Review

01 정렬 알고리즘?

02 정렬 알고리즘 종류

03 질문&답변

정렬 알고리즘이란?

정렬 알고리즘

- **정렬(Sorting)**이란 데이터를 특정한 기준에 따라 순서대로 나열하는 것을 말합니다.
- 일반적으로 문제 상황에 따라서 적절한 정렬 알고리즘이 공식처럼 사용됩니다.

여러 개의 데이터(카드)를 어떻게 **정렬**할 수 있을까요?

정렬알고리즘 = 비교 + Swap (자리바꿈)

정렬 알고리즘이란?

선택정렬
삽입정렬
퀵정렬
계수정렬

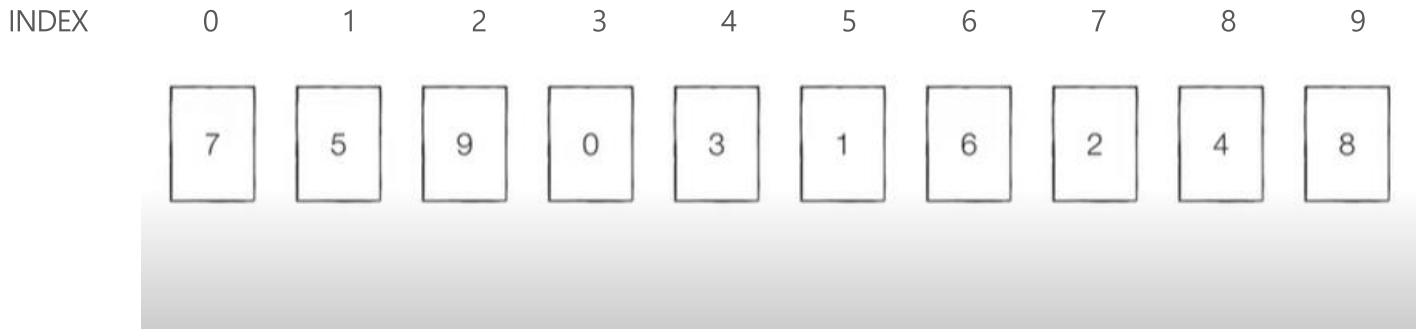
정렬 알고리즘	평균 시간 복잡도	공간 복잡도	특징
선택 정렬	$O(N^2)$	$O(N)$	아이디어가 매우 간단합니다.
삽입 정렬	$O(N^2)$	$O(N)$	데이터가 거의 정렬되어 있을 때는 가장 빠릅니다.
퀵 정렬	$O(NlogN)$	$O(N)$	대부분의 경우에 가장 적합하며, 충분히 빠릅니다.
계수 정렬	$O(N+K)$	$O(N+K)$	데이터의 크기가 한정되어 있는 경우에만 사용이 가능하지만 매우 빠르게 동작합니다.

Name	Best	Avg	Worst	Run-time(정수 60,000개) 단위: sec
삽입정렬	n	n^2	n^2	7.438
선택정렬	n^2	n^2	n^2	10.842
버블정렬	n^2	n^2	n^2	22.894
셸 정렬	n	$n^{1.5}$	n^2	0.056
퀵 정렬	$n\log_2 n$	$n\log_2 n$	n^2	0.014
힙 정렬	$n\log_2 n$	$n\log_2 n$	$n\log_2 n$	0.034
병합정렬	$n\log_2 n$	$n\log_2 n$	$n\log_2 n$	0.026

정렬 알고리즘이란?

선택정렬
삽입정렬
퀵정렬
계수정렬

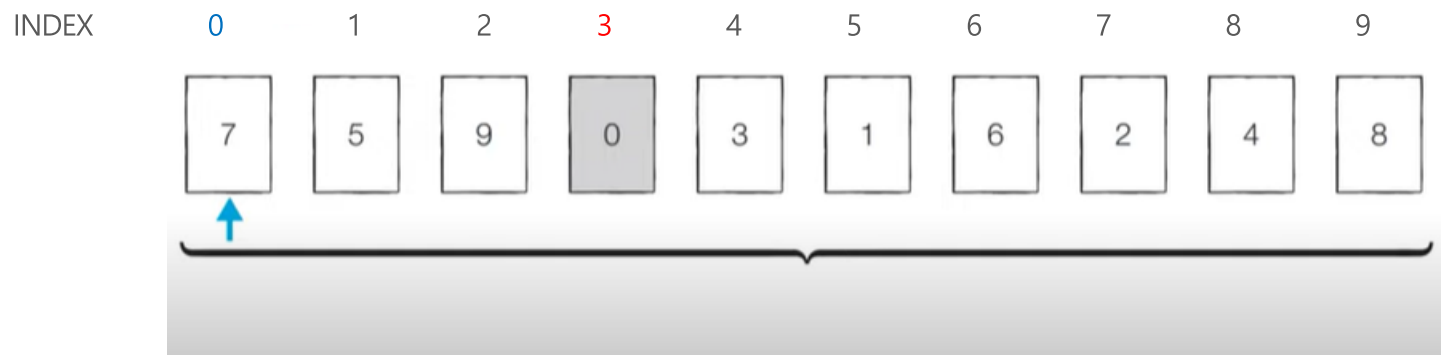
Key point : 처리되지 않은 데이터 중에서 가장 작은 데이터를 선택해 맨 앞에 있는 데이터와 바꾸는 것을 반복



정렬 알고리즘이란?

선택정렬
삽입정렬
퀵정렬
계수정렬

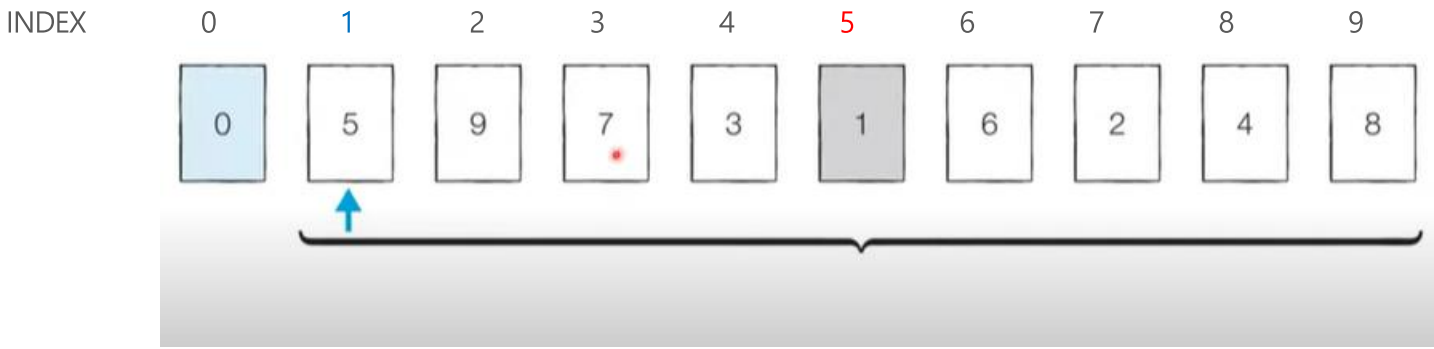
Key point : 처리되지 않은 데이터 중에서 가장 작은 데이터를 선택해 맨 앞에 있는 데이터와 바꾸는 것을 반복



