# SORT (2)

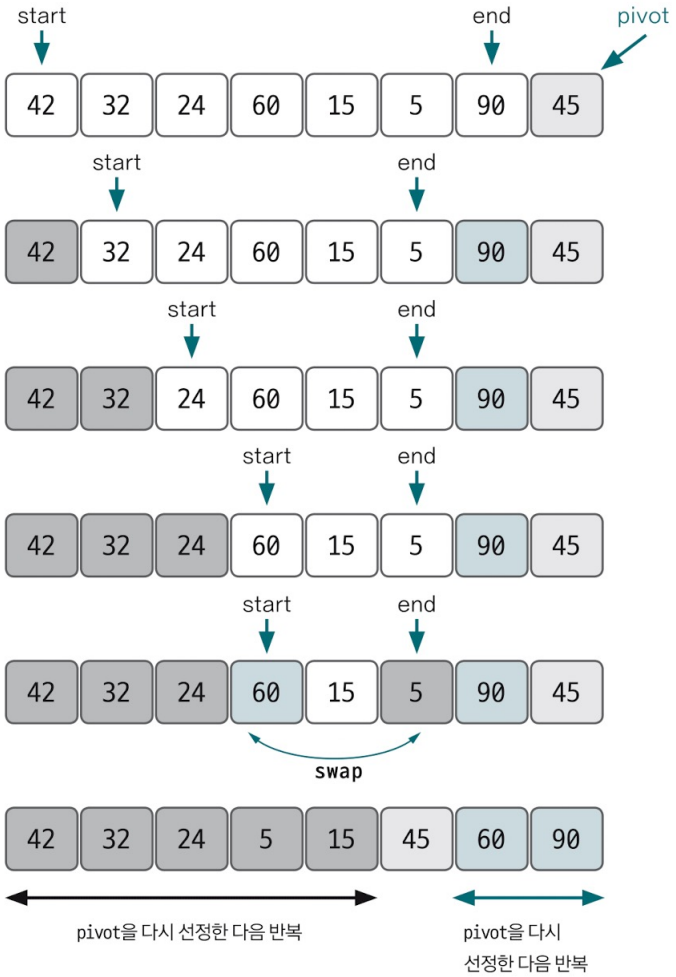
Algorithm

# 01

# Quick Sort

# 01. CONTENTS

## 퀵정렬




퀵 정렬 수행 방식

시간복잡도 : O(n^2)

