세상의 속도를
따라잡고 싶다면

Do it!

알고리즘 코딩 테스트

자바 편

이지스퍼블리싱

03

정렬 알고리즘

01 버블 정렬

02 선택 정렬

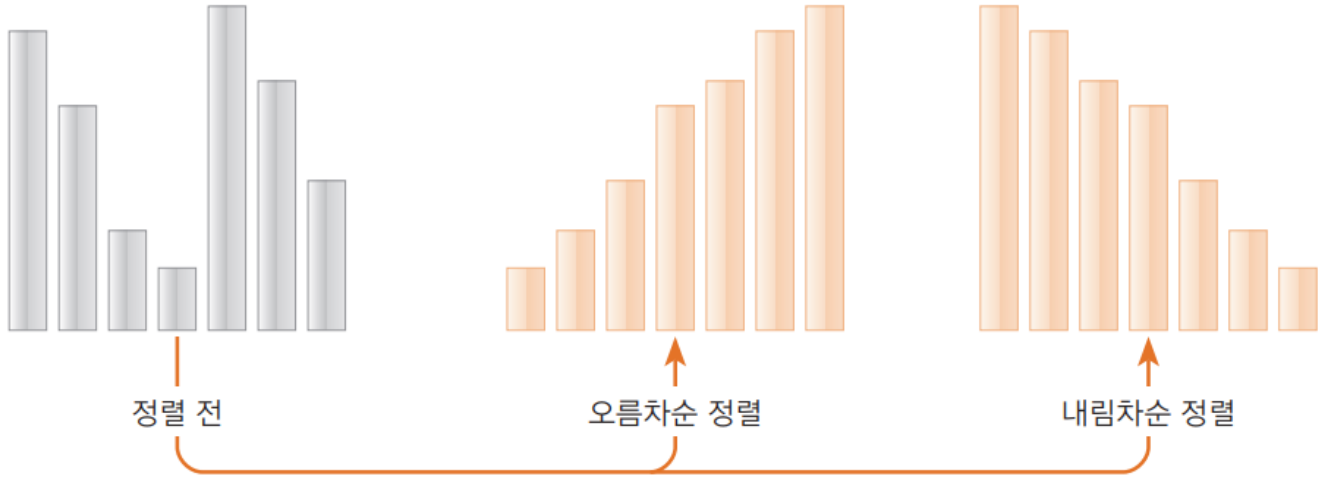
03 삽입 정렬

04 퀵 정렬

05 병합 정렬

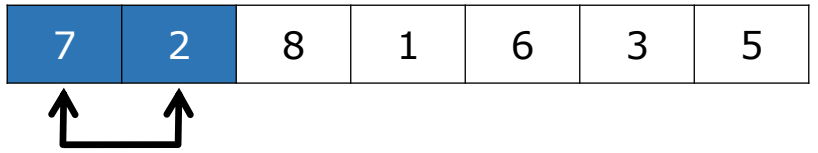
정렬 알고리즘

- 정렬(sorting)은 이름, 학번, 키 등 핵심 항목(key)의 대소 관계에 따라 데이터 집합을 일정한 순서로 나열하는 작업을 말한다.
- 데이터를 정렬하면 검색을 더 쉽게 할 수 있다.
- 값이 작은 데이터를 앞쪽에 놓으면 오름차순 정렬, 반대로 놓으면 내림차순 정렬이라고 부른다.

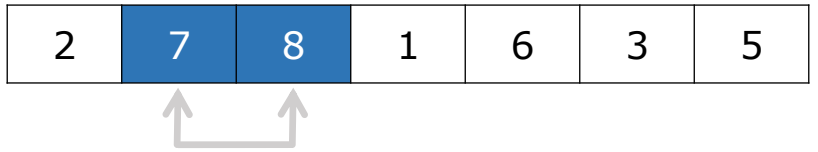


버블 정렬

- 이웃한 두 요소의 대소 관계를 비교하여 교환을 반복하는 알고리즘
- [7, 2, 8, 1, 6, 3, 5] 배열의 버블 정렬 과정
  - 앞의 두 요소 7과 2를 비교하여 왼쪽 값이 오른쪽 값보다 크면 교환

