A10. 퀵 정렬

알고리즘

- 원광대학교 컴퓨터소프트웨어공학과
- 2019학년도 2학기 화6수78
- 알고리즘 / 374015-01

목차

01. 퀵 정렬의 개념을 파악하자

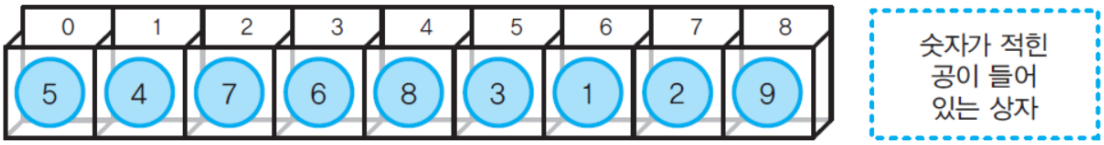
- Point
  - 데이터를 정렬하는 정렬 알고리즘 중 하나다,
  - 데이터를 대소 그룹의 둘로 나누어 분해한 후 전체를 정렬하는 방식의 알고리즘이다.
  - 실행 속도가 빠른 것이 특징이다.

01. 퀵 정렬의 개념을 파악하자

- 퀵 정렬의 개념
  - 처리 속도가 빠른 정렬 알고리즘
  - 대량의 데이터를 정렬할 때 자주 사용
  - '기준값을 선택한 후 그보다 작은 데이터 그룹과 큰 데이터 그룹으로 나눈다.'라는 처리를 반복 수행

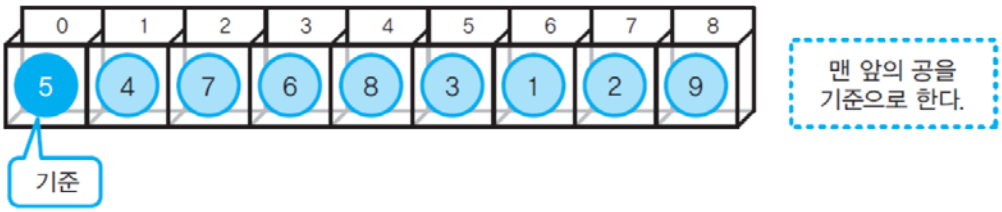
01. 퀵 정렬의 개념을 파악하자

- 퀵 정렬로 공을 오름차순으로 정렬해 보자



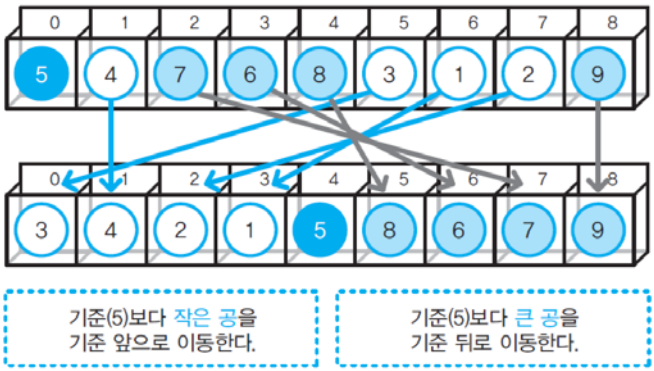
01. 퀵 정렬의 개념을 파악하자

- 맨 앞의 5를 기준으로 공을 대소로 나누기
  - 정렬 범위의 맨 앞에 있는 공을 기준으로 선택하는 방법을 사용한다



01. 퀵 정렬의 개념을 파악하자

- 맨 앞의 5를 기준으로 공을 대소로 나누기
  - 5보다 작은 공을 5의 앞으로 이동하고, 5보다 큰 공은 5의 뒤로 이동한다.
  - 어떤 절차로 이동할 것인지는 나중에 자세히 살펴보기로 한다.

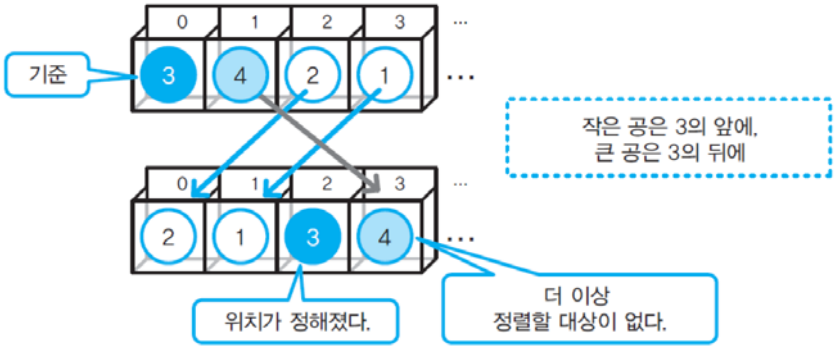


01. 퀵 정렬의 개념을 파악하자

- 맨 앞의 5를 기준으로 공을 대소로 나누기
  - 5보다 작은 공은 모두 5보다 앞에 있고, 5보다 큰 공은 5보다 뒤에 있다.
  - 앞으로 5번 공은 더 이상 건드리지 않아도 된다는 뜻이다.