정렬 알고리즘이란?

선택정렬
삽입정렬
퀵정렬
계수정렬

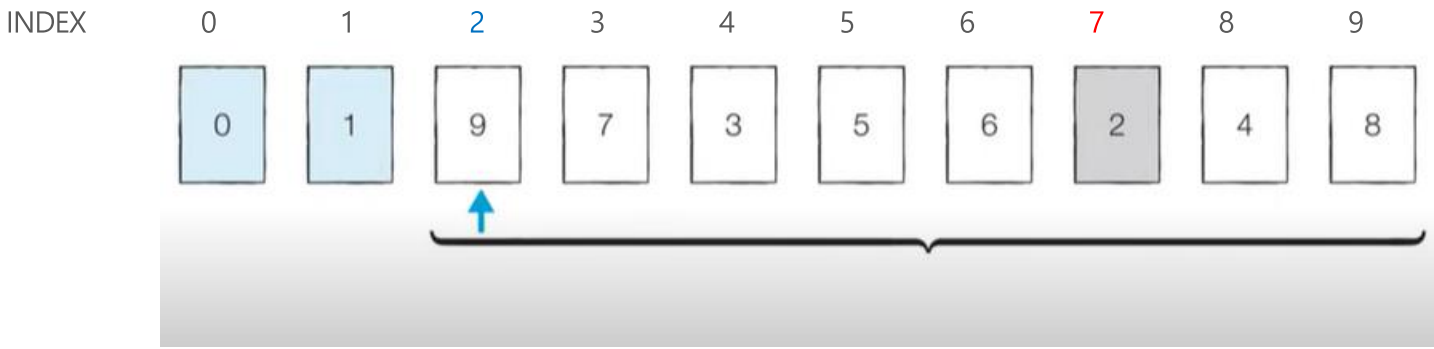
Key point : 처리되지 않은 데이터 중에서 가장 작은 데이터를 선택해 맨 앞에 있는 데이터와 바꾸는 것을 반복



정렬 알고리즘이란?

선택정렬
삽입정렬
퀵정렬
계수정렬

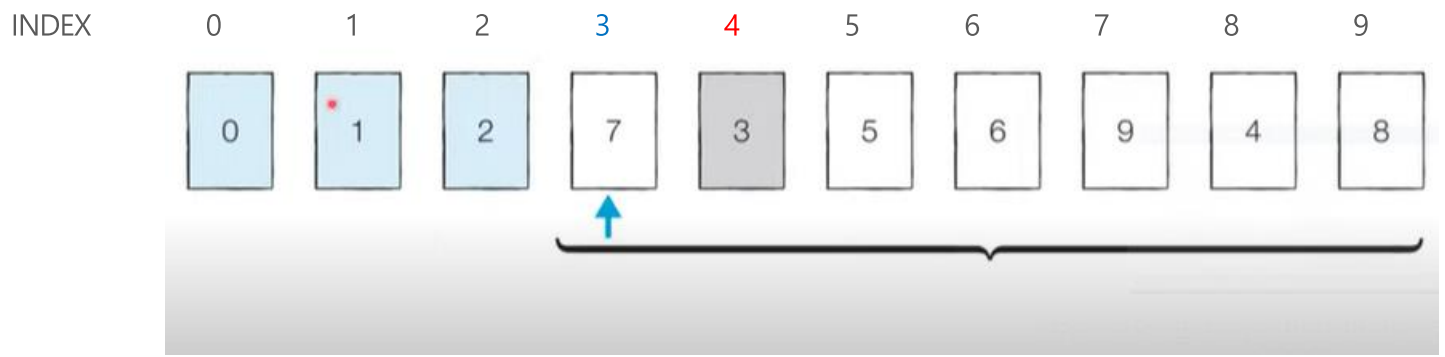
Key point : 처리되지 않은 데이터 중에서 가장 작은 데이터를 선택해 맨 앞에 있는 데이터와 바꾸는 것을 반복



정렬 알고리즘이란?

선택정렬
삽입정렬
퀵정렬
계수정렬

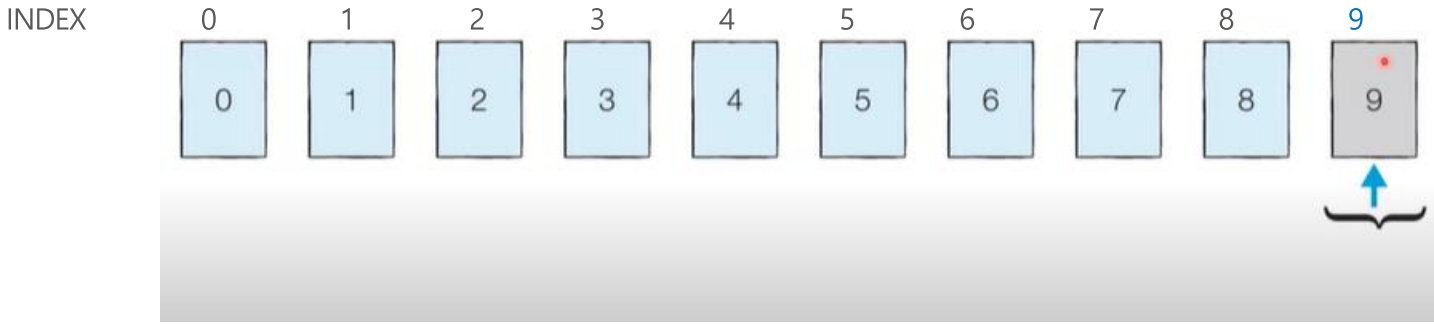
Key point : 처리되지 않은 데이터 중에서 가장 작은 데이터를 선택해 맨 앞에 있는 데이터와 바꾸는 것을 반복



정렬 알고리즘이란?