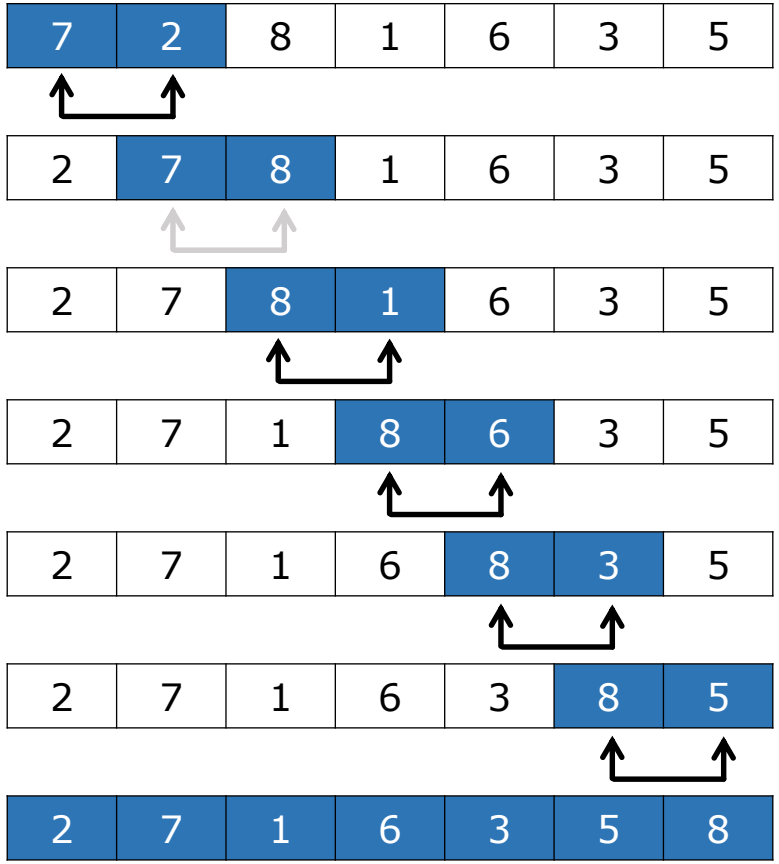


  - 그 다음 뒤쪽의 두 요소를 비교하여 왼쪽 값이 오른쪽 값보다 크면 교환
    - 7은 8보다 작으므로 교환 X



버블 정렬

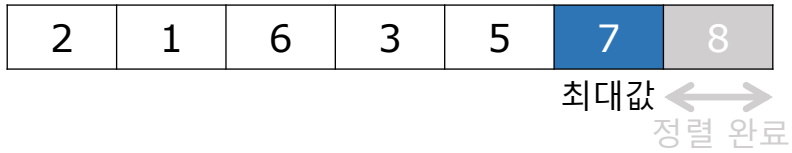
- 위와 같은 과정을 순차적으로 반복하면 다음 그림과 같이 정렬된다.



버블 정렬

- 처음부터 끝까지 이웃한 두 요소를 비교·교환을 완료하면 배열의 최대값이 가장 오른쪽에 위치

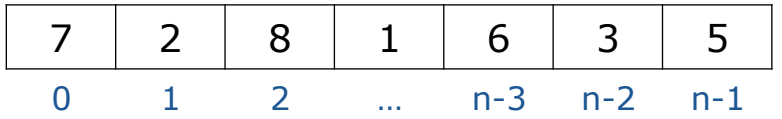
- 가장 오른쪽을 제외한 나머지 요소들을 위의 과정을 반복하면 그 다음 최대값이 가장 오른쪽에 위치



버블 정렬

- pseudo-code

```
arr // 배열
n // 배열의 크기
swap(i, j) // 배열의 i, j 번째 값을 교환

for i = 0 to n-2
    for j = 0 to n-i-2
        if arr[j] > arr[j+1] then
            swap(j, j+1)
```



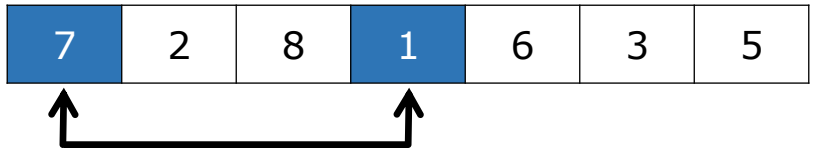
- 시간복잡도 $O(n^2)$

연습문제

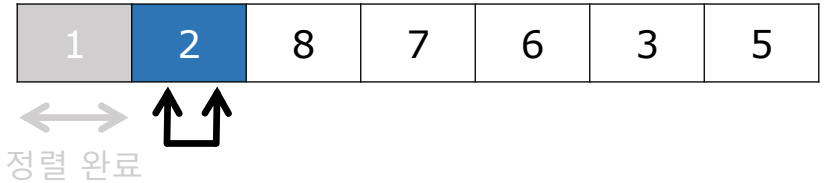
- 정올 – 1157 버블정렬
  - 정렬 연습
  - https://jungol.co.kr/problem/1157

선택 정렬

- 최소값을 맨 앞으로 이동, 그 다음 최소값을 두번째 위치로 이동 등을 반복하는 알고리즘
- [7, 2, 8, 1, 6, 3, 5] 배열의 선택 정렬 과정
  - 배열의 최소값을 찾아서 맨 앞의 요소와 교환