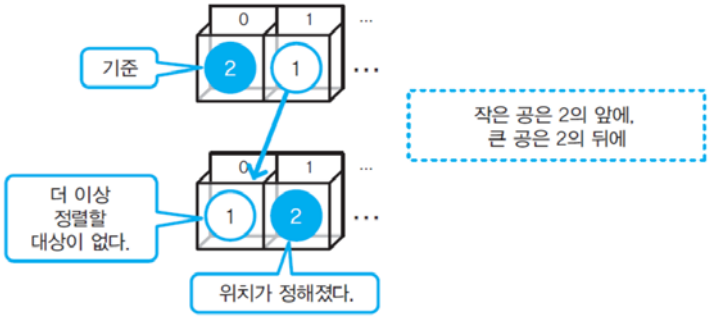
01. 퀵 정렬의 개념을 파악하자

- 맨 앞의 3을 기준으로 공을 대소로 나누기
  - 3보다 작은 공을 3의 앞으로 이동하고, 3보다 큰 공을 3의 뒤로 이동한다.
  - 앞으로 3번 공은 더 이상 건드리지 않아도 된다는 뜻이다.



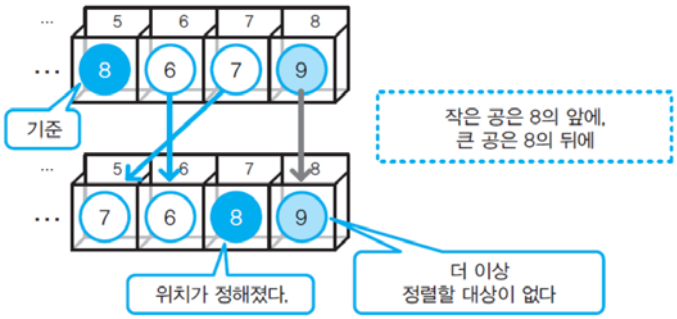
01. 퀵 정렬의 개념을 파악하자

- 맨 앞의 2를 기준으로 공을 대소로 나누기
  - 2보다 작은 공을 2의 앞으로 이동하고, 2보다 큰 공을 2의 뒤로 이동한다.
  - 앞으로 2번 공은 더 이상 건드리지 않아도 된다는 뜻이다.



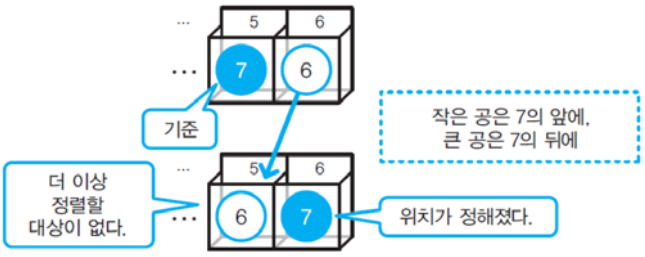
01. 퀵 정렬의 개념을 파악하자

- 맨 앞에 있는 8을 기준으로 공을 대소로 나누기
  - 8보다 작은 공을 8의 앞으로 이동하고, 8보다 큰 공을 8의 뒤로 이동한다.
  - 앞으로 8번 공은 더 이상 건드리지 않아도 된다는 뜻이다.



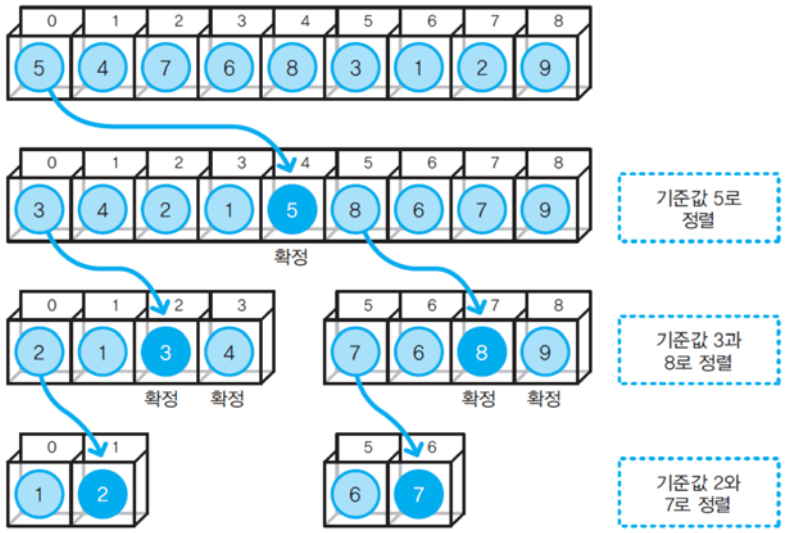
01. 퀵 정렬의 개념을 파악하자

- 앞의 7을 기준으로 공을 대소로 나누기
  - 7보다 작은 공을 7보다 앞으로 이동하고, 7보다 큰 공을 7보다 뒤로 이동한다.
  - 앞으로 7번 공은 더 이상 건드리지 않아도 된다는 뜻이다.



