자료구조와 알고리즘

7강 – 개선된 정렬 알고리즘

LECTURED BY SOONGU HONG

* 개선된 정렬 알고리즘

* 퀵 정렬 (quick sort)

- 퀵 정렬은 정렬 알고리즘 중 가장 우수한 평균 수행속도를 가집니다.
- 퀵 정렬은 분할 알고리즘(partition algorithm)을 기본으로 하며 nlogn의 시간 복잡도를 가집니다.

* 병합 정렬 (merge sort)

- 병합 정렬은 분할 정복(divide and conquer) 방식을 사용해 데이터를 분할하고 분할한 집합을 정렬하며 합치는 알고리즘입니다.
- 퀵 정렬과 마찬가지로 nlogn의 시간 복잡도를 가집니다.

* 기수 정렬 (radix sort)

- 기수 정렬은 값을 비교하지 않는 기존과는 다른 방식의 정렬 기법입니다.
- 기수 정렬은 값을 놓고 비교할 자릿수를 정한 다음 해당 자릿수만 비교합니다.
- 시간 복잡도는 O(kn)으로 가장 빠르며 여기서 k는 데이터의 자릿수를 말합니다.

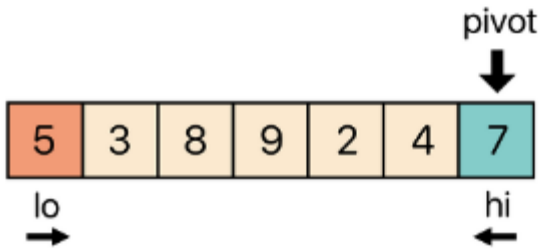
1. 퀵 정렬

* 퀵 정렬 과정

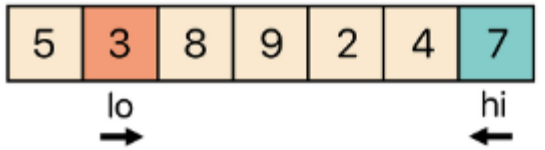
- 퀵 정렬은 pivot값을 중심으로 지속해서 데이터를 2개의 집합으로 분할하면서 정렬하는것이 핵심입니다.

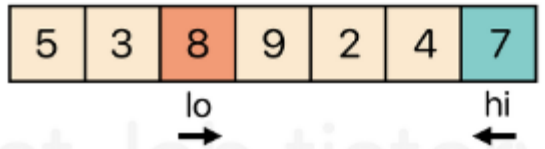
1. 데이터 분할의 기준이 되는 pivot을 설정 (예제에선 가장 오른쪽 값을 먼저 선택)
2. Pivot을 기준으로 다음 a~e과정을 거쳐 데이터를 2개의 집합으로 분리한다.
   2-a. start가 가리키는 데이터가 pivot이 가리키는 데이터보다 작으면 start를 오른쪽으로 한 칸 이동
   2-b. end가 가리키는 데이터가 pivot이 가리키는 데이터보다 크면 end를 왼쪽으로 한 칸 이동
   2-c. start가 가리키는 데이터가 pivot이 가리키는 데이터보다 크고, end가 가리키는 데이터가 pivot이 가리키는 데이터보다 작으면 start와 end의 값을 swap하고 두 포인터 모두 한칸씩 이동
   2-d. start와 end가 크로스 될 때까지 2-a ~ 2-c를 반복
   2-e. start와 end가 크로스되면 pivot과 end의 값을 swap한다.
3. 분리 집합에서 각각 다시 pivot을 선정한다.
4. 분리 집합이 1개 이하가 될 때까지 과정 1 ~ 3을 반복한다.

피벗보다 작은 값들은 왼쪽 부분에,
피벗보다 큰 값들은 오른쪽 부분에 위치하도록 해준다.

