1 퀵정렬 Quick Sort

퀵정렬 Quick Sort 알고리즘의 개념 요약

- '찰스 앤터니 리처드 호어(Charles Antony Richard Hoare)'가 개발한 정렬 알고리즘
- 퀵 정렬은 **불안정 정렬**에 속하며, 다른 요소와 비교만으로 정렬을 수행하는 **비교 정렬**에 속한다.
- 분할 정복 알고리즘의 하나로, 평균적으로 **매우 빠른 수행 속도**를 자랑하는 정렬방법
  - 합병정렬(merge sort)와 달리 퀵정렬은 리스트를 **비균등하게 분할**한다.

- **분할 정복(devide and conquer)방법**
  - 리스트를 작은 2개의 리스트로 분리하고 각각을 해결한 다음, 결과를 모아서 원래의 리스트를 정렬하는 전략이다.
  - 분할 정복 방법은 대개 순환 호출(재귀)을 이용하여 구현한다.

- 과정 설명
  1. 리스트안에 있는 한 요소를 선택한다. 이렇게 고른 요소를 피벗(pivot)이라 한다.
  2. 피벗을 기준으로 피벗보다 작은 요소들은 모두 피벗의 왼쪽으로 옮겨지고 피벗보다 큰 요소들은 모두 피벗의 오른쪽으로 옮겨진다. (피벗을 중심으로 왼쪽 : 피벗보다 작은 요소들, 오른쪽: 피벗보다 큰 요소들)
  3. 피벗을 제외한 왼쪽 리스트와 오른쪽 리스트를 다시 정렬한다.
     - 분할된 부분 리스트에 순환 호출을 이용하여 정렬을 반복한다.
     - 부분 리스트에서도 다시 피벗을 정하고 피벗을 기준으로 2개의 부분 리스트로 나누는 과정을 반복한다.
  4. 부분 리스트들이 더 이상 분할이 불가능할 때까지 반복
     - 리스트의 크기가 0이나 1이 될때까지 반복