# 01. CONTENTS

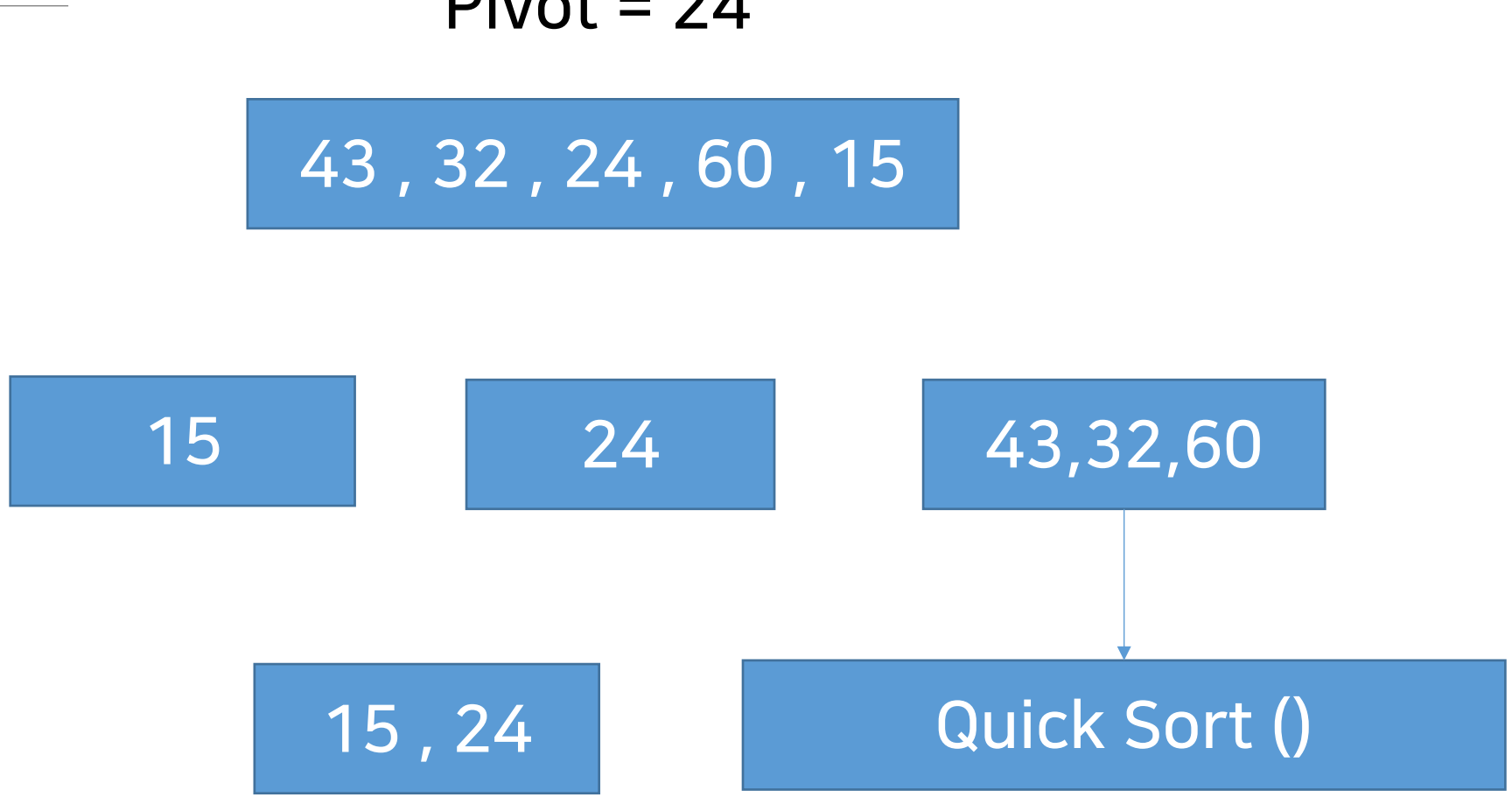
## 퀵정렬

```
def quickSort(arr):
    if len(arr) <= 1:
        #정렬할 원소가 없는 경우 그대로 리턴
        return arr
    pivot = arr[len(arr)//2]
    #pivot을 중앙값으로 선정함
    lesser, equal, greater = [], [],[]

    for num in arr:
        if num < pivot :
            lesser.append(num)
        elif num > pivot:
            greater.append(num)
        else :
            equal.append(num)
    return quickSort(lesser) + equal + quickSort(greater)
```