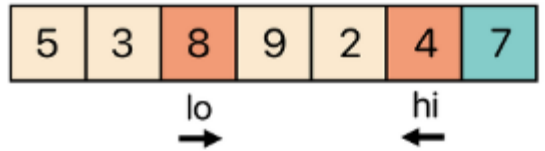


1) lo가 pivot 값보다 작으면서
lo < hi 일 때까지 lo을 증가
(pivot 보다 큰 값이 나올 때 까지)

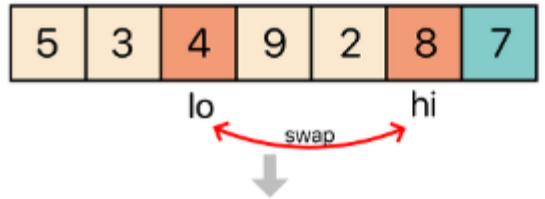




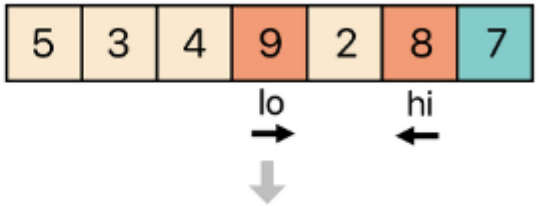
2) hi가 pivot 값보다 크거나 같으면서
lo < hi 일 때 까지 hi 감소
(pivot 보다 작은 값이 나올 때 까지)



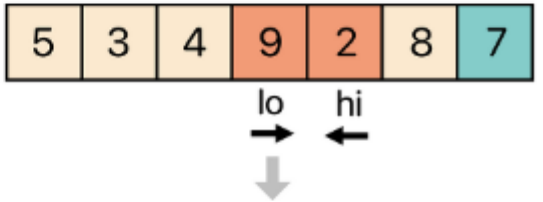
3) 1, 2과정(loop)이 끝나면
두 원소를 swap



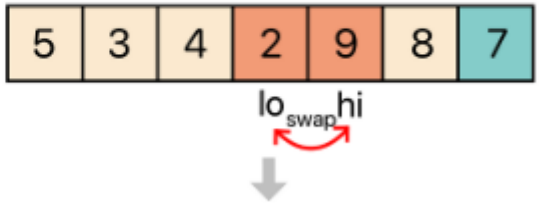
1) lo가 pivot 값보다 작으면서
lo < hi 일 때까지 lo을 증가
(pivot 보다 큰 값이 나올 때 까지)



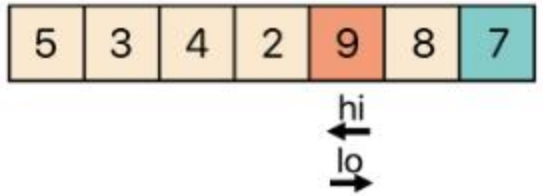
2) hi가 pivot 값보다 크거나 같으면서
lo < hi 일 때 까지 hi 감소
(pivot 보다 작은 값이 나올 때 까지)



3) 1, 2과정(loop)이 끝나면
두 원소를 swap

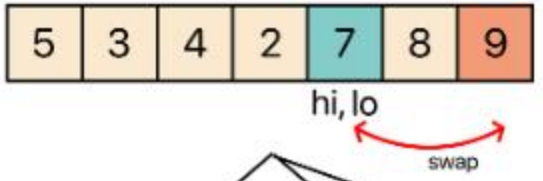


1) lo가 pivot 값보다 작으면서
lo < hi 일 때까지 lo을 증가
(pivot 보다 큰 값이 나올 때 까지)