01. 퀵 정렬의 개념을 파악하자

- 퀵 정렬의 개념
  - '기준값을 선택한 후 그보다 작은 공 그룹과 큰 공 그룹으로 나눈다.'라는 과정을 반복하여 모든 공을 정렬

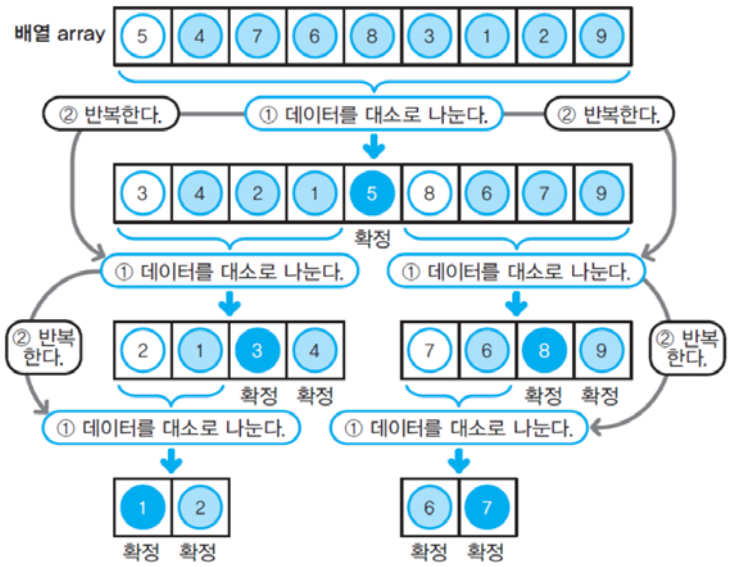


02. 퀵 정렬의 알고리즘

- Point
  - 퀵 정렬은 크게 2개의 처리로 구성되어 있다.

02. 퀵 정렬의 알고리즘

- 퀵 정렬의 알고리즘
  - ① 기준값을 경계로 데이터를 대소로 나누는 처리
  - ② 나눈 데이터에 대해 반복적으로 똑같은 작업 실행하기

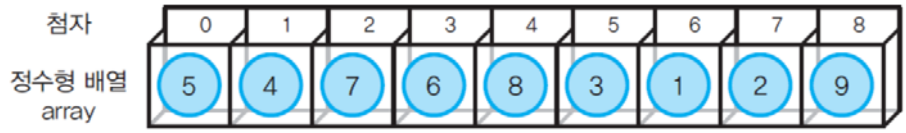


03. 기준값을 경계로 데이터를 대소로 나누는 처리

- Point
  - 퀵 정렬의 핵심은 데이터를 대소로 나누는 처리다.
  - 배열의 왼쪽과 오른쪽부터 각각 변수를 움직여 대소로 정렬한다.

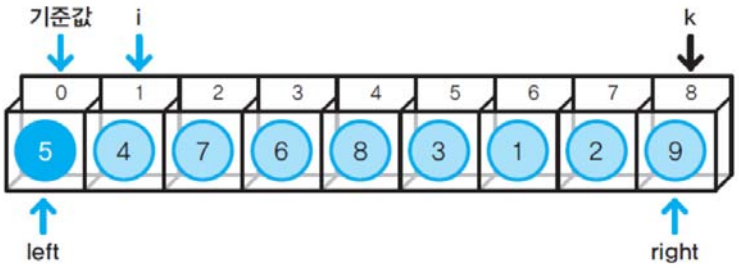
03. 기준값을 경계로 데이터를 대소로 나누는 처리

- 배열의 설정



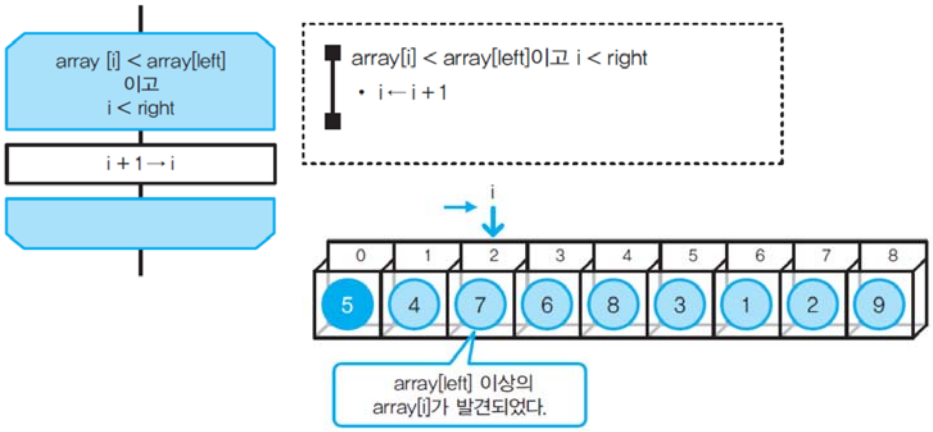
03. 기준값을 경계로 데이터를 대소로 나누는 처리

- 변수의 설정
  - left: 정렬 범위에 있어 맨 앞 요소에 첨자를 넣는 변수
  - right: 정렬 범위의 맨 끝 요소에 첨자를 넣는 변수
  - i: 기준값보다 큰 요소를 찾기 위한 변수
  - k: 기준값보다 작은 요소를 찾기 위한 변수
  - w: 데이터 교환용 변수