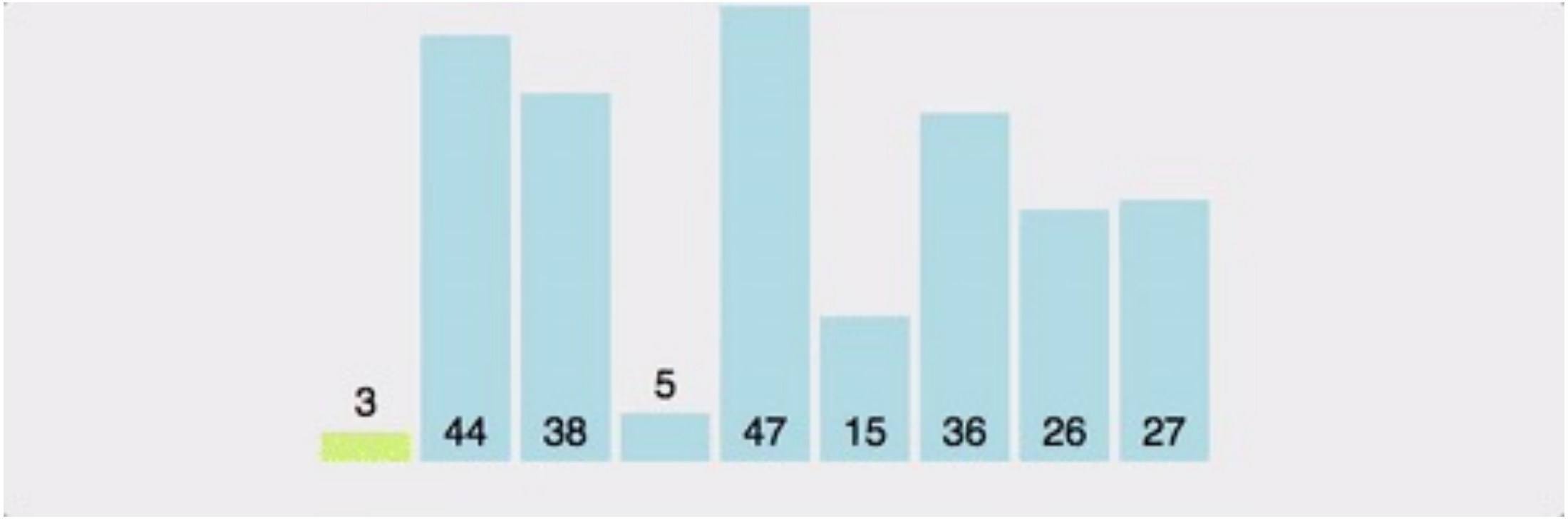
01. CONTENTS

퀵정렬



# 01. CONTENTS

퀵정렬