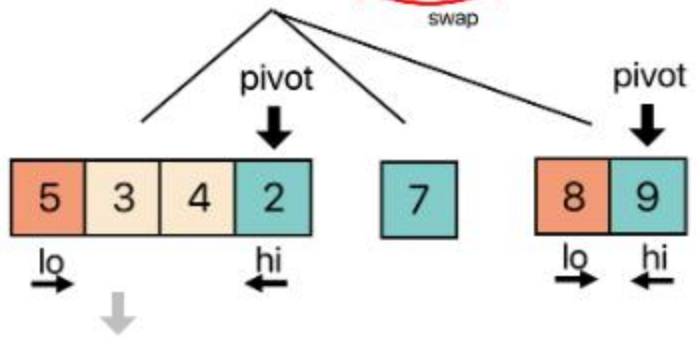


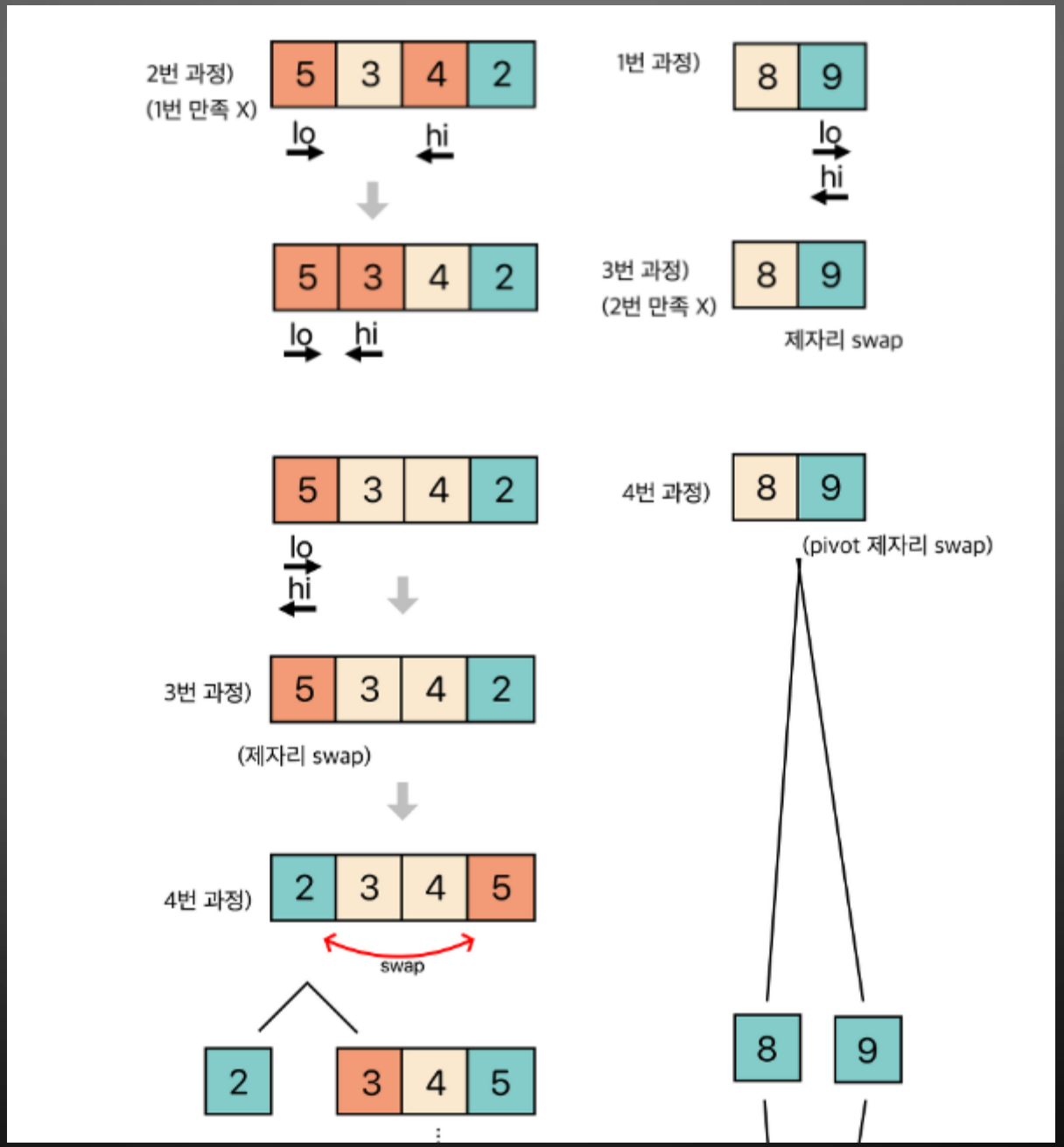
2) lo < hi 을 만족 못하여 2번 과정이
끝나고, lo와 hi가 swap됨(제자리 swap)