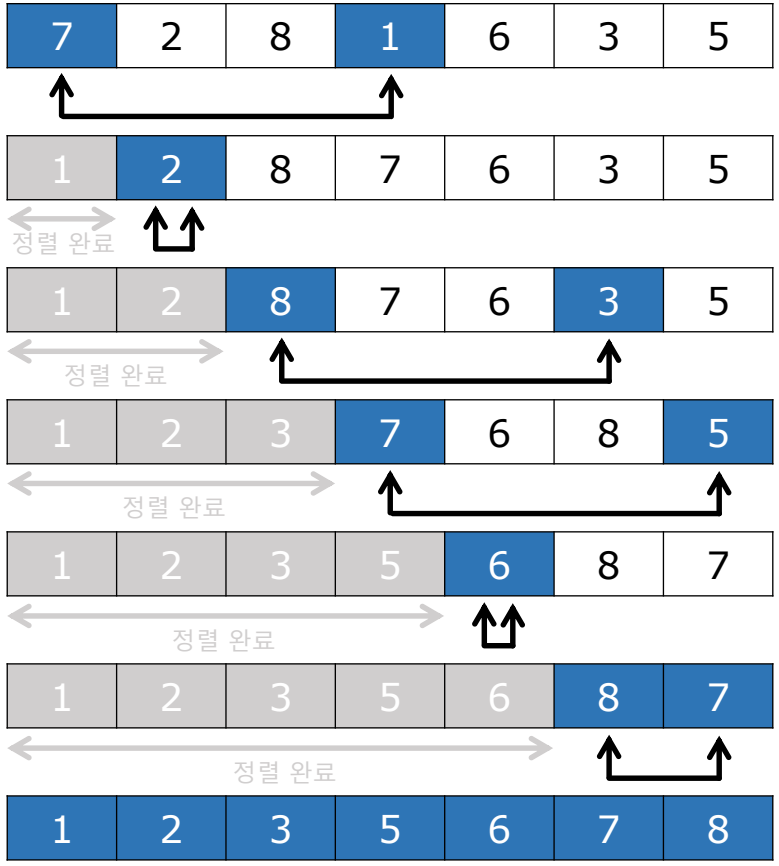


  - 맨 앞을 제외한 배열에서 위의 과정 수행
    - 그 다음 최소값이 2 이므로 자기 자신과 교환



선택 정렬

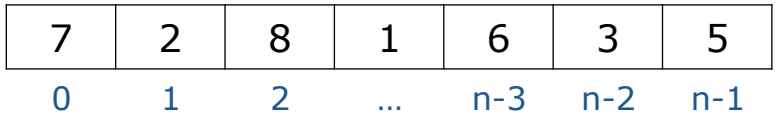
- 위와 같은 과정을 순차적으로 반복하면 다음 그림과 같이 정렬된다.



선택 정렬

- pseudo-code

```
arr // 배열
n // 배열의 크기
swap(i, j) // 배열의 i, j 번째 값을 교환

for i = 0 to n-1
    minIdx = i // 최소값의 위치 초기값
    for j = i+1 to n-1
        if arr[j] < arr[minIdx] then
            minIdx = j
    swap(minIdx, i)
```



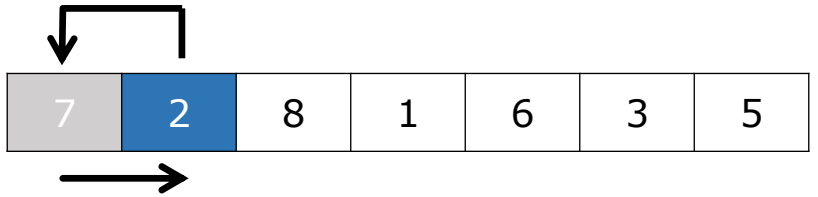
- 시간복잡도 $O(n^2)$

연습문제

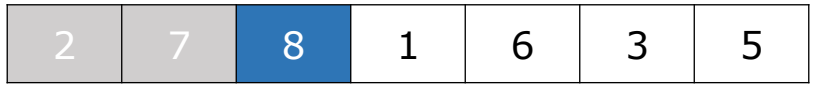
- 정올 – 1146 선택정렬
  - 정렬 연습
  - https://jungol.co.kr/problem/1146

삽입 정렬

- 선택한 요소의 앞쪽에 알맞은 위치에 삽입하는 과정을 반복하는 알고리즘
- [7, 2, 8, 1, 6, 3, 5] 배열의 삽입 정렬 과정
  - 두번째 요소를 앞쪽에 알맞은 위치에 삽입
    - 2는 7보다 앞쪽에 위치해야 하기 때문에 2를 0번 인덱스에 삽입하고 7은 뒤로 한 칸 씩 이동

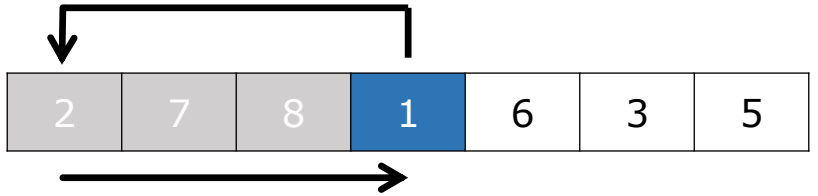


  - 세번째 요소를 앞쪽에 알맞은 위치에 삽입
    - 8은 2, 7보다 크기 때문에 변화 없음

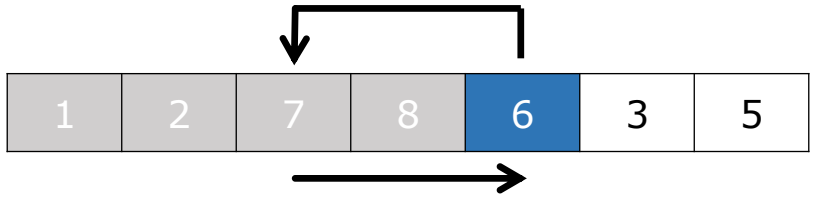


삽입 정렬

- [7, 2, 8, 1, 6, 3, 5] 배열의 삽입 정렬 과정
  - 네번째 요소를 앞쪽에 알맞은 위치에 삽입
    - 1은 2, 7, 8보다 앞쪽에 위치해야 하기 때문에 1을 0번 인덱스에 삽입하고 2, 7, 8은 뒤로 한 칸 씩 이동

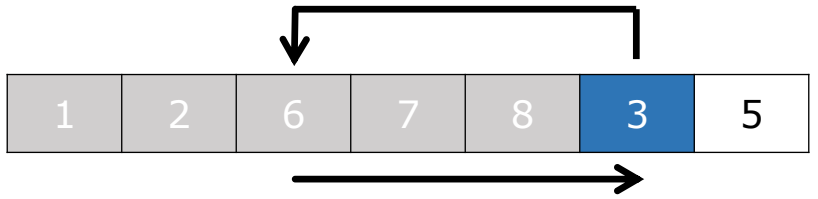


  - 다섯 번째 요소를 앞쪽에 알맞은 위치에 삽입
    - 6은 7, 8보다 앞쪽에 위치해야 하기 때문에 6을 2번 인덱스에 삽입하고 7, 8은 뒤로 한 칸 씩 이동