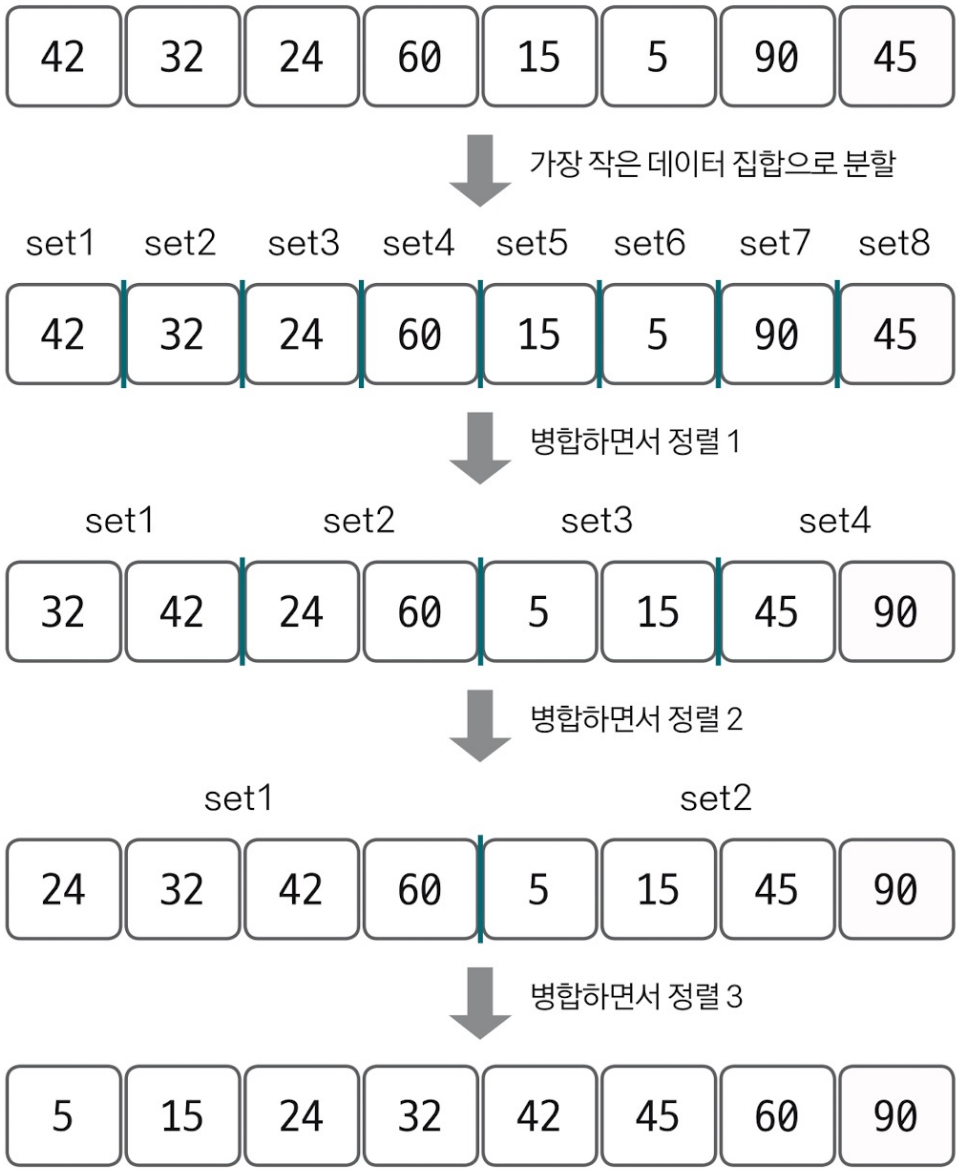


# 02

# Merge Sort

# 02. CONTENTS

## 병합정렬




시간복잡도 : O(nlogn)