선택정렬
삽입정렬
퀵정렬
계수정렬

Key point : 처리되지 않은 데이터 중에서 가장 작은 데이터를 선택해 맨 앞에 있는 데이터와 바꾸는 것을 반복



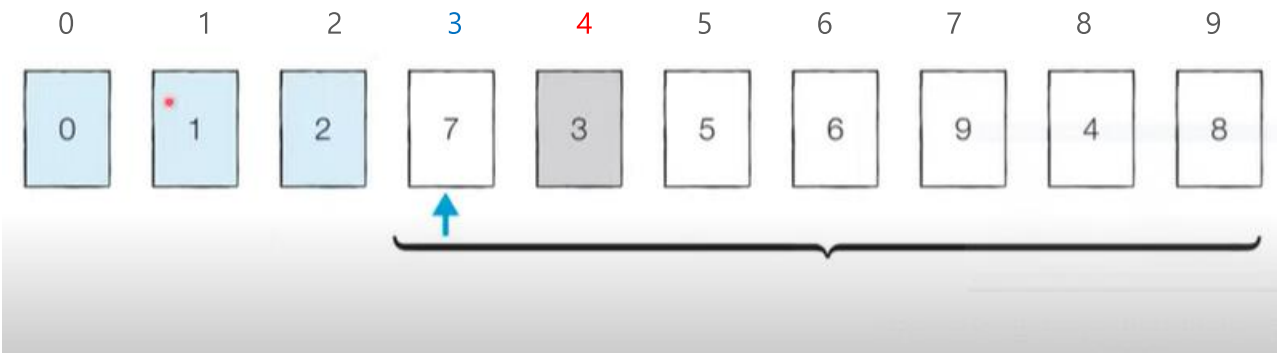
정렬 알고리즘이란?

선택정렬
삽입정렬
퀵정렬
계수정렬

```python
array = [7, 5, 9, 0, 3, 1, 6, 2, 4, 8]

for i in range(len(array)):
    min_index = i # 가장 작은 원소의 인덱스
    for j in range(i + 1, len(array)):
        if array[min_index] > array[j]:
            min_index = j
    array[i], array[min_index] = array[min_index], array[i] # 스와프

print(array)
```



정렬 알고리즘이란?

선택정렬
삽입정렬
퀵정렬
계수정렬

- 선택 정렬은 N번 만큼 가장 작은 수를 찾아서 맨 앞으로 보내야 합니다.
- 구현 방식에 따라서 사소한 오차는 있을 수 있지만, 전체 연산 횟수는 다음과 같습니다.

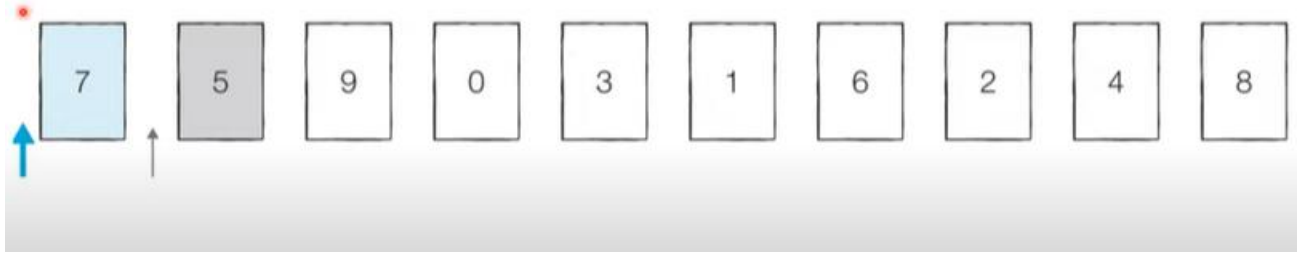
$$N + (N - 1) + (N - 2) + \dots + 2$$

- 이는 $(N^2 + N - 2) / 2$ 로 표현할 수 있는데, 빅오 표기법에 따라서 $\boldsymbol{O(N^2)}$이라고 작성합니다.

정렬 알고리즘이란?

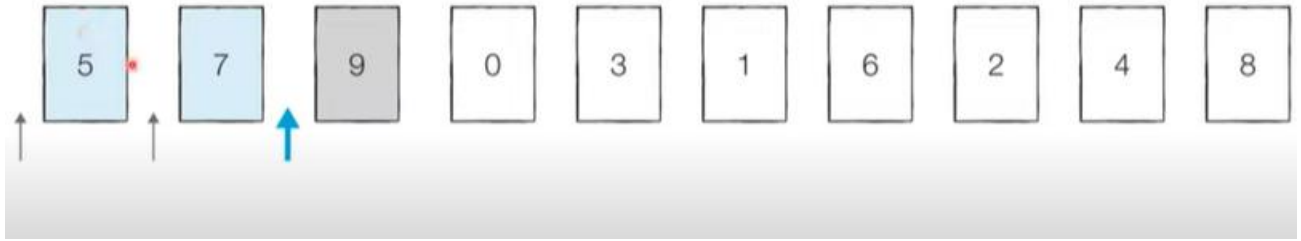
선택정렬
삽입정렬
퀵정렬
계수정렬

Key point : 처리되지 않은 데이터를 하나씩 골라 적절한 위치에 삽입



정렬 알고리즘이란?

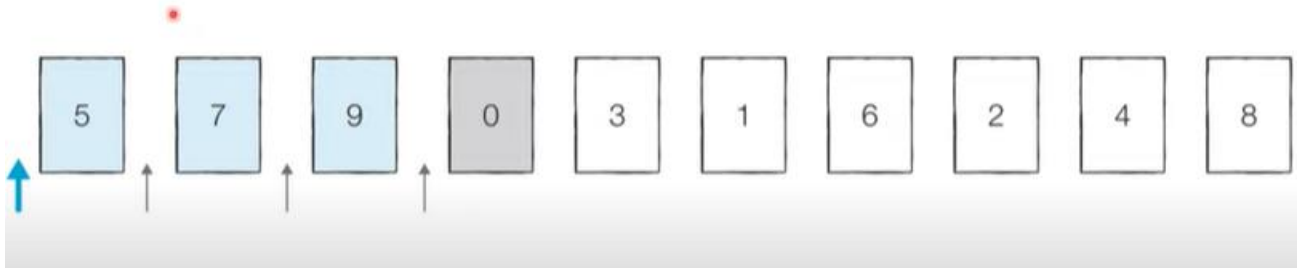
Key point : 처리되지 않은 데이터를 하나씩 골라 적절한 위치에 삽입



정렬 알고리즘이란?