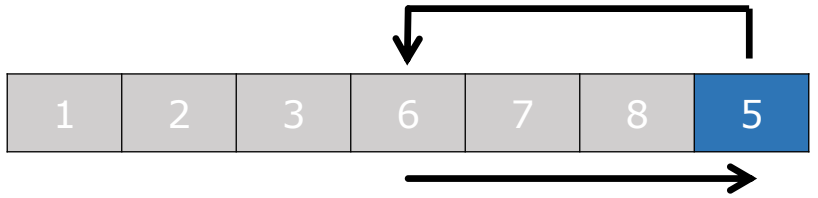


삽입 정렬

- [7, 2, 8, 1, 6, 3, 5] 배열의 삽입 정렬 과정
  - 여섯 번째 요소를 앞쪽에 알맞은 위치에 삽입
    - 3은 6, 7, 8보다 앞쪽에 위치해야 하기 때문에 3을 2번 인덱스에 삽입하고 6, 7, 8은 뒤로 한 칸 씩 이동



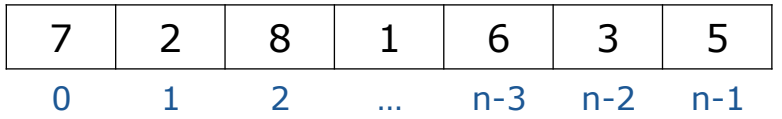
  - 일곱 번째 요소를 앞쪽에 알맞은 위치에 삽입
    - 5는 6, 7, 8보다 앞쪽에 위치해야 하기 때문에 5를 3번 인덱스에 삽입하고 6, 7, 8은 뒤로 한 칸 씩 이동



삽입 정렬

- pseudo-code

```
arr // 배열
n // 배열의 크기

for i = 1 to n-1
    insertIdx = i // 삽입할 인덱스 초기값
    for j = 0 to i-1
        if arr[j] > arr[i] then
            insertIdx = j
            break
    temp = arr[i]
    for j = i-1 to insertIdx
        arr[j+1] = arr[j]
    arr[insertIdx] = temp
```