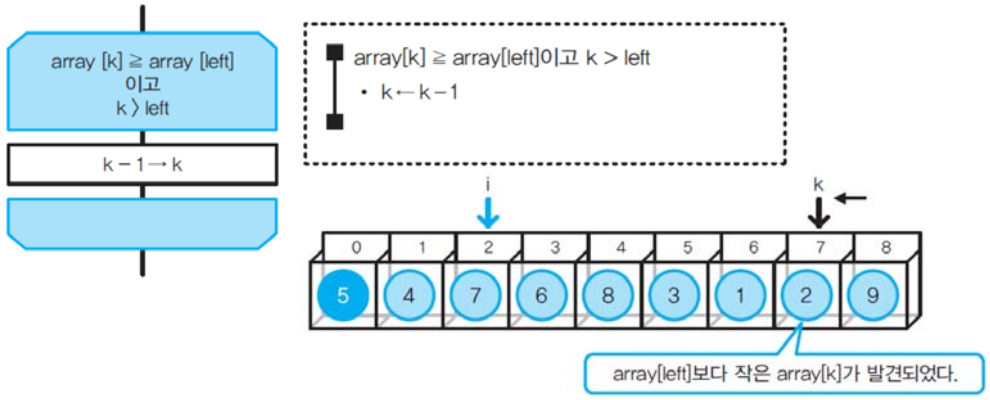
03. 기준값을 경계로 데이터를 대소로 나누는 처리

- 변수 i를 사용하여 기준값보다 큰 요소를 찾기
  - 현재 위치에서 하나씩 오른쪽으로 이동하면서 기준값보다 큰 요소가 발견되면 그곳에서 멈춘다.



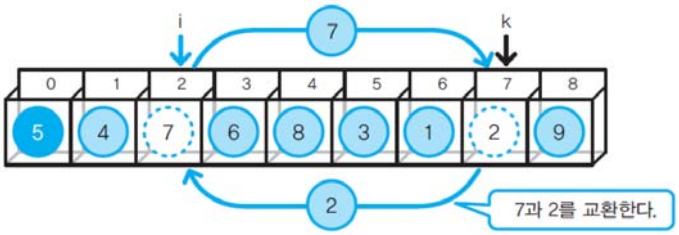
03. 기준값을 경계로 데이터를 대소로 나누는 처리

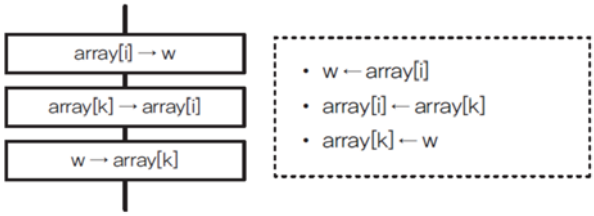
- 변수 k를 사용하여 기준값보다 작은 요소 찾기
  - 현재 위치에서 하나씩 왼쪽으로 이동하면서 기준값보다 작은 요소가 발견되면 그곳에서 멈춘다.



03. 기준값을 경계로 데이터를 대소로 나누는 처리

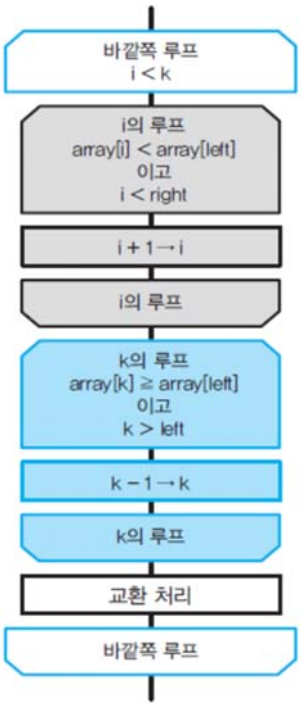
- 큰 데이터와 작은 데이터 교환하기
  - array[i]는 기준값보다 큰 데이터가 들어 있고,
  - array[k]는 기준값보다 작은 데이터가 들어 있다.
  - 두 가지 데이터를 교환하여 위치를 바꾼다.
  - i와 k의 위치는 그대로다.





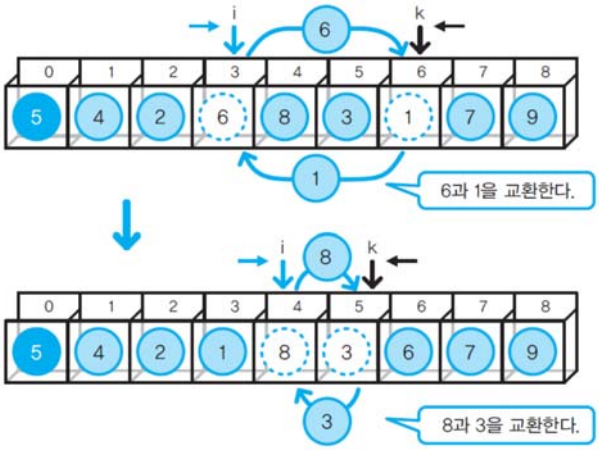
03. 기준값을 경계로 데이터를 대소로 나누는 처리

- 데이터를 찾아 교환하는 처리 반복하기
  - 지금까지의 처리를 i와 k가 엇갈릴 때까지 반복하여 교환을 진행한다.
  - 반복을 계속하는 조건은 i가 k보다 작을 동안(i < k)이다.