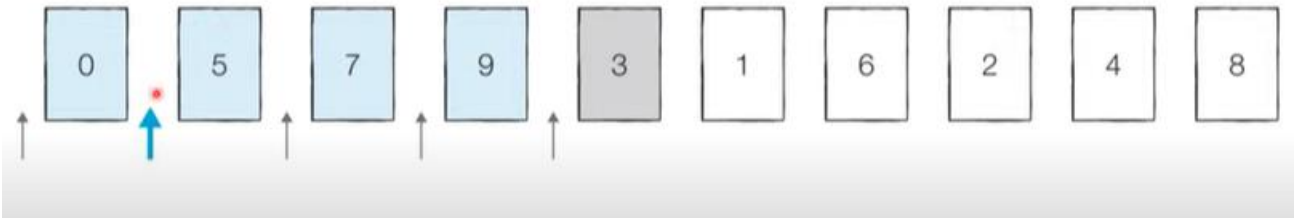
선택정렬
삽입정렬
퀵정렬
계수정렬

Key point : 처리되지 않은 데이터를 하나씩 골라 적절한 위치에 삽입



정렬 알고리즘이란?

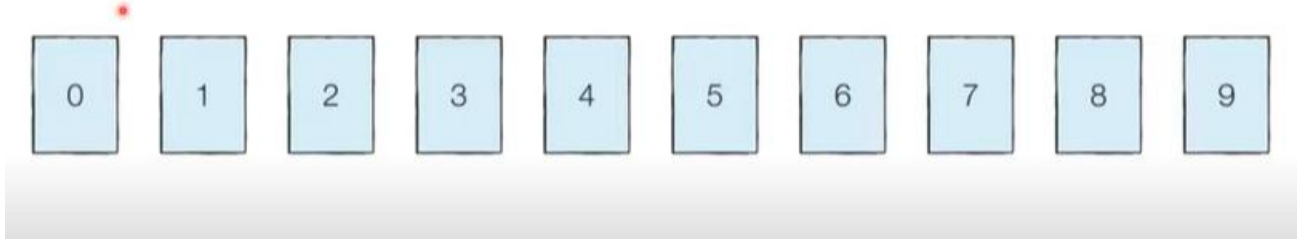
Key point : 처리되지 않은 데이터를 하나씩 골라 적절한 위치에 삽입



정렬 알고리즘이란?

선택정렬
삽입정렬
퀵정렬
계수정렬

Key point : 처리되지 않은 데이터를 하나씩 골라 적절한 위치에 삽입



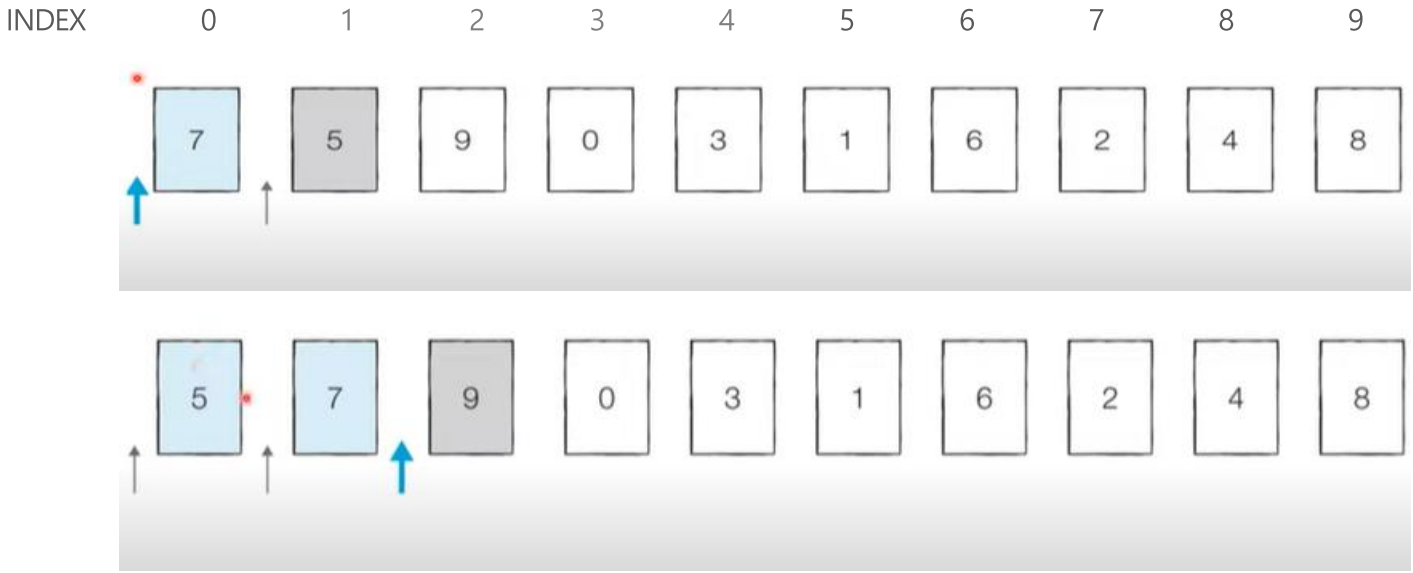
정렬 알고리즘이란?

선택정렬
삽입정렬
퀵정렬
계수정렬

```python
array = [7, 5, 9, 0, 3, 1, 6, 2, 4, 8]

for i in range(1, len(array)):
    for j in range(i, 0, -1): # 인덱스 i부터 1까지 1씩 감소하며 반복하는 문법
        if array[j] < array[j - 1]: # 한 칸씩 왼쪽으로 이동
            array[j], array[j - 1] = array[j - 1], array[j]
        else: # 자기보다 작은 데이터를 만나면 그 위치에서 멈춤
            break

print(array)
```



I 가 1인경우
j = 1 일때 array[j] =5 , array[j-1] =7 ,
if문에 걸러지므로 왼쪽인덱스로 자리위치가 이루어짐

I 가 2인경우
j = 2 일때 array[j] =9 , array[j-1] =7 ,
else문에 걸리므로
9는 인덱스 2 위치에 그대로 존재함.