- 시간복잡도 $O(n^2)$

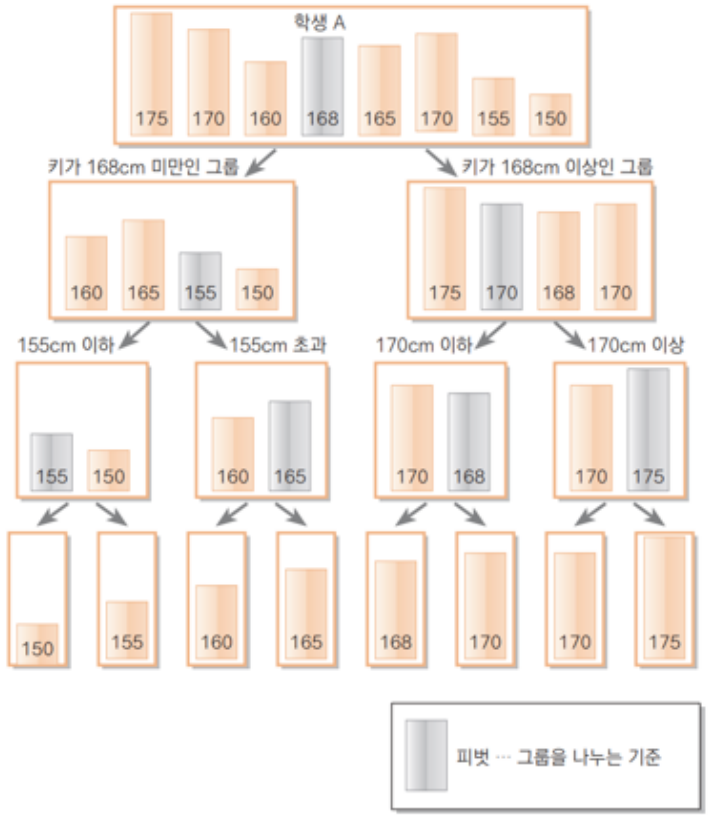
연습문제

- 정올 – 1158 삽입정렬
  - 정렬 연습
  - https://jungol.co.kr/problem/1158

- 정올 – 1814 삽입정렬 횟수 세기
  - 정렬 연습
  - https://jungol.co.kr/problem/1814

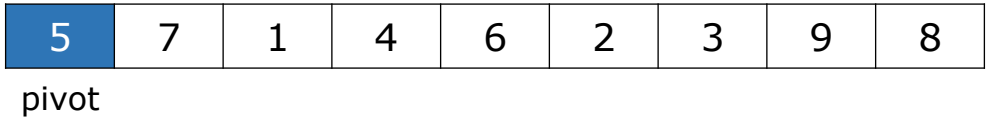
퀵 정렬

- 기준 값(pivot)을 기준으로 작은 값과 큰 값으로 나누는 작업을 반복하는 알고리즘
- 보통 재귀 호출을 이용

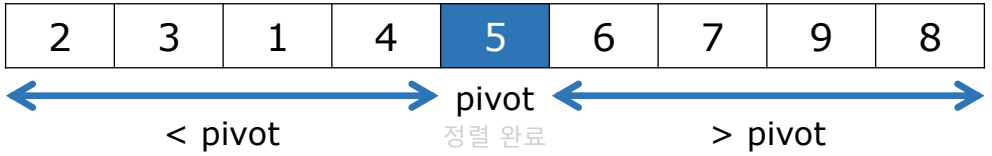


퀵 정렬

- [5, 7, 1, 4, 6, 2, 3, 9, 8] 배열의 퀵 정렬 과정
  - 첫 번째 값을 기준(pivot) 값으로 설정



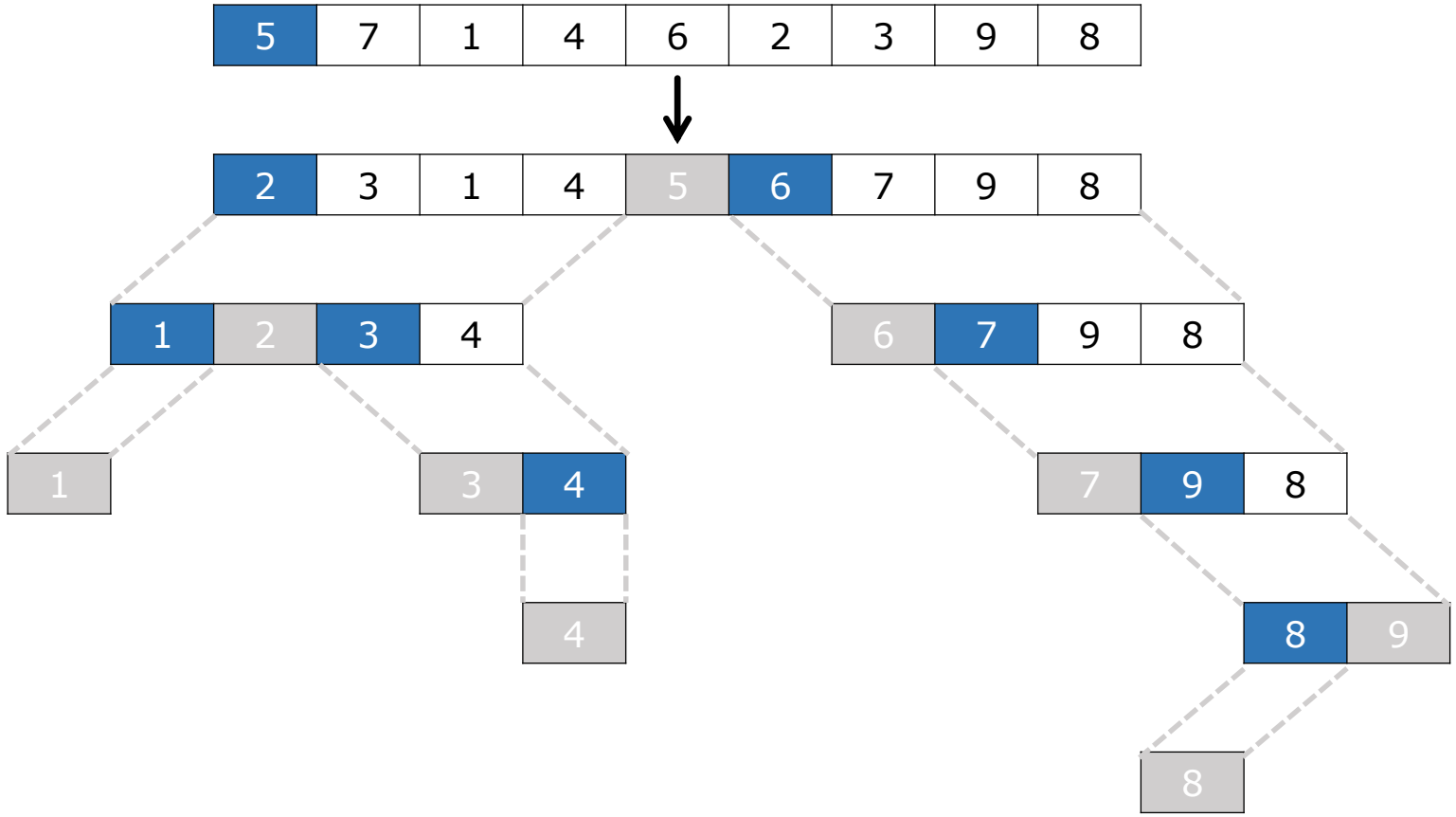
  - 기준(pivot) 값보다 작은 값은 왼쪽, 큰 값은 오른쪽에 배치



  - 기준(pivot) 값 위치의 왼쪽 부분과 오른쪽 부분을 위의 과정 반복

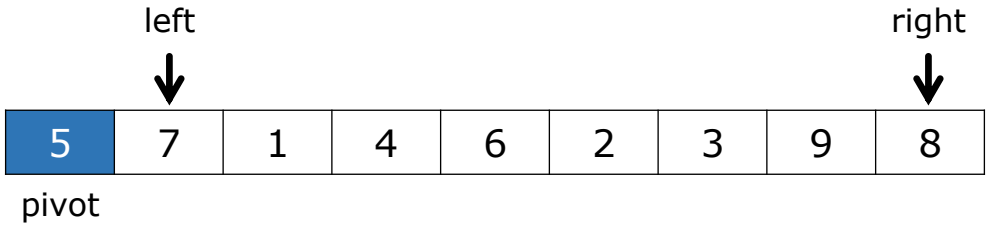
퀵 정렬

- [5, 7, 1, 4, 6, 2, 3, 9, 8] 배열의 퀵 정렬 과정

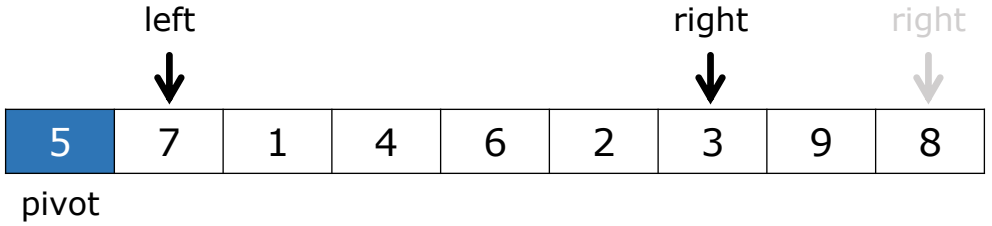


퀵 정렬

- [5, 7, 1, 4, 6, 2, 3, 9, 8] 배열의 퀵 정렬 과정 – pivot을 기준으로 값 나누는 과정
  - 첫 번째 값을 기준(pivot) 값, 나머지 배열에서 가장 왼쪽을 left, 가장 오른쪽을 right 포인터 설정

