 작동 과정

맨 오른쪽 수(73)를 대상에서 제외한다

8	31	48	3	11	73
---	----	----	---	----	----

## 버블 정렬 알고리즘의 작동 과정

반복

8	31	48	3	11	73
---	----	----	---	----	----

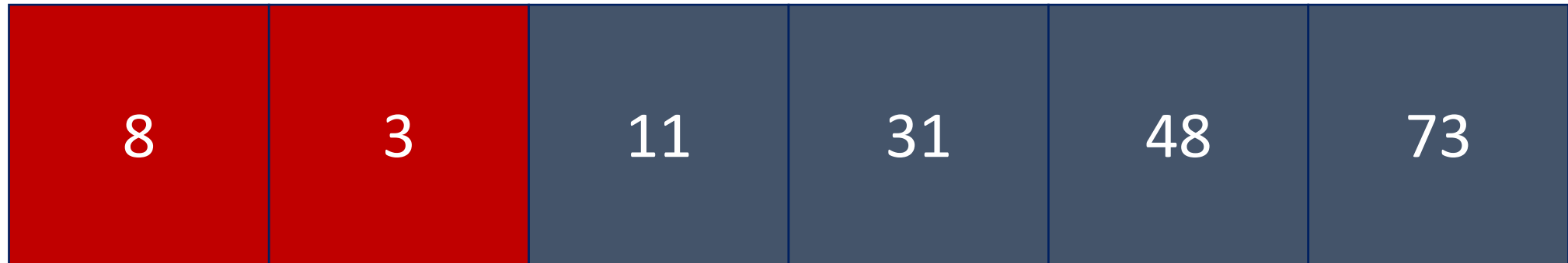
## 버블 정렬 알고리즘의 작동 과정

맨 오른쪽 수(73)를 대상에서 제외한다

8	3	11	31	48	73
---	---	----	----	----	----

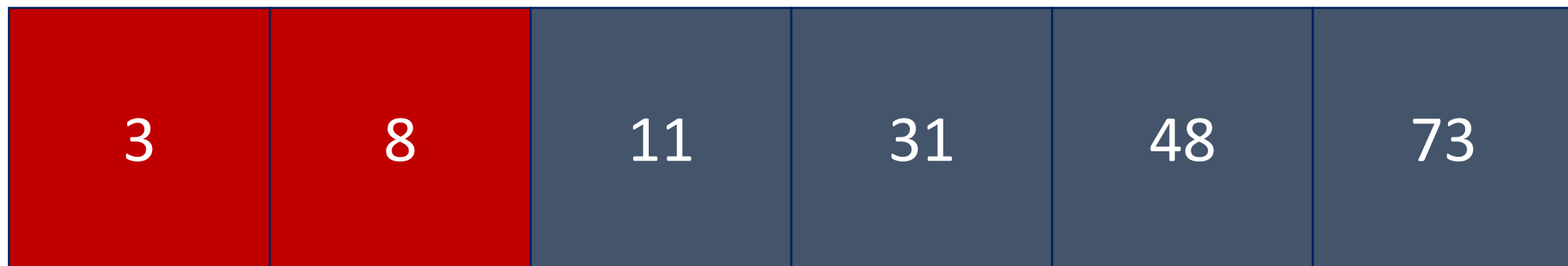
## 버블 정렬 알고리즘의 작동 과정

두 개짜리 배열의 처리를 끝으로 정렬이 완료된다



## 버블 정렬 알고리즘의 작동 과정

두 개짜리 배열의 처리를 끝으로 정렬이 완료된다



