20220802

작성일시	@2022년 8월 3일 오전 10:24
강의 번호	
유형	
자료	
복습	☐
속성	

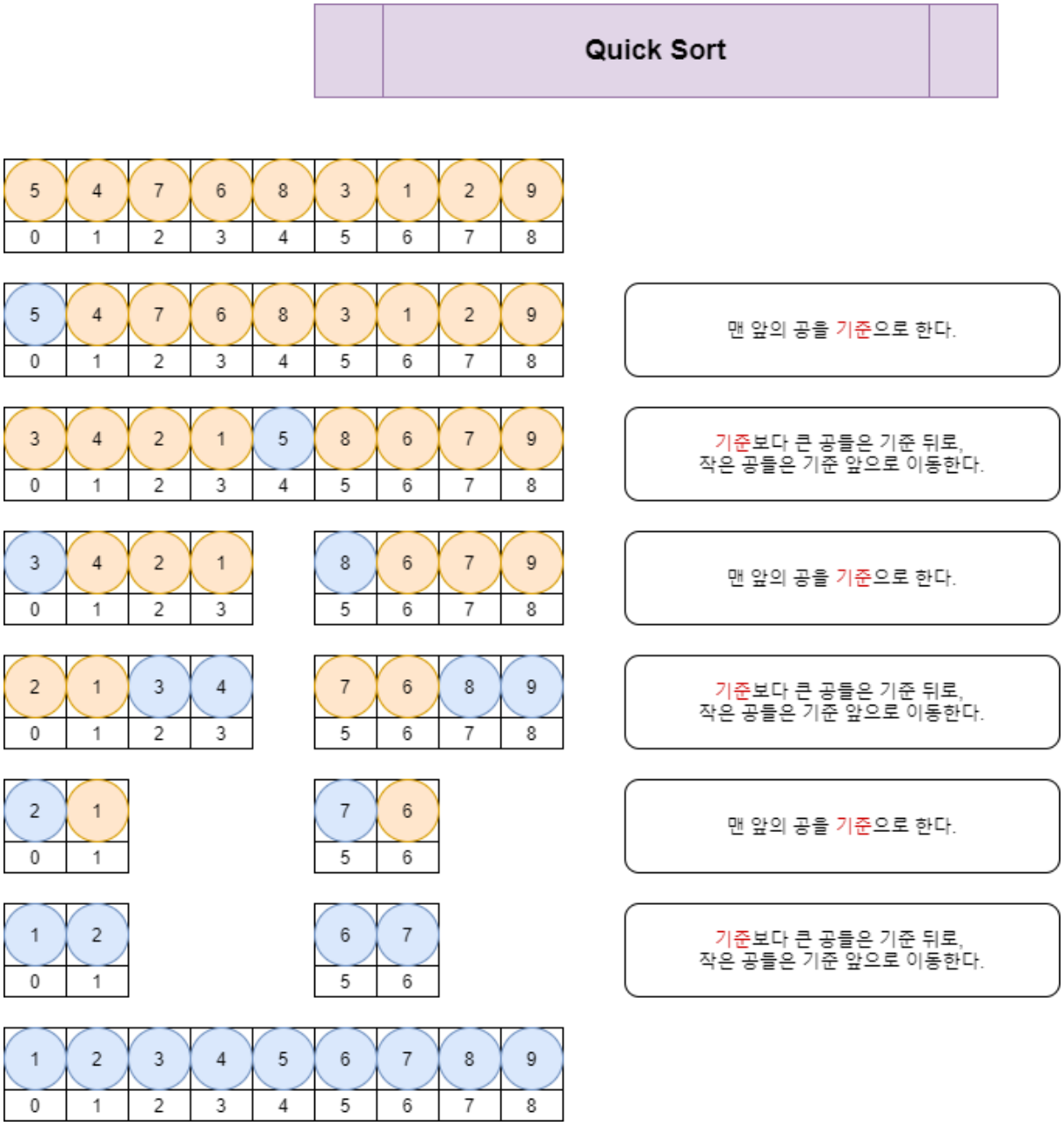
Quick Sort

데이터를 대소그룹 둘로 나누어 분해한 후에 전체를 최종적으로 정렬하는 방식의 알고리즘이다.