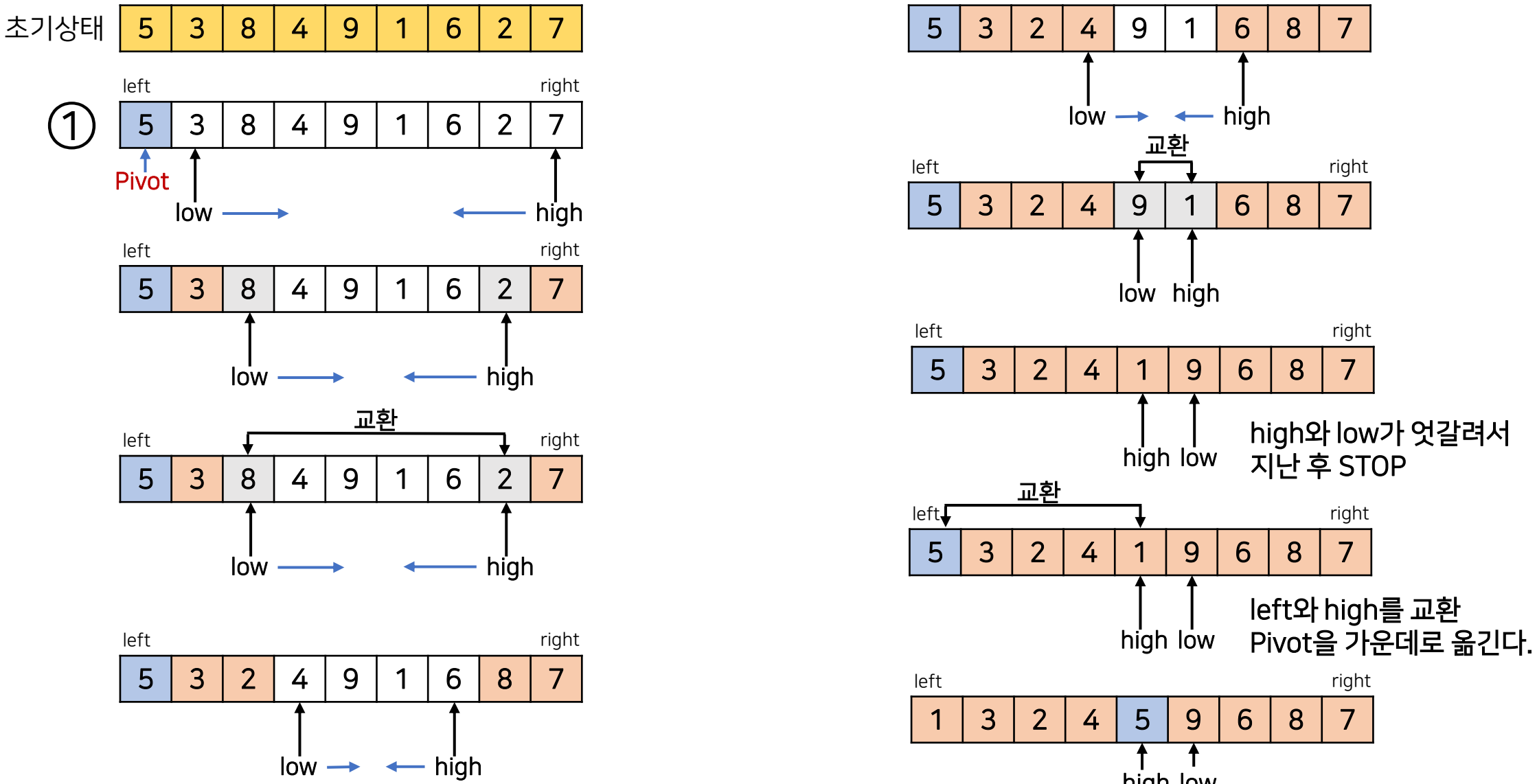
1 퀵정렬 Quick Sort

퀵정렬 Quick Sort 알고리즘의 구체적인 개념

- 하나의 리스트를 피벗(pivot)을 기준으로 두 개의 비균등한 크기로 분할하고 분할된 부분 리스트를 정렬한 다음, 두 개의 정렬된 부분 리스트를 합하여 전체가 정렬된 리스트가 되게 하는 방법이다.

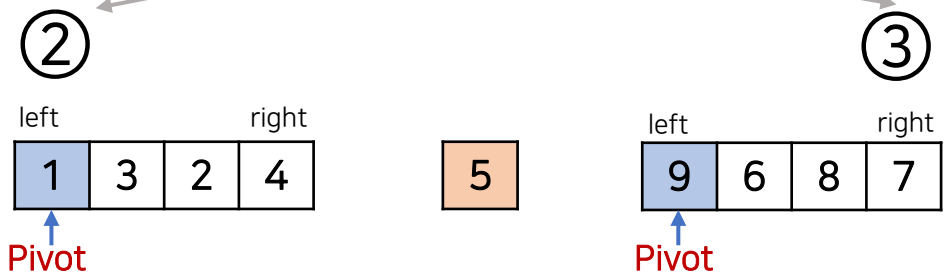
- 퀵 정렬은 다음의 단계들로 이루어진다.
  - **분할(Divide)** : 입력 리스트를 피벗을 기준으로 비균등하게 2개의 부분 리스트(피벗을 중심으로 왼쪽 : 피벗보다 작은 요소들, 오른쪽:피벗보다 큰 요소들)로 분할한다.
  - **정복(Conquer)** : 부분 배열을 정렬한다. 부분 배열의 크기가 충분하지 작지 않으면 순환 호출을 이용하여 다음 분할 정복 방법을 적용한다.
  - **결합(Combine)** : 정렬된 부분 배열들의 하나의 배열에 합병한다.

1 퀵정렬 Quick Sort Not In-Place



1 퀵정렬 Quick Sort Not In-Place

1회차 후 Pivot 5는 이미 제
위치에 있음을 알 수 있다.



Pivot을 제외한
왼쪽 리스트와 오른쪽 리스트를
각각 독립적으로 다시 퀵 정렬(quick sort)를 한다.