# 02. CONTENTS

## 병합정렬

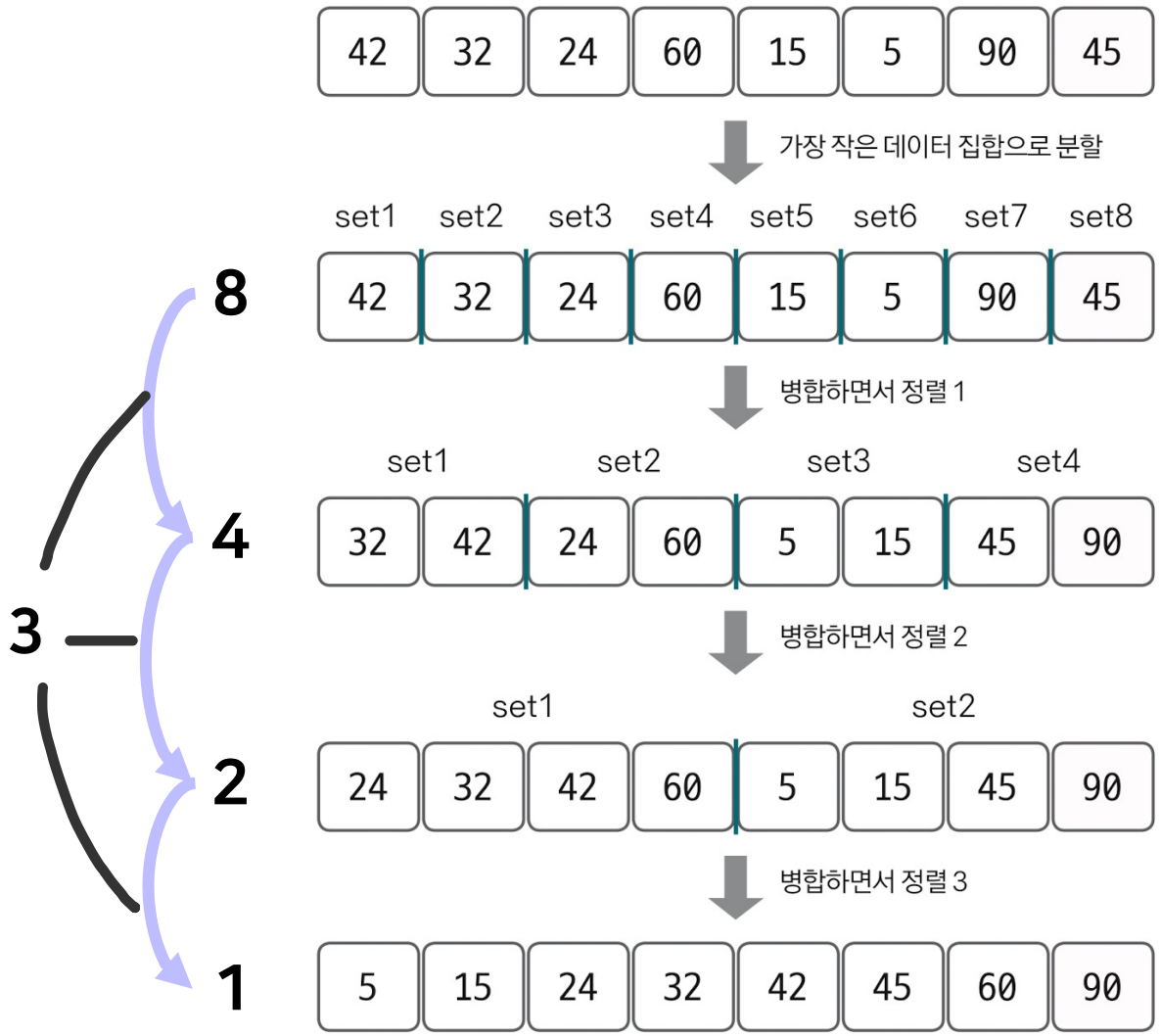
시간복잡도가 O(nlogn) 인 이유 ?