Divide and conquer(분할 정복법)

퀵 정렬은 대량의 데이터를 정렬할 때 매우 자주 사용된다. 유명한 알고리즘 중에서도 실제로 많이 사용되는 빈도가 가장 높고 중요한 알고리즘이다.

퀵 정렬은 '기준값을 선택한 후 그보다 작은 데이터 그룹과 큰 데이터 그룹으로 나눈다.'라는 처리를 반복 수행하여 데이터를 정렬하게 된다.



퀵 정렬의 알고리즘

퀵 정렬은 크게 2개의 처리로 구성된다.

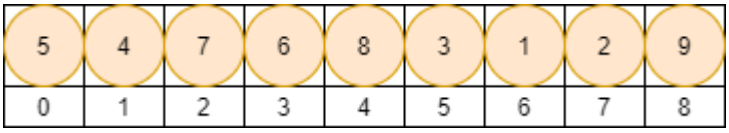
1. 기준값을 경계로 데이터를 대소로 나누는 처리
2. 나눈 데이터에 대해 반복적으로 똑같은 작업을 실행

1. 기준값을 경계로 데이터를 대소로 나누는 처리
    - 퀵 정렬의 핵심은 데이터를 대소로 나누는 처리이다.
    - 배열의 왼쪽과 오른쪽부터 각각 변수를 움직여 대소로 정렬하자.

    기준값보다 작은 공을 기준값의 앞으로 이동시키고 기준값보다 큰 공은 뒤로 이동시키는 것이 바로 퀵 정렬의 초석이 되는 처리이다.

    배열 설정 : 먼저 배열을 준비하자. 정수형 배열로 이름은 arr로 요소수는 9개로 정한다. 따라서 첨자는 0부터 8까지 사용된다.

    arr



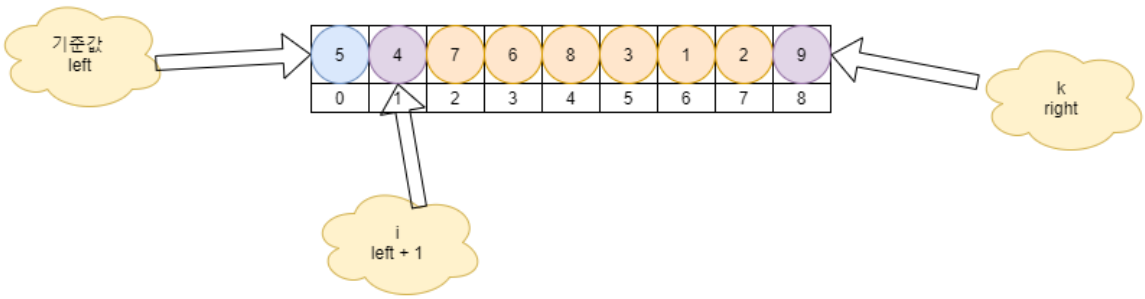
변수 설정

변수는 5개를 준비한다.

1. left - 정렬 범위에서 맨 앞 요소에 첨자를 넣는 변수
2. right - 정렬 범위에서 맨 끝 요소에 첨자를 넣는 변수
3. i - 기준값보다 큰 요소를 찾기 위한 변수
4. k - 기준값보다 작은 요소를 찾기 위한 변수
5. w - 데이터 교환용 임시 변수 temp

이 다섯개의 변수를 사용하여 우선 left와 right에 각각 정렬 범위 맨 앞 요소의 첨자와 마지막 요소의 첨자를 대입한다. 따라서 이번에는 (처음에는) left는 0, right은 8이 된다. 기준은 맨 앞 요소로 하기 때문에 arr[left]이 된다.