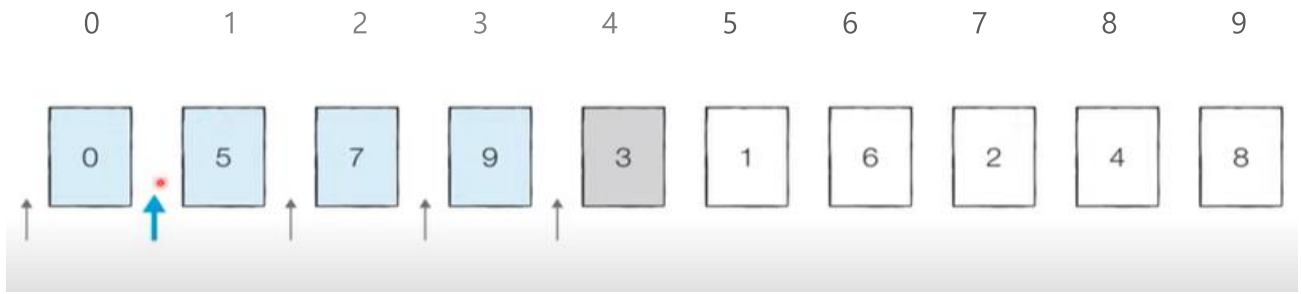
정렬 알고리즘이란?

선택정렬
삽입정렬
퀵정렬
계수정렬

```python
array = [7, 5, 9, 0, 3, 1, 6, 2, 4, 8]

for i in range(1, len(array)):
    for j in range(i, 0, -1): # 인덱스 i부터 1까지 1씩 감소하며 반복하는 문법
        if array[j] < array[j - 1]: # 한 칸씩 왼쪽으로 이동
            array[j], array[j - 1] = array[j - 1], array[j]
        else: # 자기보다 작은 데이터를 만나면 그 위치에서 멈춤
            break

print(array)
```



I 가 4인경우
j = 4 일때 array[j] =3 , array[j-1] =9 ,
3과 9는 자리변경 (if문에 걸려서)

0 5 7 3 9

j = 3 일때 array[j] =3 , array[j-1] =7,
3과 7은 자리변경 (if 문에 걸려서)

0 5 3 7 9

j = 2 일때 array[j] =3 , array[j-1] =5,
3과 5는 자리변경 (if 문에 걸려서)

0 3 5 7 9

j = 1 일때 array[j] =3 , array[j-1] =0,
break (j for문 탈출) -> I ++

정렬 알고리즘이란?

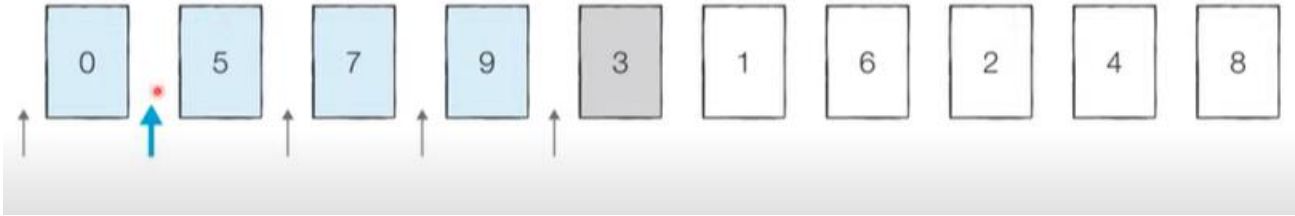
선택정렬
삽입정렬
퀵정렬
계수정렬

정렬 알고리즘	평균 시간 복잡도	공간 복잡도	특징
선택 정렬	$O(N^2)$	$O(N)$	아이디어가 매우 간단합니다.
삽입 정렬	$O(N^2)$	$O(N)$	데이터가 거의 정렬되어 있을 때는 가장 빠릅니다.
퀵 정렬	$O(NlogN)$	$O(N)$	대부분의 경우에 가장 적합하며, 충분히 빠릅니다.
계수 정렬	$O(N+K)$	$O(N+K)$	데이터의 크기가 한정되어 있는 경우에만 사용이 가능하지만 매우 빠르게 동작합니다.

```python
array = [7, 5, 9, 0, 3, 1, 6, 2, 4, 8]

for i in range(1, len(array)):
    for j in range(i, 0, -1): # 인덱스 i부터 1까지 1씩 감소하며 반복하는 문법
        if array[j] < array[j - 1]: # 한 칸씩 왼쪽으로 이동
            array[j], array[j - 1] = array[j - 1], array[j]
        else: # 자기보다 작은 데이터를 만나면 그 위치에서 멈춤
            break

print(array)
```



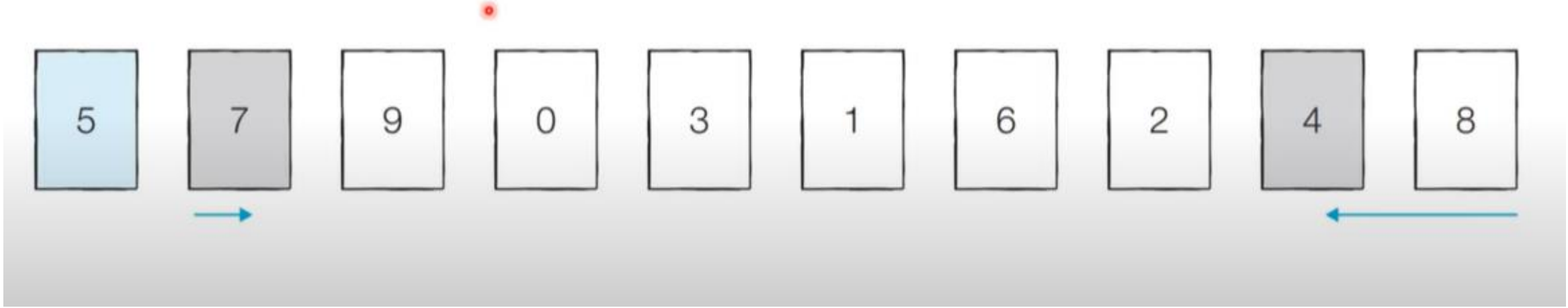
- 삽입 정렬의 시간 복잡도는 $O(N^2)$이며, 선택 정렬과 마찬가지로 반복문이 두 번 중첩되어 사용됩니다.
- 삽입 정렬은 현재 리스트의 데이터가 거의 정렬되어 있는 상태라면 매우 빠르게 동작합니다.
  - 최선의 경우 $O(N)$의 시간 복잡도를 가집니다.
  - 이미 정렬되어 있는 상태에서 다시 삽입 정렬을 수행하면 어떻게 될까요?