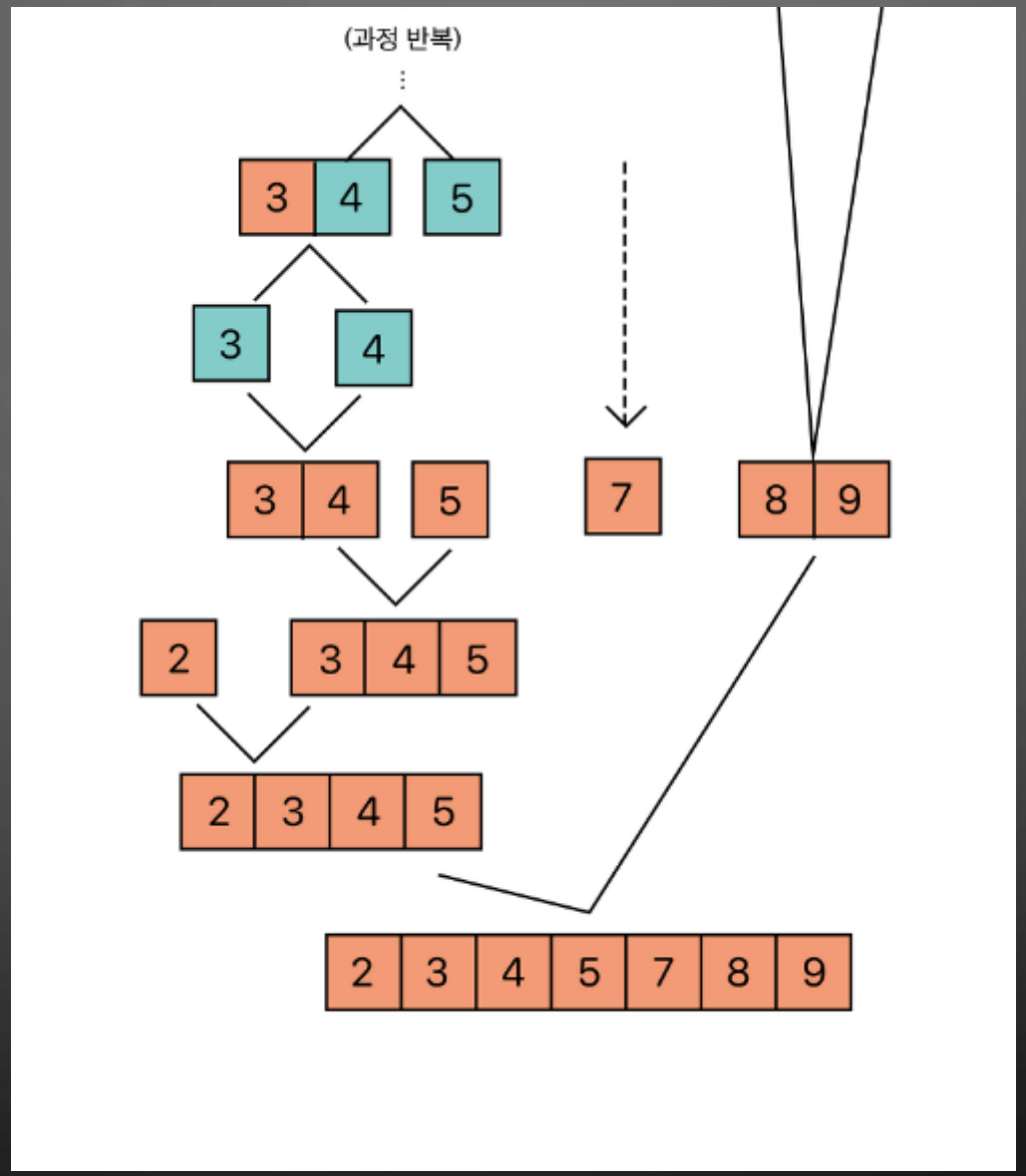
4) lo < hi를 더이상 만족하지 못한다면
pivot과 hi를 교환



5) pivot을 기준으로 왼쪽과 오른쪽
부분리스트로 쪼개서 위 과정을 반복




2번 과정)
(1번 만족 X)
5 3 4 2
lo
hi
5 3 4 2
lo hi
5 3 4 2
lo
hi
3번 과정)
5 3 4 2
(제자리 swap)
4번 과정)
2 3 4 5
swap
2
3 4 5
1번 과정)
8 9
lo
hi
3번 과정)
(2번 만족 X)
8 9
제자리 swap
4번 과정)
8 9
(pivot 제자리 swap)
8
9