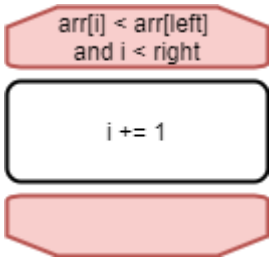
그리고 i에 left의 하나 오른쪽 left + 1로 정하고 k에는 right을 대입한다.



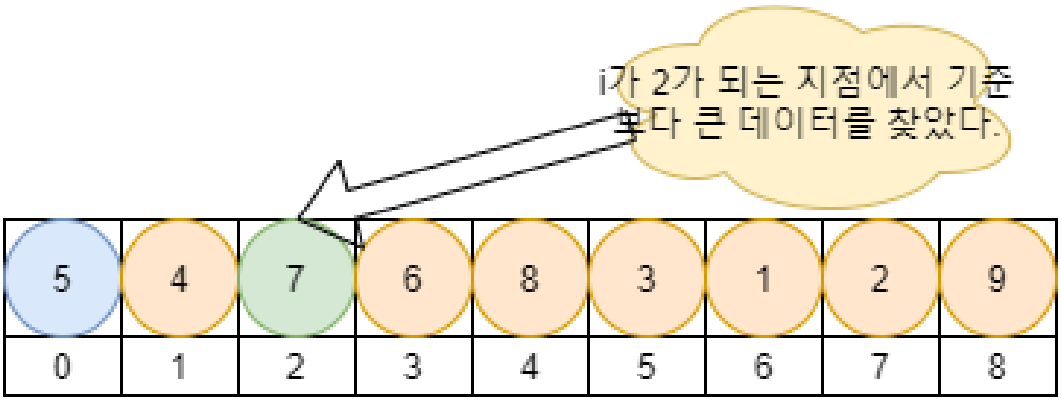
- 변수 i를 사용하여 기준값보다 큰 요소 찾기

i는 '기준값보다 큰 요소를 찾는 변수'이다. 현재 위치에서 하나씩 오른쪽으로 이동하면서 기준값보다 큰 요소가 있는지 확인하고 발견되면 그곳에서 멈춘다.

```
arr[i] > arr[left]      // arr[left]는 기준을 표시
```



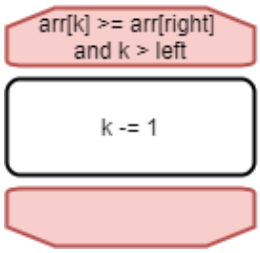
2가지 조건, 기준값 arr[left]보다 큰 값을 찾고 오른쪽 끝까지 찾지 못할 때까지 반복한다.

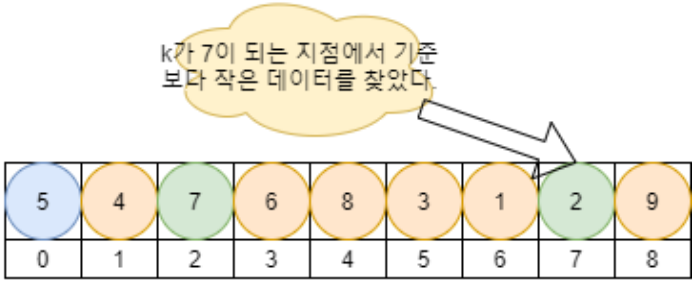


기준값보다 큰 요소를 발견했기 때문에 i는 일단 여기에서 멈춘다. 그리고 반대쪽 변수 k, 즉 작은 값 찾기로 넘어간다.