정렬 알고리즘이란?

- 기준 데이터를 설정하고 그 **기준보다 큰 데이터와 작은 데이터의 위치를 바꾸는 방법**입니다.
- 일반적인 상황에서 가장 많이 사용되는 정렬 알고리즘 중 하나입니다.
- 병합 정렬과 더불어 대부분의 프로그래밍 언어의 정렬 라이브러리의 근간이 되는 알고리즘입니다.
- 가장 기본적인 퀵 정렬은 **첫 번째 데이터를 기준 데이터(Pivot)로 설정**합니다.

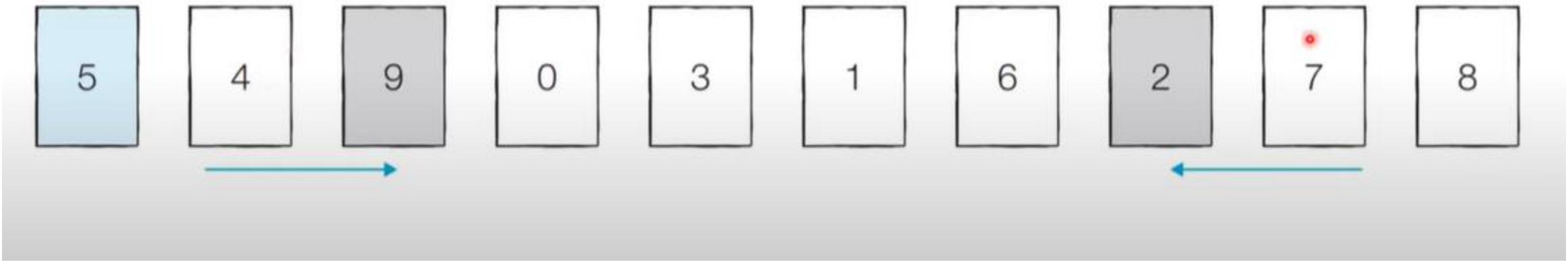
정렬 알고리즘이란?

선택정렬
삽입정렬
퀵정렬
계수정렬



정렬 알고리즘이란?

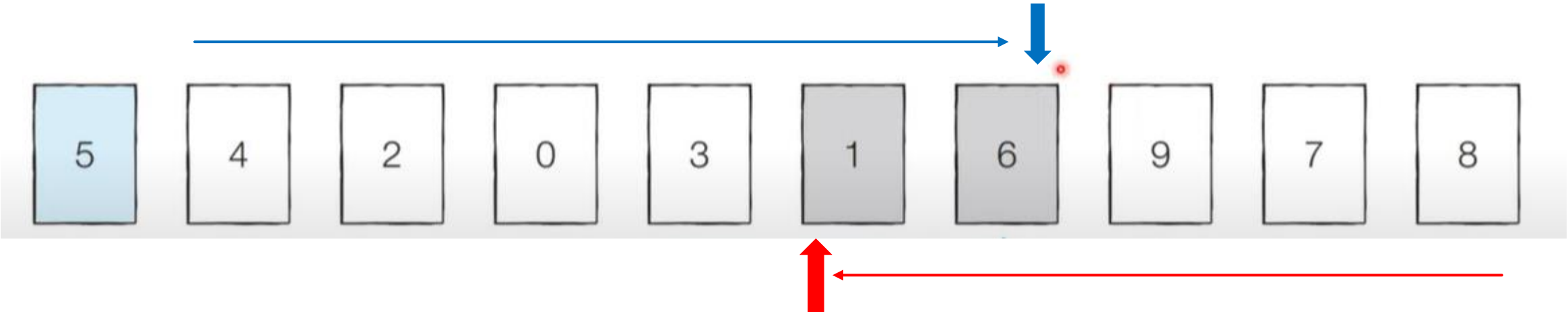
선택정렬
삽입정렬
퀵정렬
계수정렬



정렬 알고리즘이란?

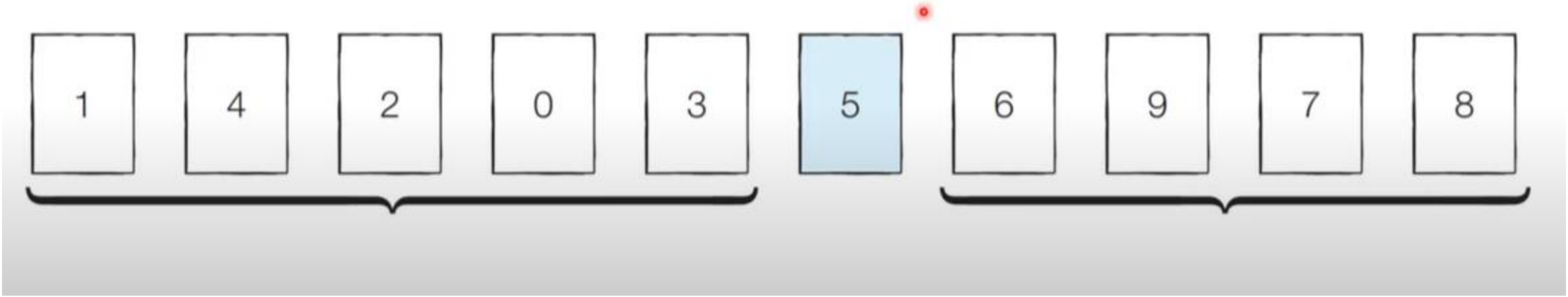
선택정렬
삽입정렬
퀵정렬
계수정렬

두 포인터가 겹쳐질때, pivot과 빨간포인터 인덱스와 자리바꿈!