(과정 반복)
3
4
5
3
4
7
8
9
3
4
5
2
3
4
5
2
3
4
5
2
3
4
5
7
8
9

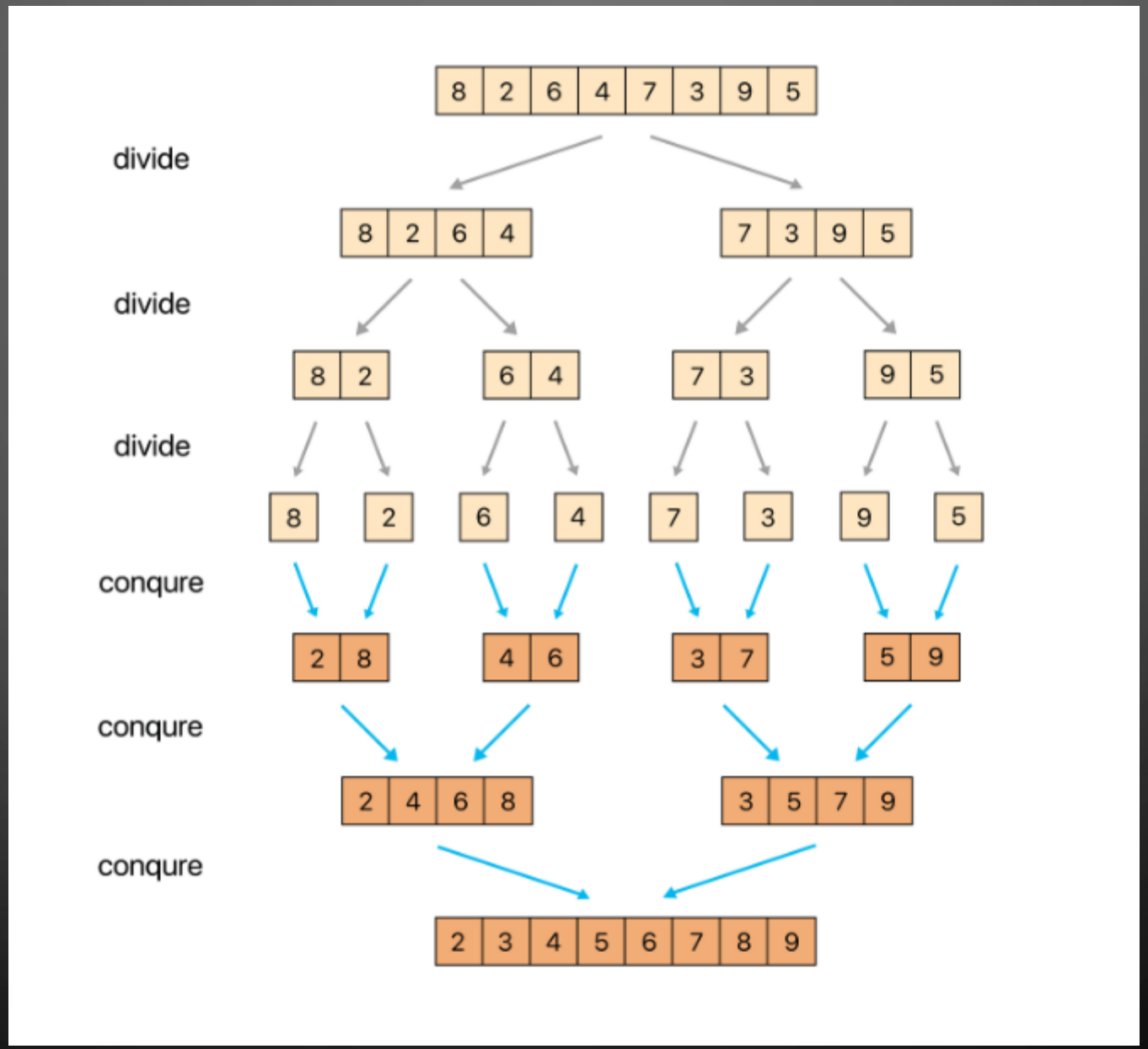
이미지 출처: https://st-lab.tistory.com/250

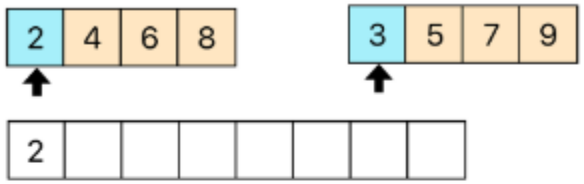
2. 병합 정렬

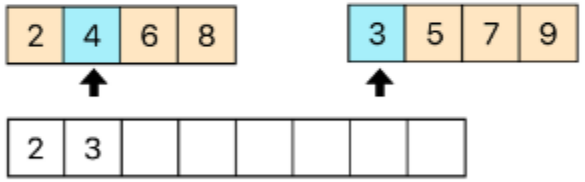
* 병합 정렬 과정

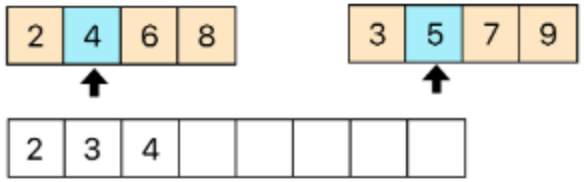
- 병합 정렬은 분할 정복 알고리즘을 사용해 데이터를 분할하고 분할한 집합을 정렬하며 합치는 알고리즘입니다.

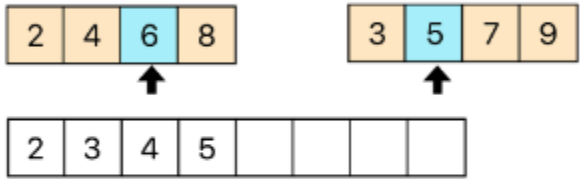
1. 집합을 2개씩의 그룹으로 지속적으로 분할합니다. 예를들면 8개 요소가 있는 배열이면 2개씩 4그룹으로 나누어 오름차 정렬합니다.
2. 투 포인터 알고리즘을 이용하여 왼쪽, 오른쪽 그룹을 병합합니다. 왼쪽 포인터와 오른쪽 포인터의 값을 비교하여 작은 값을 결과 배열에 추가하며 추가한 쪽의 포인터를 이동합니다.

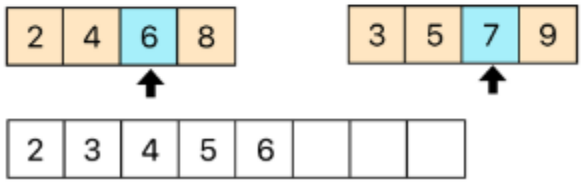

8 2 6 4 7 3 9 5
divide
8 2 6 4
7 3 9 5
divide
8 2
6 4
7 3
9 5
divide
8
2
6
4
7
3
9
5
conqure
2 8
4 6
3 7
5 9
conqure
2 4 6 8
3 5 7 9
conqure
2 3 4 5 6 7 8 9