- 변수 k를 사용하여 기준값보다 작은 요소 찾기

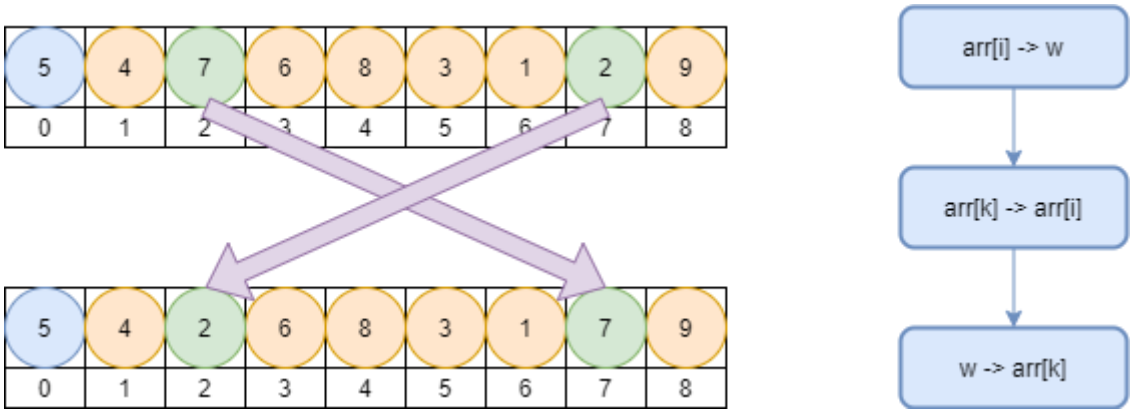
k는 '기준값보다 작은 요소를 찾는 변수'이다. 현재 위치에서 하나씩 왼쪽으로 이동하면서 기준값보다 작은 요소가 있는지 확인하고 발견되면 그곳에서 멈춘다.





기준값보다 작은 요소를 발견했기 때문에 k도 일단 여기에서 멈춘다.

- 큰 데이터와 작은 데이터 교환하기



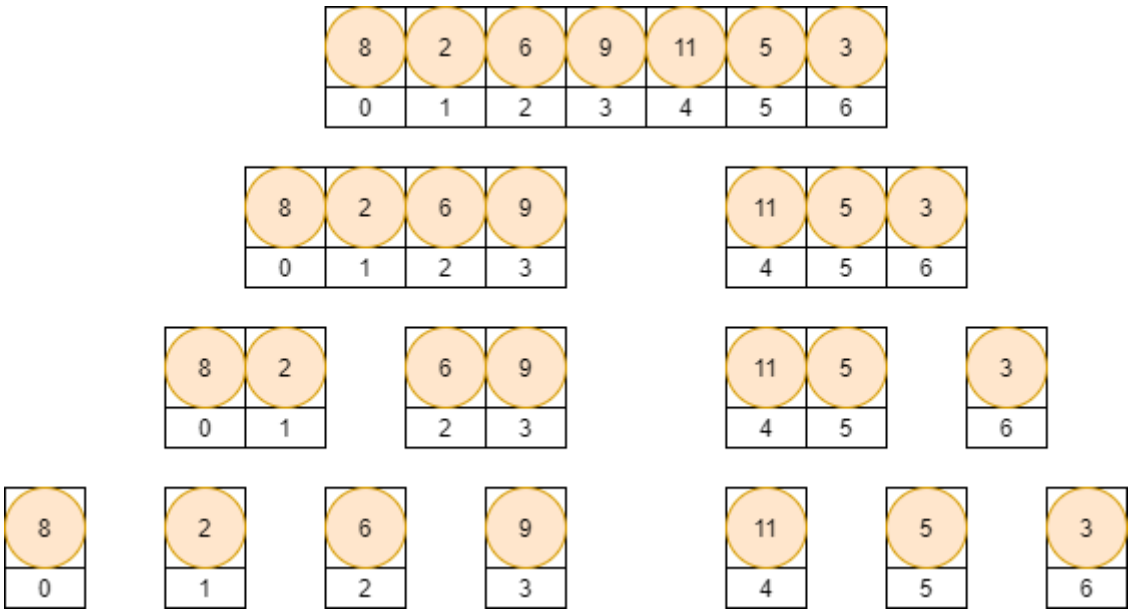

```
package quickSort;

import java.util.Arrays;

public class QuickSort {

  public static void main(String[] args) {
    // TODO Auto-generated method stub
    int[] arr = {5,4,7,6,8,3,1,2,9};
    arr = quickSort(arr, 0, arr.length-1);  // 어레이, 시작, 끝
    System.out.println(Arrays.toString(arr));
  }

  static int[] quickSort(int[] arr, int start, int end) {
    int p = partition(arr, start, end);

    if(start < p-1) {
      quickSort(arr, start, p-1);
    }
    if(p < end) {
      quickSort(arr, p, end);
    }
    return arr;
  }

  static int partition(int[] arr, int start, int end) {
    int pivot = arr[(start + end) /2];
    while(start <= end) {
      while(arr[start] < pivot) start++;
      while(arr[end] > pivot) end--;
      if(start <= end) {
        int tmp = arr[start];
        arr[start] = arr[end];
        arr[end] = tmp;
        start++;
        end--;
      }
```

```
    }
    return start;
  }

}

* 결과값은 [1, 2, 3, 4, 5, 6, 7, 8, 9] *
```