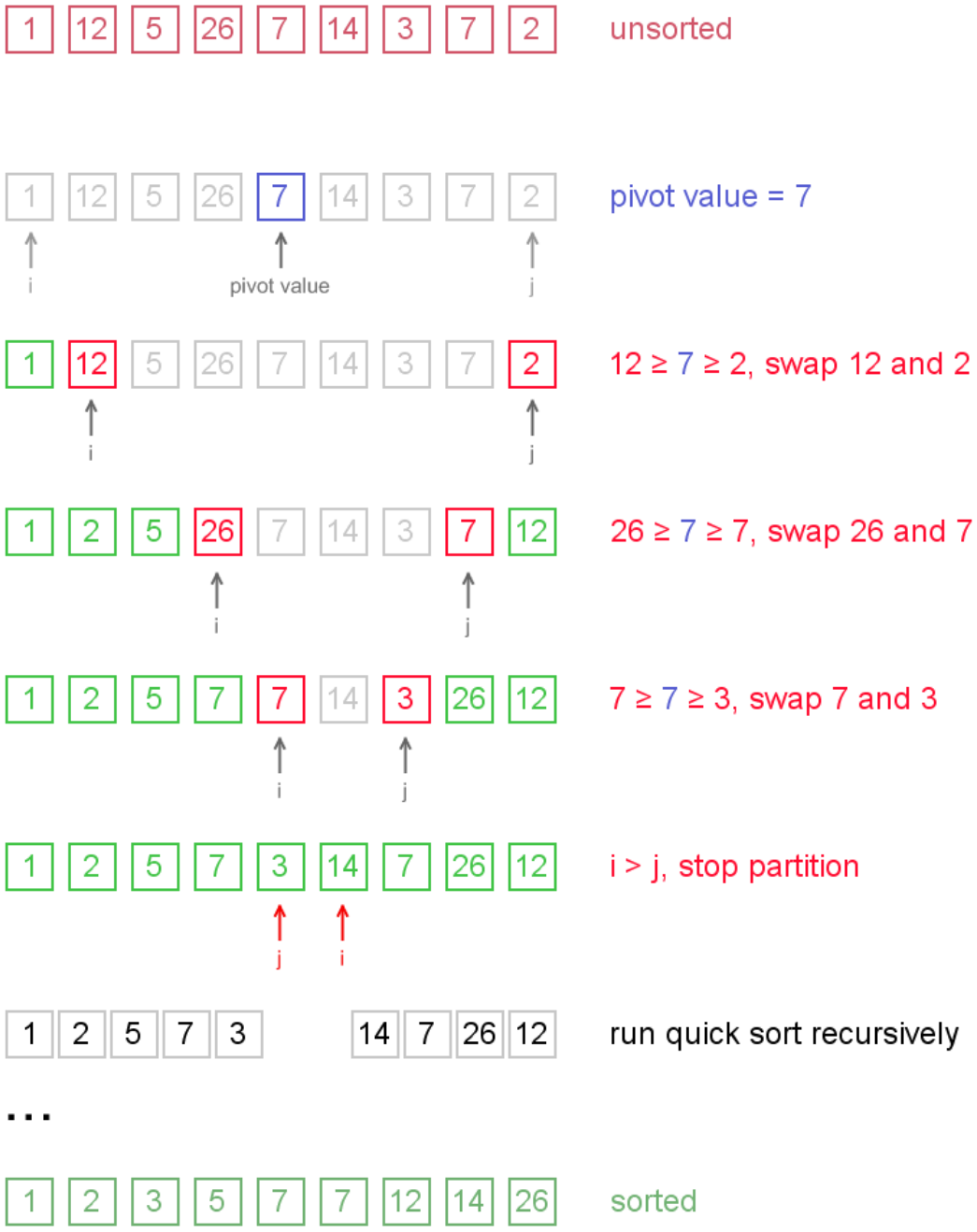
리스트의 크기가 0이나 1이 될 때까지 반복

1 퀵정렬 Quick Sort Not In-Place

퀵정렬 Quick Sort Not In-Place 알고리즘의 구체적인 개념

- 피벗 값을 입력 리스트의 첫번쨰 데이터로 하자.
- 2개의 인덱스 변수(low, high)를 이용해서 리스트를 두 개의 부분 리스트로 나눈다.
- **1회차 : 피벗이 5인 경우,**
  - low는 왼쪽에서 오른쪽으로 탐색하나가다 피벗보다 큰 데이터(8)을 찾으면 멈춘다.
  - high는 오른쪽에서 왼쪽으로 탐색해나가다 피벗보다 작은 데이터(2)를 찾으면 멈춘다.
  - low와 high가 가르키는 두 데이터를 서로 교환한다.
  - 이 탐색-교환 과정은 low와 high가 엇갈릴 때까지 반복한다.
- **2회차 : 피벗(1회전의 왼쪽 부분 리스트의 첫번째 데이터)이 1인 경우,**
  - 위와 동일한 방법으로 반복
- **3회차 : 피벗(1회전의 오른쪽 부분 리스트의 첫번째 데이터)이 9인 경우,**