## 버블 정렬 알고리즘의 작동 과정

두 개짜리 배열의 처리를 끝으로 정렬이 완료된다

3	8	11	31	48	73
---	---	----	----	----	----

# 버블 정렬 알고리즘은 몇 번 연산을 해야할까?

- 양 옆에 원소들의 크기를 비교하는 연산
- 양 옆 원소들의 위치를 교환하는 연산



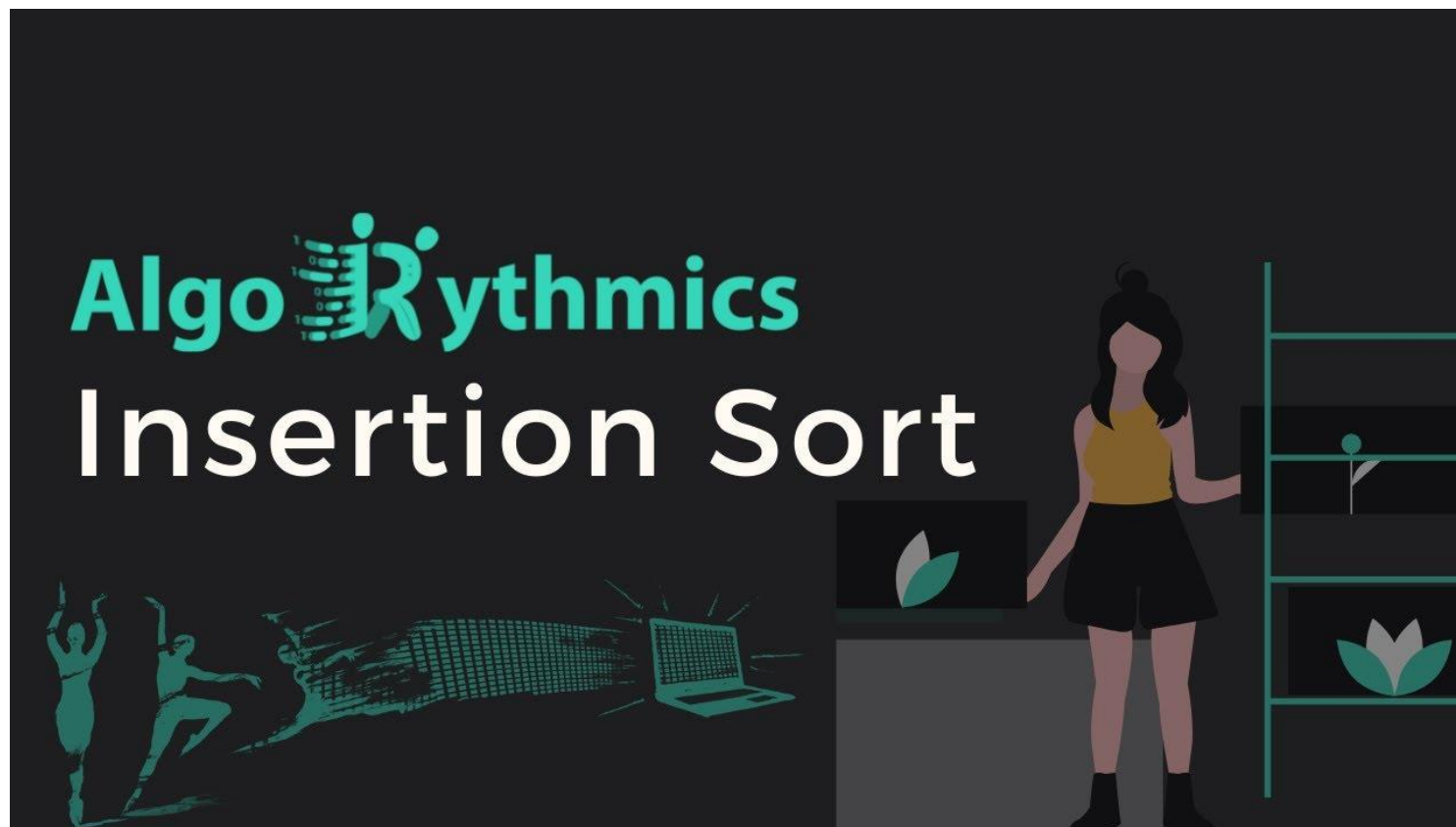
## 버블 정렬 알고리즘은 몇 번 연산을 해야할까?

- 양 옆에 원소들의 크기를 비교하는 연산
- 양 옆 원소들의 위치를 교환하는 연산
- 위에 연산들을  $N^2$ 번 반복해야 하니까 대충  $N^2$  만큼 연산하면 정렬이 됨  
(시간복잡도의 개념을 설명하지 않았기 때문에 대충 설명함.)