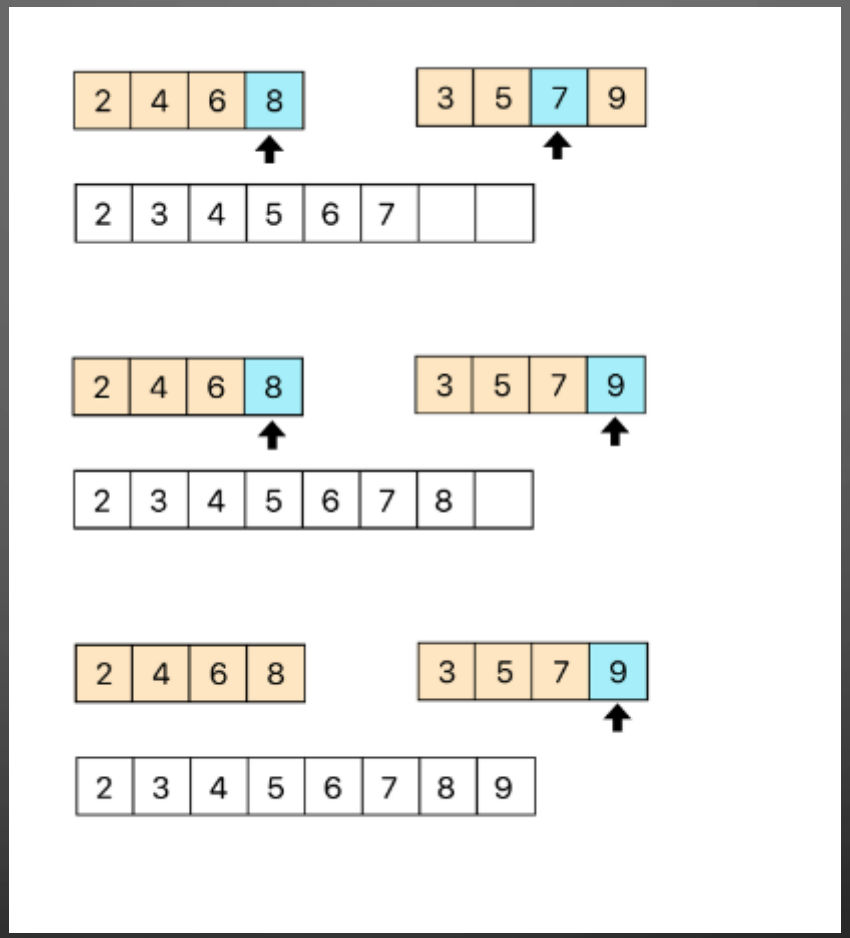

2 4 6 8
3 5 7 9
2


2 4 6 8
3 5 7 9
2 3


2 4 6 8
3 5 7 9
2 3 4


2 4 6 8
3 5 7 9
2 3 4 5


2 4 6 8
3 5 7 9
2 3 4 5 6


2 4 6 8
3 5 7 9
2 3 4 5 6 7
2 4 6 8
3 5 7 9
2 3 4 5 6 7 8
2 4 6 8
3 5 7 9
2 3 4 5 6 7 8 9

이미지 출처: https://st-lab.tistory.com/233?category=892973

3. 기수 정렬

* 기수 정렬 과정

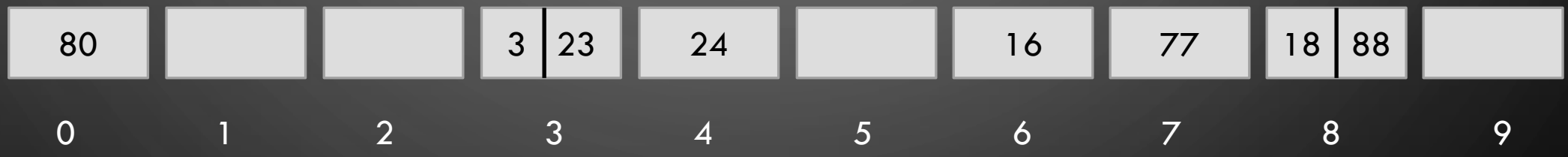
- 기수 정렬은 값을 비교하지 않는 특이한 정렬입니다.
- 기수 정렬은 1의 자리부터 높은 자리수까지 자리수만 지속적으로 비교합니다.