  - 위와 동일한 방법으로 반복한다.

1 퀵정렬 Quick Sort In-Place



1 퀵정렬 Quick Sort In-Place

퀵정렬 Quick Sort In-Place 알고리즘의 구체적인 개념

- 정렬되지 않은 데이터에서 가운데 원소를 pivot으로 설정하고 가장 왼쪽 요소와 가장 오른쪽 요소가 시작점이다.

- 가장 왼쪽부터 값을 pivot값을 비교하여 pivot보다 큰 값이 나타날 때까지 칸을 오른쪽으로 이동한다.
- 가장 오른쪽부터 값을 pivot값을 비교하여 pivot보다 작은 값이 나타날 때까지 칸을 왼쪽으로 이동한다.
- pivot보다 큰 왼쪽 값과 pivot보다 작은 오른쪽 값을 서로 교환한다.

- 왼쪽 인덱스가 오른쪽 인덱스보다 커지면 이동을 멈추고 그 자리에서 두 배열로 갈라서 각 배열에 위와 같은 방식으로 재귀 호출하여 정렬한다.

- 이 방법은 추가적인 공간을 필요로하지 않기 때문에 메모리 공간이 절약된다는 장점이 있으나, 왼쪽과 오른쪽 값과 교환하는 과정에서 중복 값의 위치가 바뀔 수 있으므로 unstable한 정렬 방법이다.