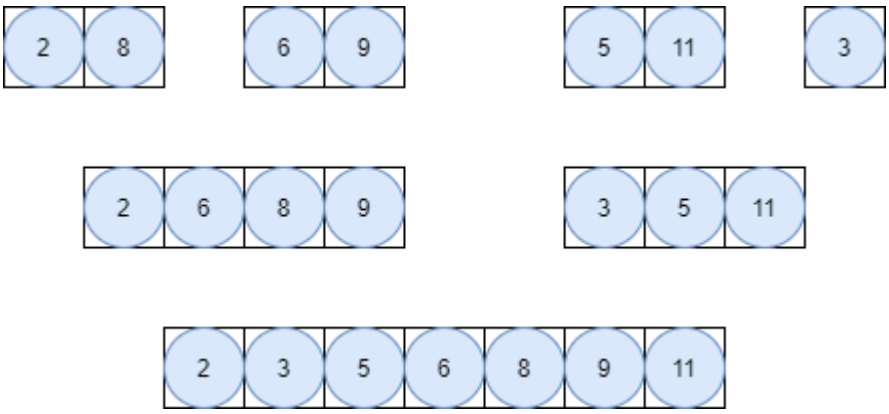
Merge Sort 병합 정렬

분할 정복 (Divide and Conquer)을 사용하는 방법이다. 분할 정복은 주어진 문제를 해결하기 쉬운 단계까지 분할한 후에 분할된 문제를 해결하고 그 결과를 다시 결합하는 알고리즘이다.

- 퀵 정렬이 제일 빠르고, 그 다음이 병합 정렬이 2번째로 빠르다.



먼저 데이터들을 정렬을 생각하지 않고 1개씩 될 때까지 나눈다. 나누는 방법은 배열을 반으로 쪼개고 그 쪼개진 배열을 또 다시 반으로 쪼개고 이 과정을 반복한다. 위에서 7개의 데이터를 분할하기 위해 이 과정을 3번 거쳤다.



분할이 완료된 후에는 데이터를 비교하면서 결합한다. 제일 앞에 있는 2와 8을 다시 합치는 데

작은 수 2가 앞으로 큰 수 8이 뒤로 보낸 상태로 결합한다. 이 과정을 각각 2개씩 반복하여 합치고

그 합친 2개를 다시 합친다. 이때 각각의 그룹에서 먼저 제일 첫번째 값을 비교하여 작은 값을 추출한다. 그 다음 다시 한번 각각의 그룹의 제일 왼쪽 즉 작은 값을 또 다시 비교하여 둘 중 작은 값을 다시 추출한다. 이 과정을 반복하여 전체 정렬을 마치게 된다.