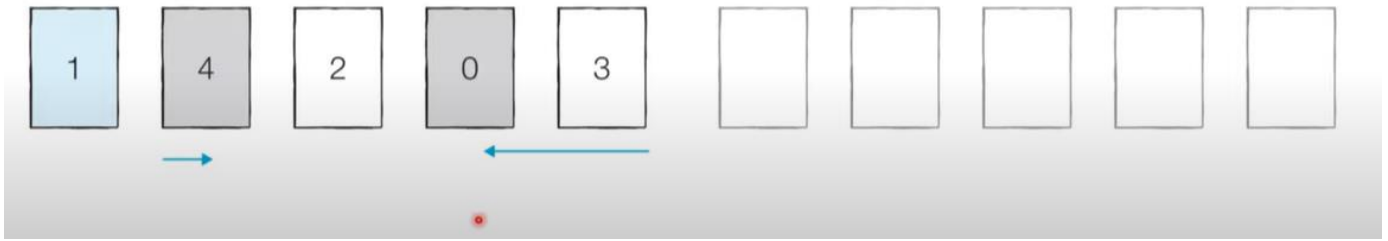
정렬 알고리즘이란?

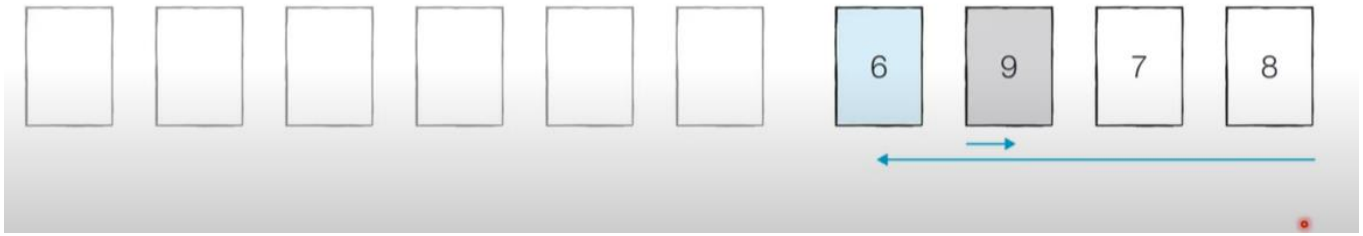
선택정렬
삽입정렬
퀵정렬
계수정렬



정렬 알고리즘이란?

선택정렬
삽입정렬
퀵정렬
계수정렬

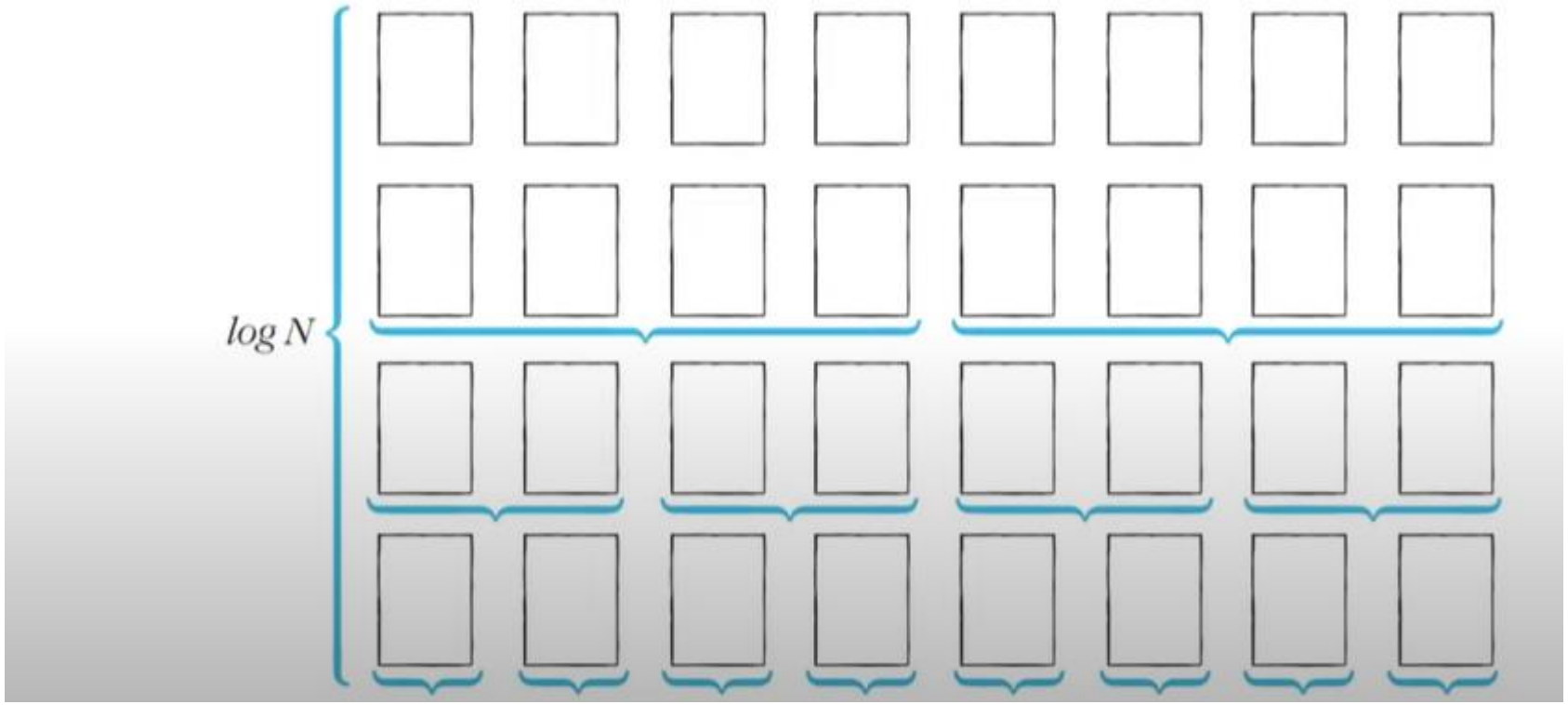




정렬 알고리즘이란?