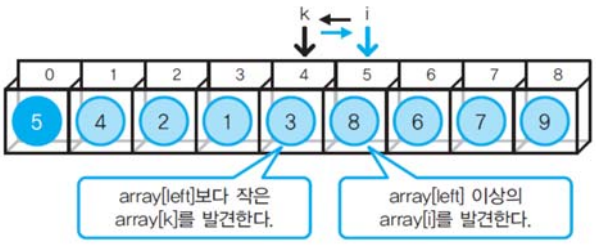
03. 기준값을 경계로 데이터를 대소로 나누는 처리

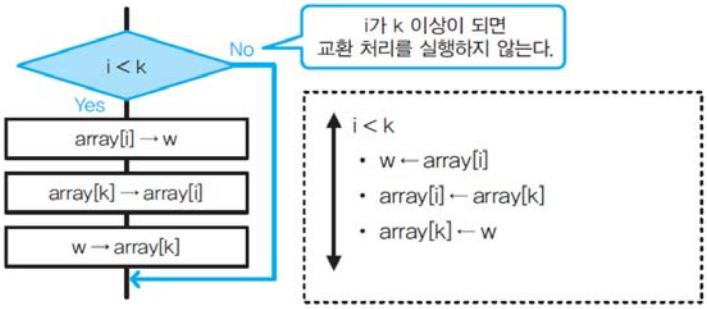
- 데이터를 찾아 교환하는 처리 반복하기



03. 기준값을 경계로 데이터를 대소로 나누는 처리

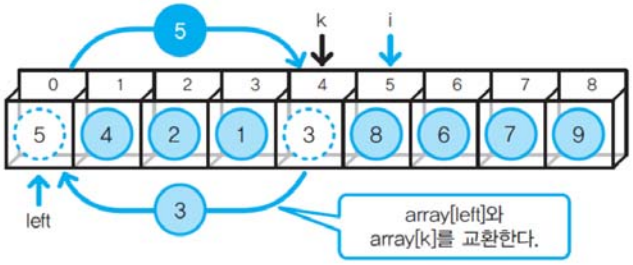
- 데이터를 찾아 교환하는 처리 반복하기
  - 조건식 i< k 동안만, 교환 처리가 실행된다.





03. 기준값을 경계로 데이터를 대소로 나누는 처리

- 기준값을 작은 데이터와 큰 데이터의 중앙으로 이동한다
  - i가 k보다 커진 시점에서
    - 기준값보다 작은 데이터(4, 2, 1, 3)는 왼쪽으로, 큰 데이터(8, 6, 7, 9)는 오른쪽으로 모인 상태다.
  - 앞으로 할 일은 기준값의 데이터를 이 두 그룹의 중앙으로 가져가는 것이다.
    - 이를 위해서는 기준값 데이터를 작은 그룹의 가장 오른쪽 데이터와 교환하면 될 것이다.
    - 기준값은 첫 번째 요소이므로 array[left]와 array[k]를 교환한다.



03. 기준값을 경계로 데이터를 대소로 나누는 처리

- 기준값을 작은 데이터와 큰 데이터의 중앙으로 이동한다

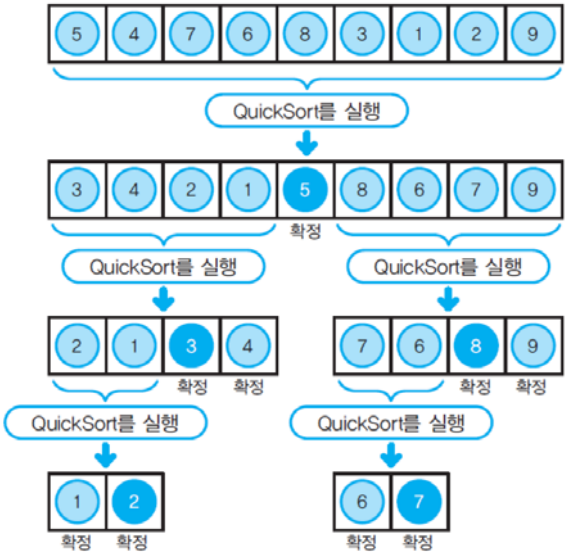
```
○ 정수형 : array[9] = {5, 4, 7, 6, 8, 3, 1, 2, 9}
○ 정수형 : left, right, i, k, w
• left ← 0
• right ← 8
• i ← left + 1
• k ← right
■ i < k
  ■ array[i] < array[left] 또는 i < right
    • i ← i + 1
  ■ array[k] ≥ array[left] 또는 k > left
    • k ← k − 1
  ▲ i < k
    • w ← array[i]
    • array[i] ← array[k]
    • array[k] ← w
• w ← array[left]
• array[left] ← array[k]
• array[k] ← w
```

04. 나눈 데이터에 다시 한 번 같은 처리를 실행하는 처리

- Point
  - '기준값을 경계로 데이터를 대소로 나누는 처리'를 보조 프로그램으로 한다.
  - 보조 프로그램 안에서 재차 보조 프로그램을 사용함으로써 반복 처리를 실행한다.