1 퀵정렬 Quick Sort

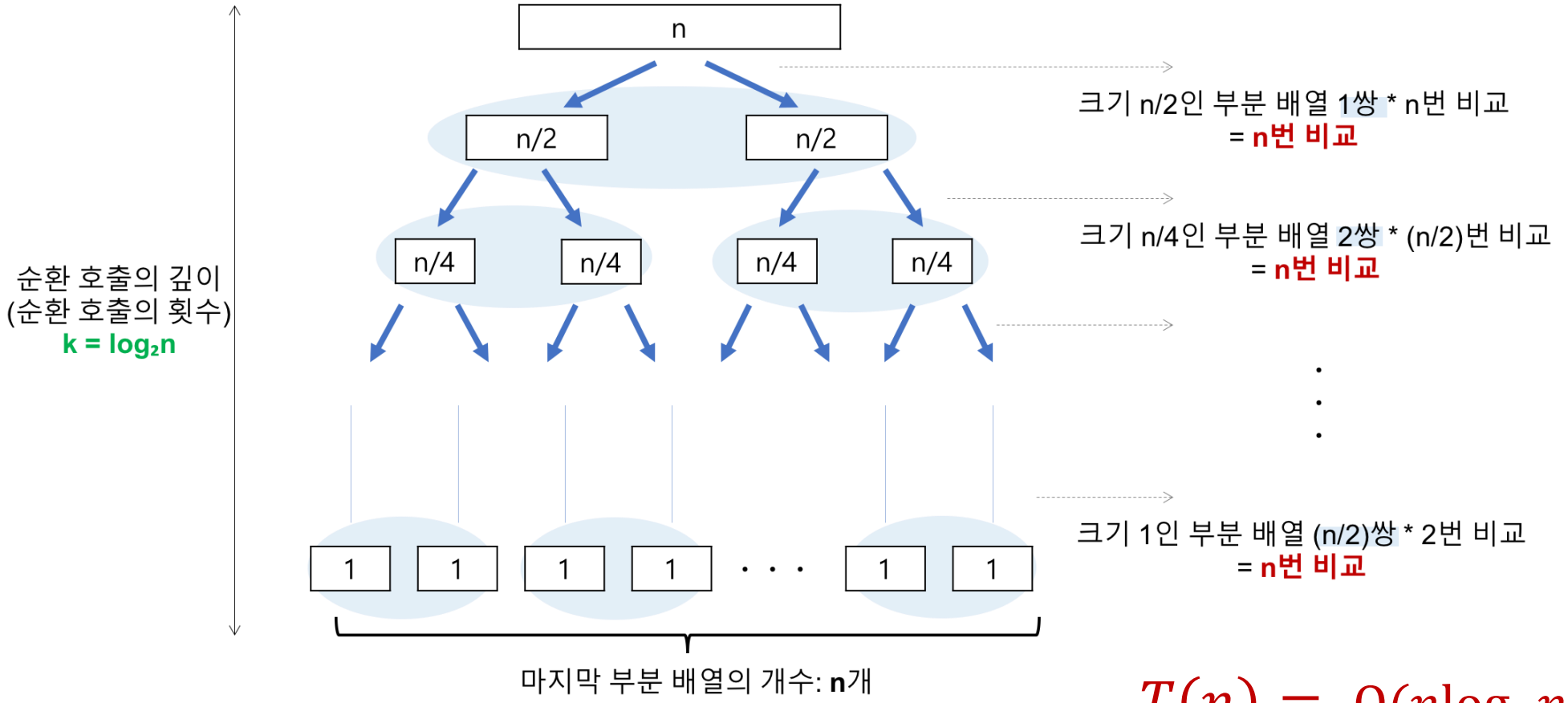
퀵정렬 Quick Sort 알고리즘의 특징

- 장점
  1. 속도가 빠르다.
     - 시간 복잡도가 $O(n\log_2 n)$를 가지는 다른 정렬과 비교했을 때도 가장 빠르다.
  2. 추가 메모리 공간을 필요로 하지 않는다.
     - 퀵 정렬은 $O(\log_2 n)$만큼의 메모리를 필요로 한다.
- 단점
  1. 정렬된 리스트에 대해서는 퀵정렬의 불균형 분할에 의해 오히려 수행시간이 더 많이 걸린다.
- 퀵 정렬의 불균형 분할을 방지하기 위하여 피벗을 선택할 때 더욱 리스트를 균등하게 분할할 수 있는 데이터를 선택한다.
  - EX)리스트 내의 몇개의 데이터 중에서 크기순으로 중간 값(medium)을 피벗을 선택한다.