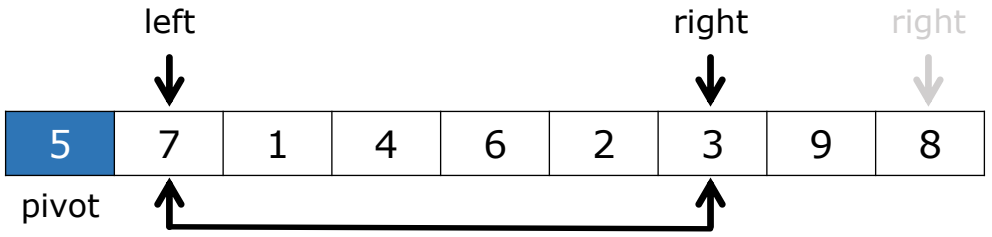


  - 기준 값보다 큰 값이 나올 때까지 left 1씩 증가, 기준 값보다 작은 값이 나올 때까지 right 1씩 감소

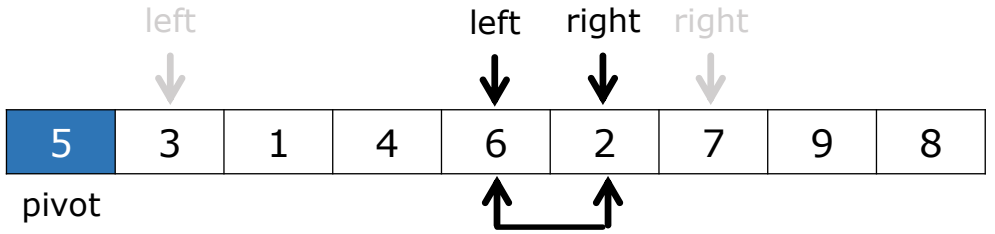


퀵 정렬

- [5, 7, 1, 4, 6, 2, 3, 9, 8] 배열의 퀵 정렬 과정 – pivot을 기준으로 값 나누는 과정
  - left의 값과 right의 값 교환

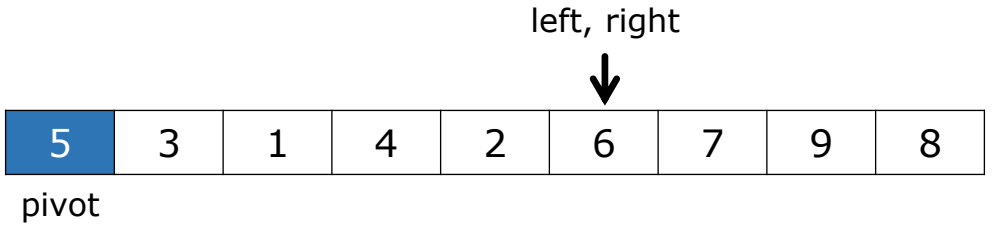


  - 같은 방식으로 기준 값보다 큰 값이 나올 때까지 left 1씩 증가, 기준 값보다 작은 값이 나올 때까지 right 1씩 감소 후 교환

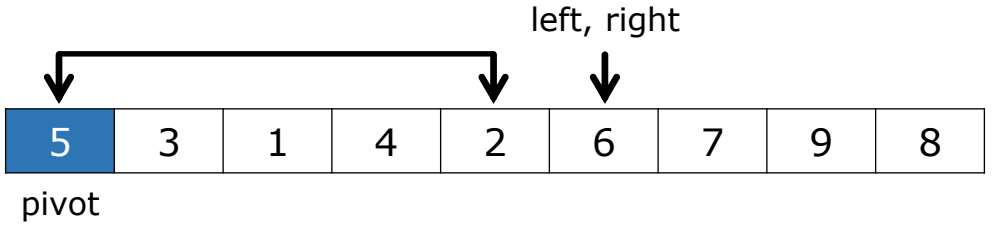


퀵 정렬

- [5, 7, 1, 4, 6, 2, 3, 9, 8] 배열의 퀵 정렬 과정 – pivot을 기준으로 값 나누는 과정
  - left와 right가 같아질 때까지 위의 과정을 반복



  - left의 값이 기준 값보다 크면 left-1의 값과 기준 값 교환, 그렇지 않으면 left의 값과 기준 값 교환



퀵 정렬

- [5, 7, 1, 4, 6, 2, 3, 9, 8] 배열의 퀵 정렬 과정 – pivot을 기준으로 값 나누는 과정
  - 기준 값을 기준으로 왼쪽 배열과 오른쪽 배열을 위의 과정 반복