04. 나눈 데이터에 다시 한 번 같은 처리를 실행하는 처리

- 나눈 데이터에 다시 한 번 같은 처리를 실행하는 처리
  - 기준값을 경계로 데이터를 대소로 나누는 처리를 'QuickSort'라고 불린다.
  - 모든 데이터에 대해 QuickSort를 실행한 후,
    - 작은 데이터 그룹과 큰 데이터 그룹에 대해서도 똑같이 실행한다.



04. 나눈 데이터에 다시 한 번 같은 처리를 실행하는 처리

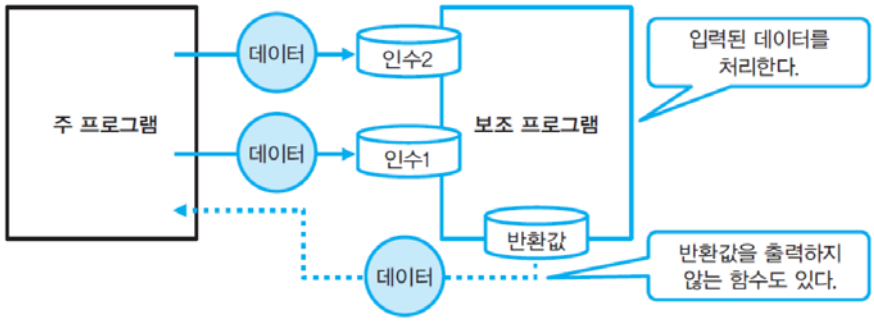
- QuickSort를 보조 프로그램으로 한다

함수와 보조 프로그램

함수는 해시 탐색법에서도 알아본 적이 있다. 앞에서는 '함수는 어떤 값에 대해 그에 대응한 값이 계산되는 계산식이다.'라고 설명했는데, 사실 함수에는 좀 더 넓은 의미가 있다. 프로그래밍 용어에서는 프로그램에서 사용할 수 있는 작은 프로그램을 '함수'라고 한다. 의사 언어에서는 함수를 '보조 프로그램'이라고도 하며, 보조 프로그램을 사용하는 프로그램을 '주 프로그램'이라고 한다. 여기서 보조 프로그램은 '서브 루틴'이라고도 한다.

04. 나눈 데이터에 다시 한 번 같은 처리를 실행하는 처리

- QuickSort를 보조 프로그램으로 한다



04. 나눈 데이터에 다시 한 번 같은 처리를 실행하는 처리

- QuickSort를 보조 프로그램으로 한다
  - 보조 프로그램으로 정의하려면?
    - 그 처리에 보조 프로그램의 이름을 붙인 후 인수(및 반환값)를 설정하면 된다.

```
프로그램: QuickSort(인수1, 인수2, ... )
```


04. 나눈 데이터에 다시 한 번 같은 처리를 실행하는 처리

- QuickSort를 보조 프로그램으로 한다
  - 보조 프로그램으로 정의하려면?
    - 보조 프로그램을 다시 보조 프로그램 안에 기술하여 사용함에 따라
      - 배열 array의 선언과 초기화를 보조 프로그램의 외부에서 실시하는 형태로 수정했다.
      - 보조 프로그램을 호출할 때마다 배열이 선언되고, 초기화되는 불필요한 낭비를 줄일 수 있다.
    - 전역 선언
      - 보조 프로그램이 공통적으로 사용하는 변수와 배열을 프로그램의 외부에 선언하는 방법

04. 나눈 데이터에 다시 한 번 같은 처리를 실행하는 처리

- QuickSort를 보조 프로그램으로 한다
  - 보조 프로그램으로 정의하려면?

```
○ 정수형 : array[9] = {5, 4, 7, 6, 8, 3, 1, 2, 9}
○ 정수형 : left, right, i, k, w
• left ← 0
• right ← 8
• i ← left + 1
• k ← right
■ i < k
  ■ array[i] < array[left] 또는 i < right
    • i ← i + 1
  ■ array[k] ≥ array[left] 또는 k > left
    • k ← k − 1
  ▲▼ i < k
    • w ← array[i]
    • array[i] ← array[k]
    • array[k] ← w
• w ← array[left]
• array[left] ← array[k]
• array[k] ← w
```

```
○ 전역 :   정수형 : array[9] = {5, 4, 7, 6, 8, 3, 1, 2, 9};      ← 전역 선언
○ 보조 프로그램 : QuickSort(정수형 : array[], 정수형 : left, 정수형 : right)
○ 정수형 : left, right, i, k, w                                   ← 선언부
                                  ← 인수를 설정했다.
• i ← left + 1                    ← 기준값을 경계로 데이터를 대소로 나누는 처리
• k ← right

■ i < k
  ■ array[i] < array[left] 또는 i < right
    • i ← i + 1
  ■ array[k] ≥ array[left] 또는 k > left
    • k ← k − 1
  ▲▼ i < k
    • w ← array[i]
    • array[i] ← array[k]
    • array[k] ← w
• w ← array[left]
• array[left] ← array[k]
• array[k] ← w
```