1 퀵정렬 Quick Sort

퀵정렬 Quick Sort 시간복잡도

- 최선의 경우
  - 비교횟수



$$T(n) = O(n\log_2 n)$$

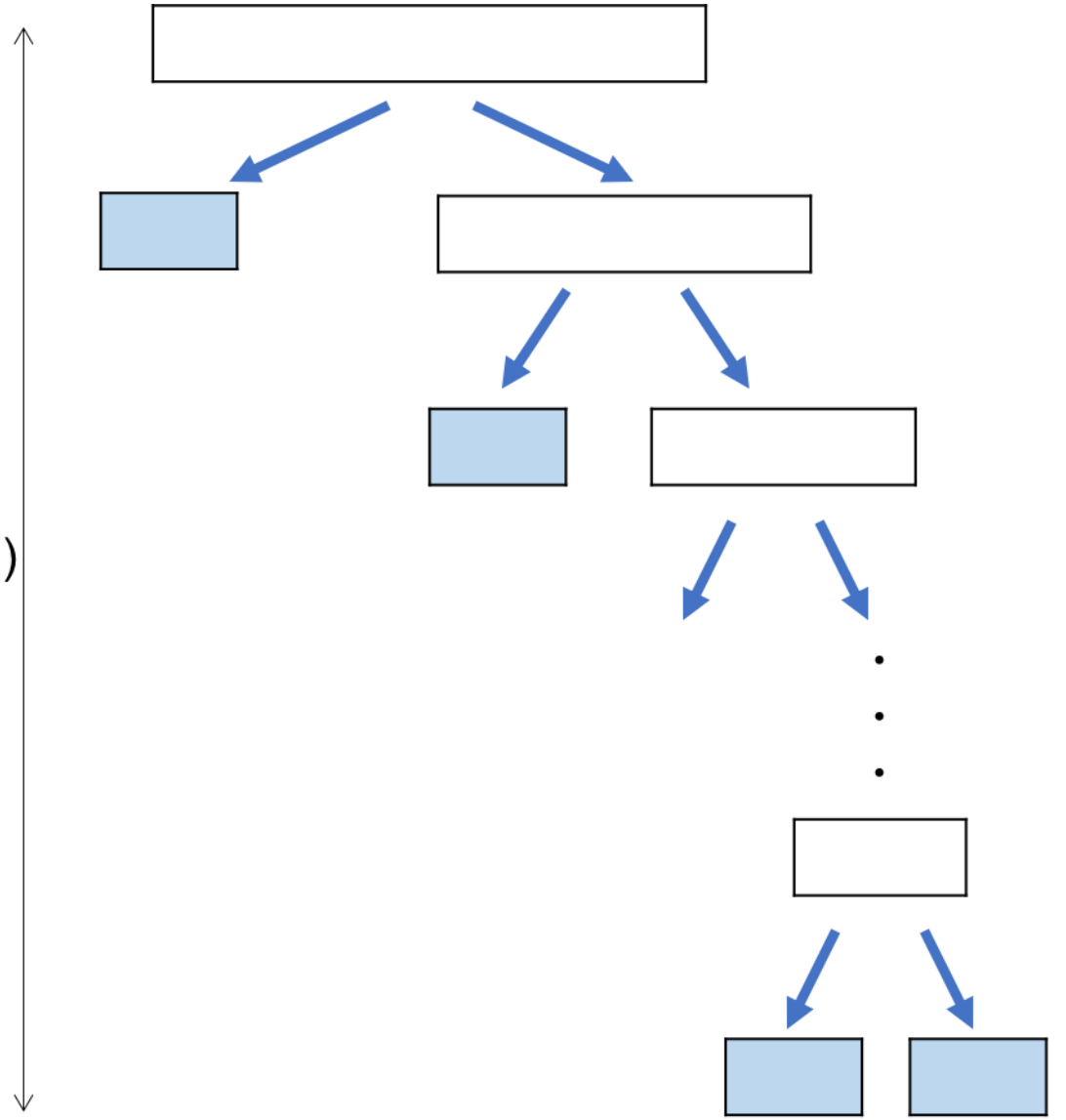
1 퀵정렬 Quick Sort

퀵정렬 Quick Sort 시간복잡도

- 최악의 경우
  - 리스트가 계속 불균형하게 나누어지는 경우
  - 특히 이미 정렬된 리스트에 대하여 퀵정렬을 실행하는 경우

$$T(n) = O(n^2)$$

순환 호출의 깊이
(순환 호출의 횟수)
n



1 퀵정렬 Quick Sort

퀵정렬 Quick Sort 시간복잡도

- 평균
  - 평균 $T(n) = O(n\log_2 n)$
  - 시간복잡도가 $O(n\log_2 n)$을 가지는 다른 정렬 알고리즘과 비교했을 때도 가장 빠르다.
  - 퀵정렬이 불필요한 데이터의 이동을 줄이고 먼 거리의 데이터를 교환할 뿐만 아니라, 한번 결정된 피벗들이 추후 연산에서 제외되는 특성때문이다.

4 정렬 알고리즘 시간 복잡도 비교

Name	Best	Avg	Worst	Run-time(정수:60,000개) 단위 : sec
삽입정렬	n	n^2	n^2	7.438
선택정렬	n^2	n^2	n^2	10.842
버블정렬	n^2	n^2	n^2	22.894
퀵정렬	$n\log_2 n$	$n\log_2 n$	n^2	0.014