## 삽입 정렬 알고리즘의 개념

- 이미 정렬되어 있는  $i$  크기의 배열에 하나의 원소를 더 더하여 정렬된  $i+1$ 개 크기의 배열을 만드는 과정을 반복하는 알고리즘
- 조금 더 쉽게 설명하자면, 앞에 있는 원소부터 차례대로 보면서, 올바른 자리를 찾아서 집어넣는 알고리즘이다.
- 이해하기 쉽도록 유튜브 영상을 준비했다.

## 삽입 정렬 알고리즘의 작동 과정



# 삽입 정렬 알고리즘은 몇 번 연산을 해야할까?

- (1) 원소들의 크기를 비교하는 연산
- (2) 원소들의 위치를 교환하는 연산

## 삽입 정렬 알고리즘은 몇 번 연산을 해야할까?

- (1) 원소들의 크기를 비교하는 연산
- (2) 원소들의 위치를 교환하는 연산
- (1) 연산을  $N$ 번 반복하고, (2) 연산을  $N$ 번하면 정렬이 됨
- 근데, 이미 정렬되어 있는 배열을 정렬하면 (2) 연산을  $N$ 번 반복 안 해도 돼서 (1) 연산만 하면 정렬이 됨  
(시간복잡도의 개념을 설명하지 않았기 때문에 대충 설명함.)

# 파이썬으로 정렬 알고리즘 구현

```
selection_sort.py

def selection_sort(num_list):
    for i in range(len(num_list)):
        min_idx = i
        for j in range(i + 1, len(num_list)):
            if num_list[j] < num_list[min_idx]:
                min_idx = j
        num_list[i], num_list[min_idx] = num_list[min_idx], num_list[i]
```

snappify.com

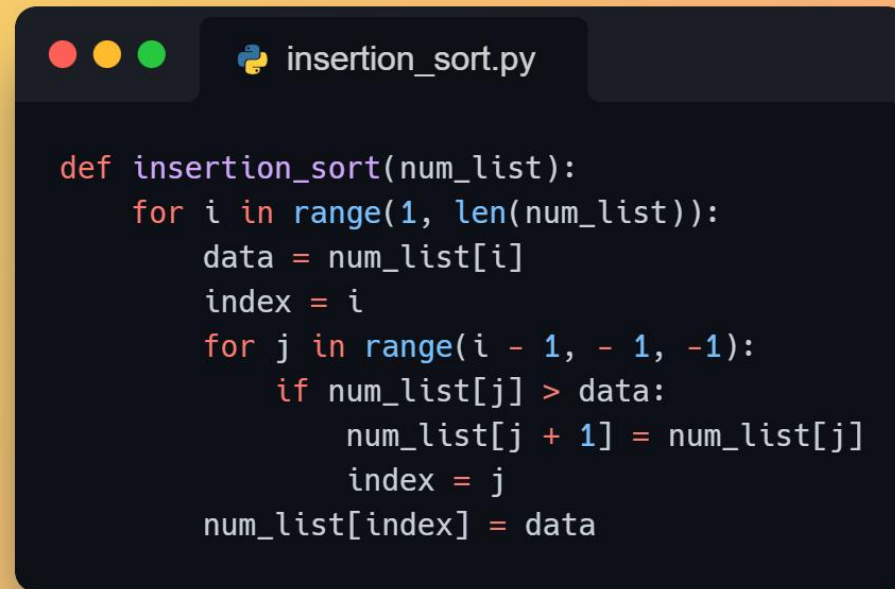
# 파이썬으로 정렬 알고리즘 구현



```
def bubble_sort(num_list):  
    for i in range(len(num_list) - 1):  
        for j in range(len(num_list) - 1):  
            if num_list[j] > num_list[j + 1]:  
                num_list[j], num_list[j + 1] = num_list[j + 1], num_list[j]
```

snappify.com

# 파이썬으로 정렬 알고리즘 구현



```
def insertion_sort(num_list):  
    for i in range(1, len(num_list)):  
        data = num_list[i]  
        index = i  
        for j in range(i - 1, -1, -1):  
            if num_list[j] > data:  
                num_list[j + 1] = num_list[j]  
                index = j  
        num_list[index] = data
```

snappify.com



# Thank You!

---

송기태 ([kitae040522@gmail.com](mailto:kitae040522@gmail.com))  
Soongsil Univ. (Computer Science and Engineering)