04. 나눈 데이터에 다시 한 번 같은 처리를 실행하는 처리

- QuickSort를 보조 프로그램으로 한다
  - 보조 프로그램을 사용하려면?
    - 예를 들어,
      - QuickSort라는 보조 프로그램에게
      - array라는 배열의 첨자 0에서 8까지의 요소에 대해 작업을 실행해 주길 원할 때는
      - 다음과 같이 작성한다.

```
• QuickSort(array, 0, 8)
```

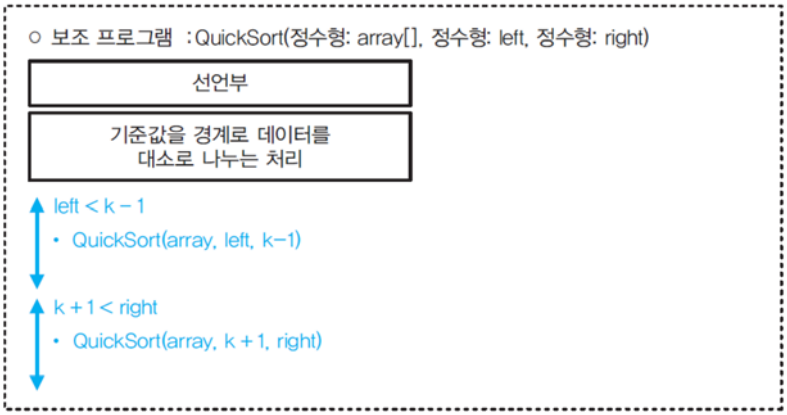
처리하고자 하는 배열명

처리하고자 하는 범위의 맨 앞 요소의 첨자

처리하고자 하는 범위의 맨 끝 요소의 첨자

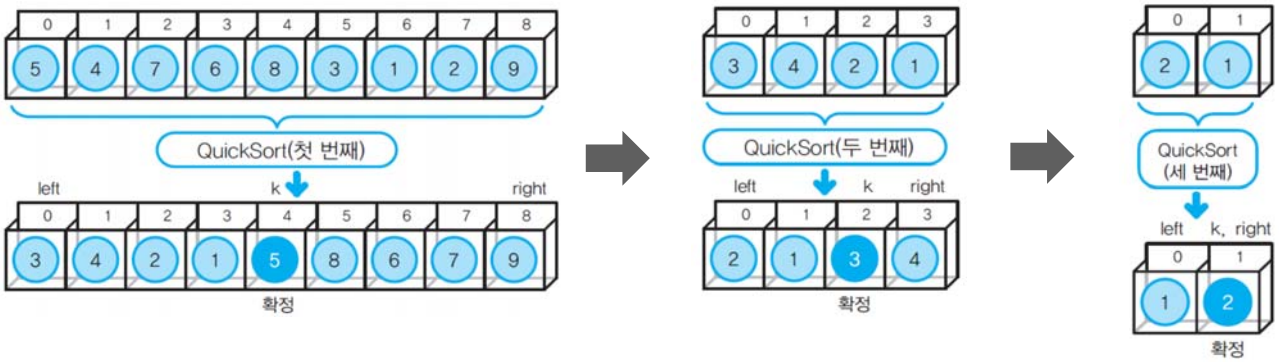
04. 나눈 데이터에 다시 한 번 같은 처리를 실행하는 처리

- QuickSort 안에서 QuickSort 사용하기
  - QuickSort를 보조 프로그램으로 만들면 편리하게 활용할 수 있다.
  - QuickSort 안에서 QuickSort를 사용할 수도 있다.
    - 보조 프로그램은 보조 프로그램 안에 기술하여 사용할 수 있으며,
    - 별개의 보조 프로그램이 아니라 자기 자신을 사용할 수도 있다.