log 8 = 3

현재 접근할 수 있는 원소의 개수 = 8개

8개의 원소에 log 8 만큼의 연산을 한다

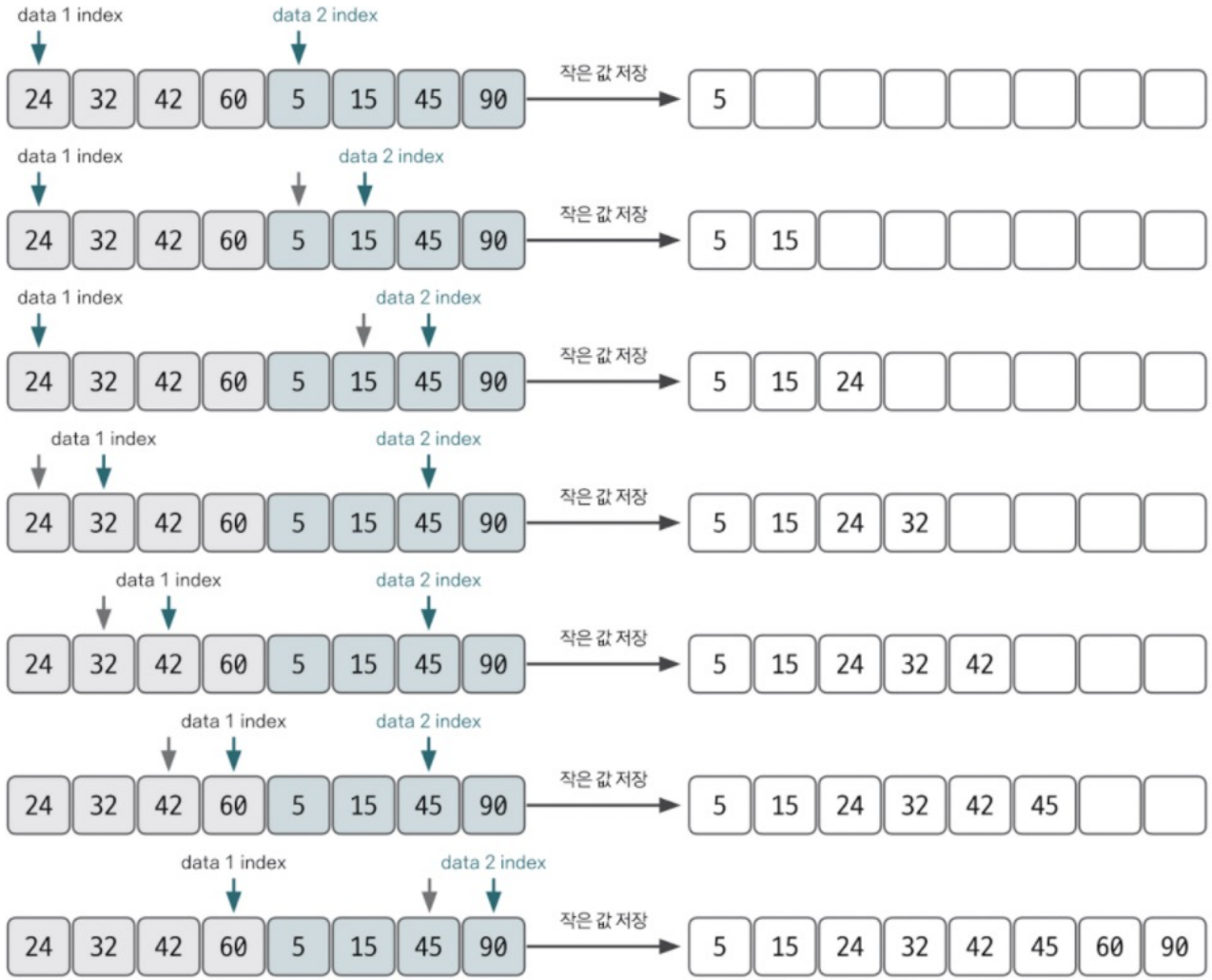
8 log 8 => n log n



# 02. CONTENTS

## 병합정렬

투포인터 방식



# 02. CONTENTS

## 병합정렬

```
def mergeSort(arr):
    if len(arr) < 2:
        return arr
    #길이가 1이 될때까지 쪼개기 위함
    mid = len(arr)//2
    #배열을 반으로 쪼갬
    low_arr = mergeSort(arr[:mid])
    high_arr = mergeSort(arr[mid:])

    merged_arr = []
    l = h = 0
    # l , low_arr 시작점
    # h , high_arr 시작점
    while l < len(low_arr) and h < len(high_arr):
        if low_arr[l] < high_arr[h]:
            merged_arr.append(low_arr[l])
            l += 1
        else:
            merged_arr.append(high_arr[h])
            h += 1
    merged_arr += low_arr[l:]
    merged_arr += high_arr[h:]

    return merged_arr
```

# 02. CONTENTS

## 병합정렬

# 03

## 관련해서 풀어보면 좋은 문제

# 03. CONTENTS

## 참조

### 버블 소트

☆

플래티넘 V

| 시간 제한 | 메모리 제한 | 제출 | 정답 | 맞힌 사람 | 정답 비율 |
|---|---|---|---|---|---|
| 1 초 | 512 MB | 21767 | 5926 | 3914 | 29.587% |

### 문제

N개의 수로 이루어진 수열 A[1], A[2], …, A[N]이 있다. 이 수열에 대해서 버블 소트를 수행할 때, Swap이 총 몇 번 발생하는지 알아내는 프로그램을 작성하시오.

버블 소트는 서로 인접해 있는 두 수를 바꿔가며 정렬하는 방법이다. 예를 들어 수열이 3 2 1 이었다고 하자. 이 경우에는 인접해 있는 3, 2가 바뀌어야 하므로 2 3 1 이 된다. 다음으로는 3, 1이 바뀌어야 하므로 2 1 3 이 된다. 다음에는 2, 1이 바뀌어야 하므로 1 2 3 이 된다. 그러면 더 이상 바꿔야 할 경우가 없으므로 정렬이 완료된다.

### 입력

첫째 줄에 N(1 ≤ N ≤ 500,000)이 주어진다. 다음 줄에는 N개의 정수로 A[1], A[2], …, A[N]이 주어진다. 각각의 A[i]는 0 ≤ |A[i]| ≤ 1,000,000,000의 범위에 들어있다.

### 출력

첫째 줄에 Swap 횟수를 출력한다

링크

# Thank you

Alogorithm