퀵 정렬

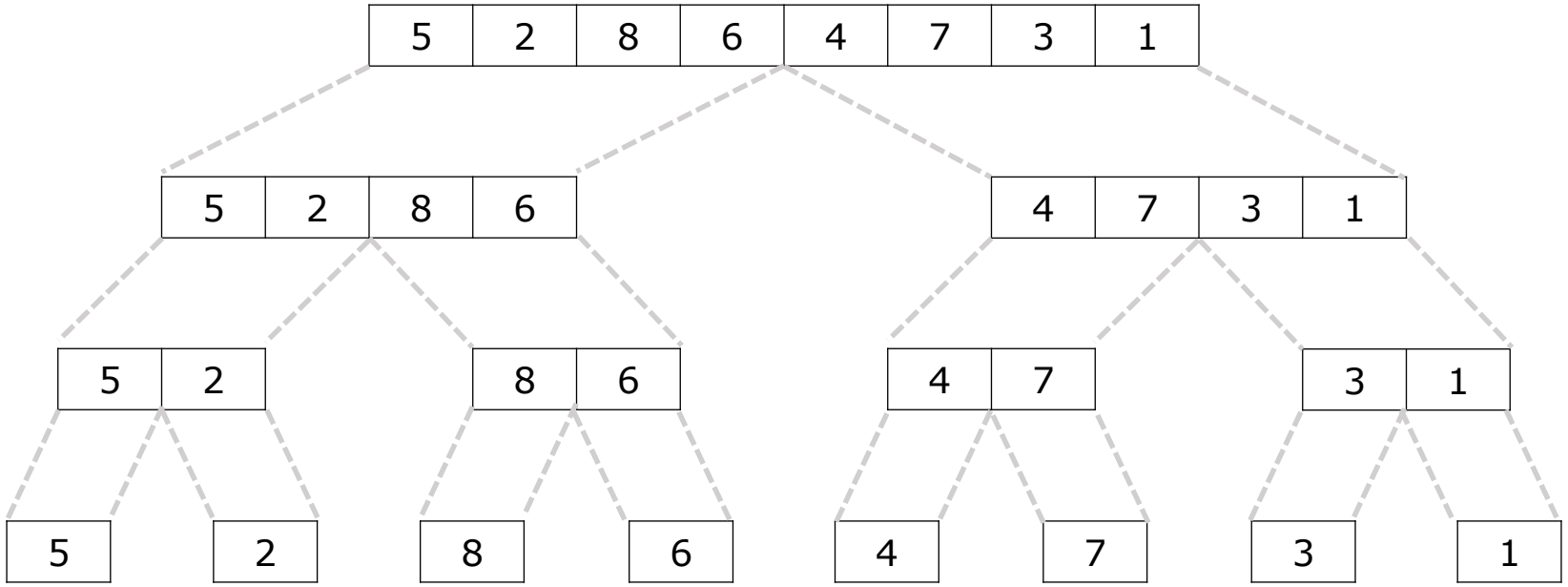
- pseudo-code

```
arr // 배열
n // 배열의 크기
partition(left, right) // 배열 left부터 right까지 pivot 기준으로 나누고 pivot 위치 반환

quickSort(left, right)
    if left >= right then 종료
    center = partition(left, right)
    quickSort(left, center-1)
    quickSort(center+1, right)

quickSort(0, n-1)
```