1. 기수 정렬은 10개의 큐를 이용합니다. 각 큐는 값의 자리수를 대표합니다.
2. 처음에는 1의 자리수를 기준으로 순서대로 각 큐에 요소들을 집어넣습니다.
3. 그런 다음 큐의 순서대로 poll한 다음에 다시 한번 10의 자리수를 기준으로 큐에 집어넣습니다.
4. 최고 자리수까지 이와 같은 행동을 반복하면 정렬이 완료됩니다.

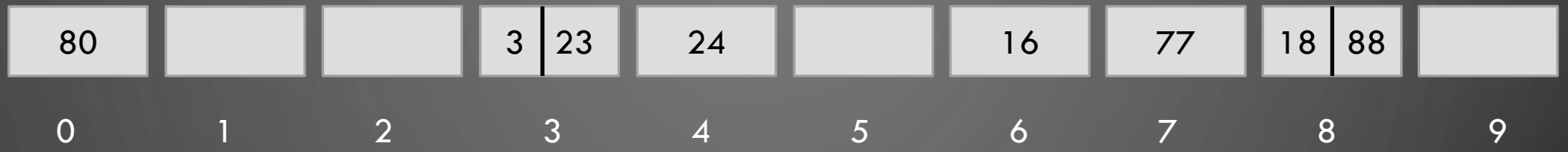
대상 데이터 16 80 18 77 3 24 88 23

1의 자리를 기준으로 데이터 저장

자리수별 10개의 큐를 준비하고 넣기

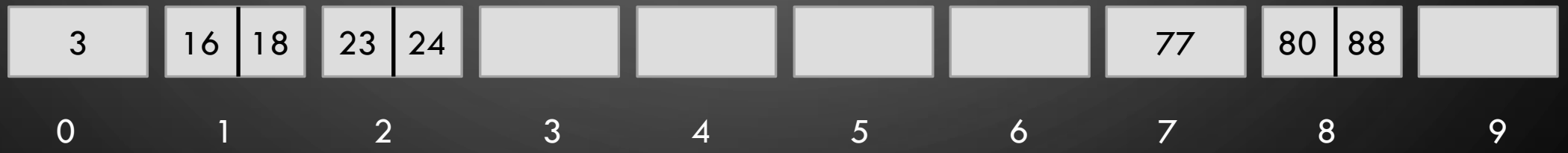


대상 데이터 16 80 18 77 3 24 88 23

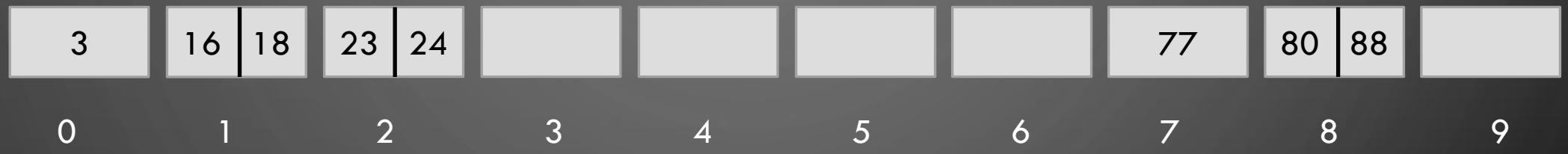


큐 순서대로 poll 80 3 23 24 16 77 18 88

10의 자리를 기준으로 다시 큐에 데이터 저장



대상 데이터 16 80 18 77 3 24 88 23



큐 순서대로 poll 3 16 18 23 24 77 80 88 -> 정렬 완료