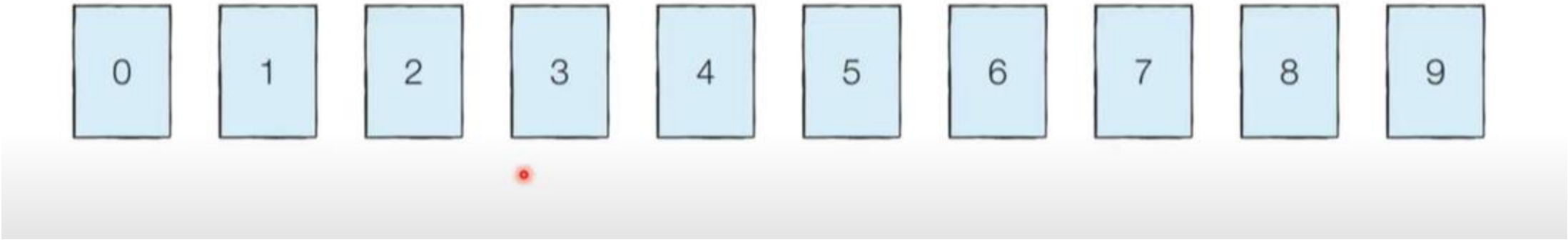
선택정렬
삽입정렬
퀵정렬
계수정렬

- 너비 X 높이 = $N \times logN = NlogN$



정렬 알고리즘이란?

선택정렬
삽입정렬
퀵정렬
계수정렬



정렬 알고리즘이란?

선택정렬
삽입정렬
퀵정렬
계수정렬

```python
array = [5, 7, 9, 0, 3, 1, 6, 2, 4, 8]

def quick_sort(array, start, end):
    if start >= end: # 원소가 1개인 경우 종료
        return
    pivot = start # 피벗은 첫 번째 원소
    left = start + 1
    right = end
    while(left <= right):
        # 피벗보다 큰 데이터를 찾을 때까지 반복
        while(left <= end and array[left] <= array[pivot]):
            left += 1
        # 피벗보다 작은 데이터를 찾을 때까지 반복
        while(right > start and array[right] >= array[pivot]):
            right -= 1
        if(left > right): # 엇갈렸다면 작은 데이터와 피벗을 교체
            array[right], array[pivot] = array[pivot], array[right]
        else: # 엇갈리지 않았다면 작은 데이터와 큰 데이터를 교체
            array[left], array[right] = array[right], array[left]
    # 분할 이후 왼쪽 부분과 오른쪽 부분에서 각각 정렬 수행
    quick_sort(array, start, right - 1)
    quick_sort(array, right + 1, end)

quick_sort(array, 0, len(array) - 1)
```

실행 결과

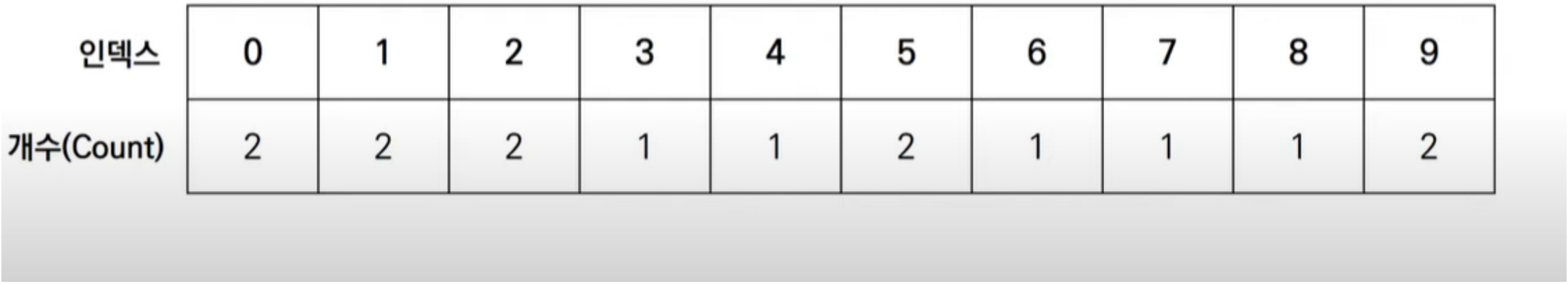
정렬 알고리즘이란?

- 특정한 조건이 부합할 때만 사용할 수 있지만 **매우 빠르게 동작하는** 정렬 알고리즘입니다.
  - 계수 정렬은 **데이터의 크기 범위가 제한되어 정수 형태로 표현할 수 있을 때** 사용 가능합니다.
- 데이터의 개수가 N, 데이터(양수) 중 최댓값이 K일 때 최악의 경우에도 수행 시간 $\boldsymbol{O(N+K)}$를 보장합니다.

정렬 알고리즘이란?

선택정렬
삽입정렬
퀵정렬
계수정렬

- **정렬할 데이터**: 7 5 9 0 3 1 6 2 9 1 4 8 0 5 2



인덱스	0	1	2	3	4	5	6	7	8	9
개수(Count)	2	2	2	1	1	2	1	1	1	2

정렬 알고리즘이란?

선택정렬
삽입정렬
퀵정렬
계수정렬

인덱스	0	1	2	3	4	5	6	7	8	9
개수(Count)	2	2	2	1	1	2	1	1	1	2

- **출력 결과**: 0 0

정렬 알고리즘이란?

선택정렬
삽입정렬
퀵정렬
계수정렬

인덱스	0	1	2	3	4	5	6	7	8	9
개수(Count)	2	2	2	1	1	2	1	1	1	2

- **출력 결과**: 0 0 1 1

정렬 알고리즘이란?

선택정렬
삽입정렬
퀵정렬
계수정렬

인덱스	0	1	2	3	4	5	6	7	8	9
개수(Count)	2	2	2	1	1	2	1	1	1	2

- **출력 결과**: 0 0 1 1 2 2

정렬 알고리즘이란?

선택정렬
삽입정렬
퀵정렬
계수정렬

인덱스	0	1	2	3	4	5	6	7	8	9
개수(Count)	2	2	2	1	1	2	1	1	1	2

- **출력 결과**: 0 0 1 1 2 2 3 4 5 5 6 7 8 9 9

정렬 알고리즘이란?

선택정렬
삽입정렬
퀵정렬
계수정렬

```python
# 모든 원소의 값이 0보다 크거나 같다고 가정
array = [7, 5, 9, 0, 3, 1, 6, 2, 9, 1, 4, 8, 0, 5, 2]
# 모든 범위를 포함하는 리스트 선언(모든 값은 0으로 초기화)
count = [0] * (max(array) + 1)

for i in range(len(array)):
    count[array[i]] += 1 # 각 데이터에 해당하는 인덱스의 값 증가

for i in range(len(count)): # 리스트에 기록된 정렬 정보 확인
    for j in range(count[i]):
        print(i, end=' ') # 띄어쓰기를 구분으로 등장한 횟수만큼 인덱스 출력
```

이상입니다

감사합니다