- 시간복잡도는 평균 nlogn이지만 최악의 경우 $O(n^2)$

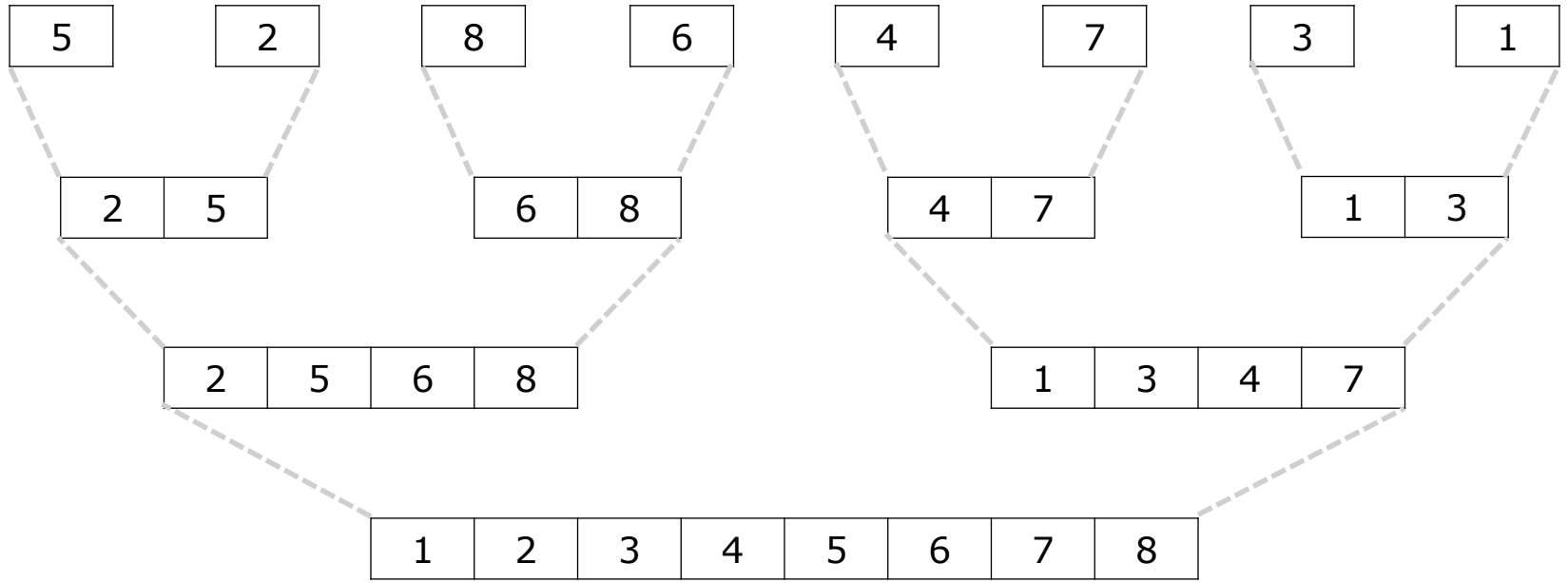
병합 정렬

- 배열을 분할하고 분할한 배열을 정복하며 정렬하는 알고리즘
- 주어진 배열을 분할하여 더 이상 나눌 수 없을 때까지 분할



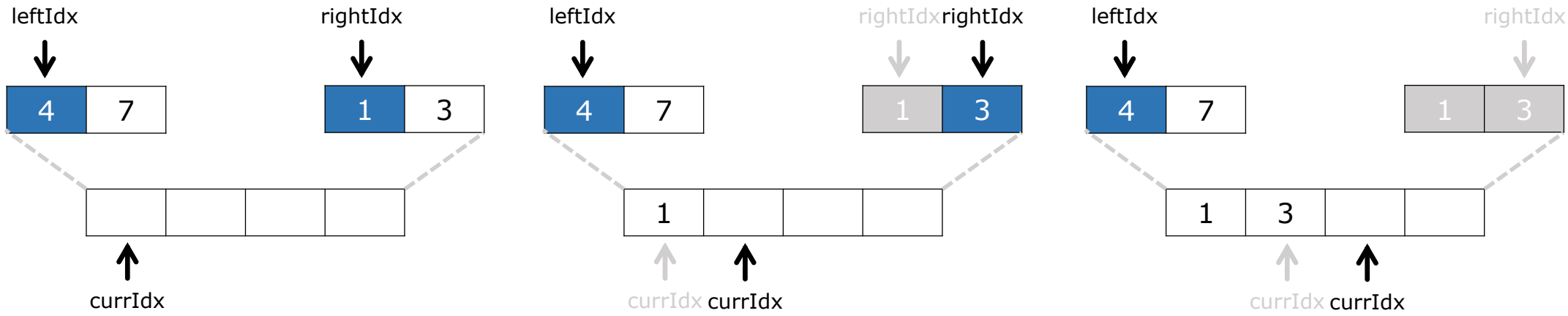
병합 정렬

- 인접한 부분 배열들을 정렬하면서 병합



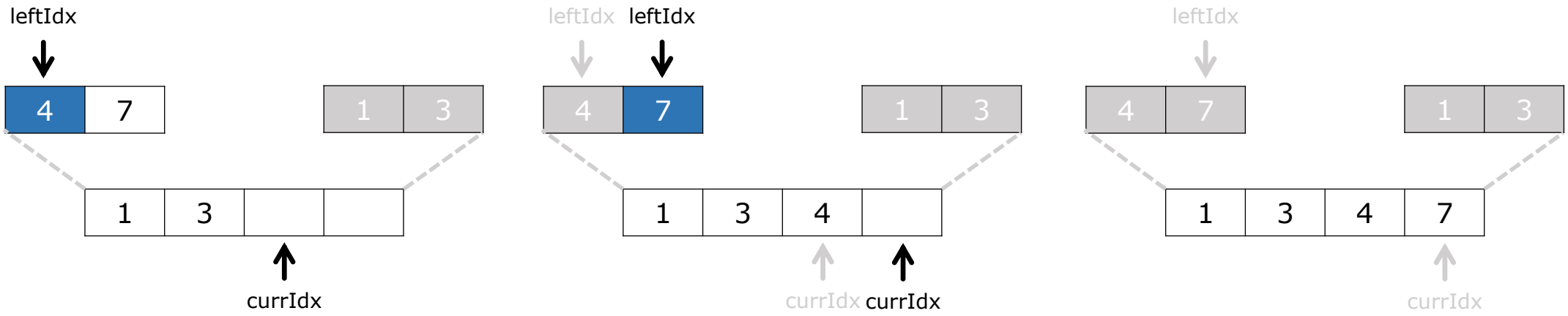
병합 정렬

- 인접한 부분 배열들은 정렬된 상태이므로 앞부분부터 순차적으로 비교하면서 병합하며 정렬
  - leftIdx의 값과 rightIdx의 값을 비교하여 작은 값을 currIdx에 추가 후 해당 인덱스 1씩 증가



병합 정렬

- 인접한 부분 배열들은 정렬된 상태이므로 앞부분부터 순차적으로 비교하면서 병합하며 정렬
  - 더 이상 비교할 값이 없으면 남은 값들을 순차적으로 currIdx에 추가



- 부분 배열 병합의 시간 복잡도 O(N)

병합 정렬

- pseudo-code

```
arr // 배열
n // 배열의 크기

mergeSort(left, right)
    if left >= right then 종료
    center = (left + right) / 2
    // Divide
    mergeSort(left, center)
    mergeSort(center+1, right)
    // Conquer
    // center를 기준으로 왼쪽, 오른쪽 배열을 병합하며 정렬

mergeSort(0, n-1)
```

- 시간복잡도 $O(n \log n)$

정렬 알고리즘

알고리즘	최악	최선	평균
버블 정렬	n^2	n^2	n^2
선택 정렬	n^2	n^2	n^2
삽입 정렬	n^2	n	n^2
퀵 정렬	n^2	$n \log n$	$n \log n$
병합 정렬	$n \log n$	$n \log n$	$n \log n$

연습문제

- 백준 온라인 저지 – 2750 수 정렬하기(정렬 알고리즘 제출용)

추가문제

- 백준 온라인 저지 – 11399 ATM