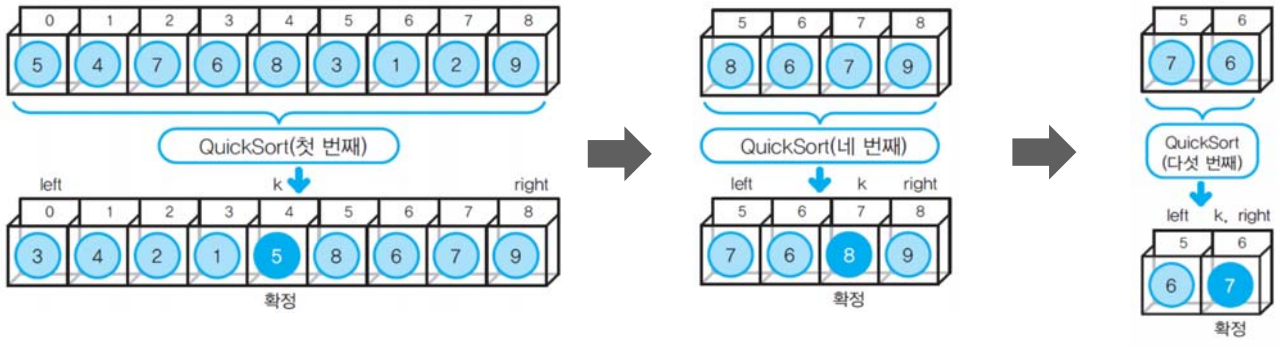
04. 나눈 데이터에 다시 한 번 같은 처리를 실행하는 처리

- QuickSort 안에서 QuickSort 사용하기
  - 첫 번째 QuickSort
    - 5의 위치 확정(left = 0, k = 4, right = 8)
  - 두 번째 QuickSort
    - (left<k-1 경우) QuickSort(array,left,k-1) = QuickSort(array,0,4-1) = QuickSort(array,0,3)
    - 3의 위치 확정(left = 0, k = 2, right = 3)
  - 세 번째 QuickSort
    - (left<k-1 경우) QuickSort(array,left,k-1) = QuickSort(array,0,2-1) = QuickSort(array,0,1)
    - 2의 위치 확정(left = 0, k = 1, right = 3)



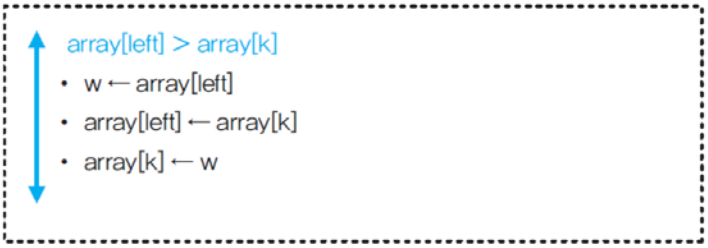
04. 나눈 데이터에 다시 한 번 같은 처리를 실행하는 처리

- QuickSort 안에서 QuickSort 사용하기
  - 첫 번째 QuickSort
    - 5의 위치 확정(left = 0, k = 4, right = 8)
  - 네 번째 QuickSort
    - (k+1<right 경우) QuickSort(array,k+1,right) = QuickSort(array,4+1,8) = QuickSort(array,5,8)
    - 8의 위치 확정(left = 5, k = 7, right = 8)
  - 다섯 번째 QuickSort
    - (left<k-1 경우) QuickSort(array,left,k-1) = QuickSort(array,5,7-1) = QuickSort(array,5,6)
    - 2의 위치 확정(left = 5, k = 6, right = 6)



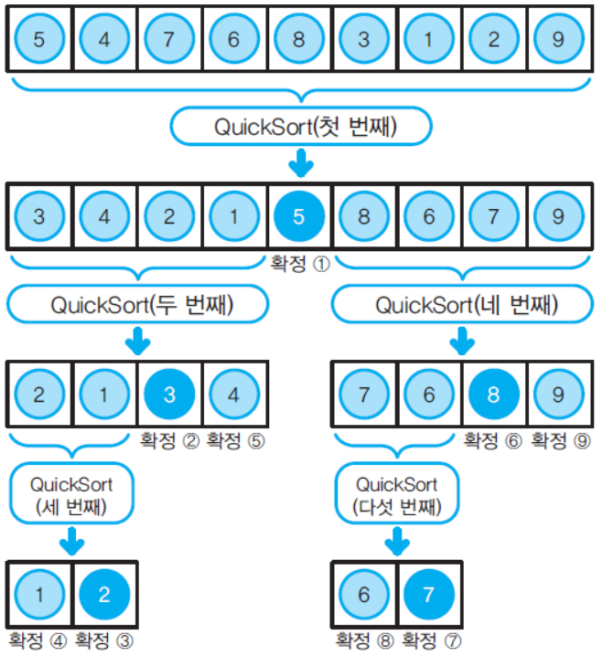
04. 나눈 데이터에 다시 한 번 같은 처리를 실행하는 처리

- QuickSort 안에서 QuickSort 사용하기
  - 교환



04. 나눈 데이터에 다시 한 번 같은 처리를 실행하는 처리

- 퀵 정렬 완료



04. 나눈 데이터에 다시 한 번 같은 처리를 실행하는 처리

- 의사 언어

```
○ 전역 : 정수형 : array[9] = {5, 4, 7, 6, 8, 3, 1, 2, 9} :        전역 선언
○ 보조 프로그램 : QuickSort(정수형 : array[], 정수형 : left, 정수형 : right)
○ 정수형 : left, right, i, k, w
  • i ← left + 1                                  기준값을 경계로 데이터를
  • k ← right                                     대소로 나누는 처리
  ■ i < k
    ■ array[i] < array[left]かつi < right
      • i ← i + 1
    ■ array[k] ≧ array[left]かつk > left
      • k ← k - 1
    i < k
      • w ← array[i]
      • array[i] ← array[k]
      • array[k] ← w
  array[left] > array[k]
    • w ← array[left]
    • array[left] ← array[k]
    • array[k] ← w
  left < k - 1                                    나눈 데이터에 대해 반복적으로
    • QuickSort(array, left, k - 1)               똑같은 처리를 실행하는 처리
  k + 1 < right
    • QuickSort(array, k + 1, right)
  • 맨 앞부터 순서대로 요소의 데이터를 출력한 후에 종료한다
```

● 퀵 정렬의 알고리즘(의사 언어)