퀵 정렬

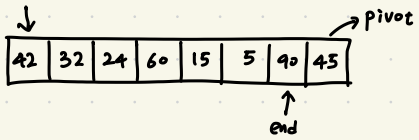
재귀 사용

- 한 영역에서 start와 end, pivot 지정
- int[] A 지정
- A[start] < A[pivot] 이면 start++
- A[end] > A[pivot] 이면 end--
- A[start] > A[pivot] && A[end] < A[pivot] 이면
  A[start] 와 A[end] 를 swap
  start++, end--

  (start == end 까지 반복)
- start == end 이면
  A[start] < A[pivot] 이면 A[pivot] 을 A[++start] 로
  A[end] > A[pivot] 이면 A[pivot] 을 A[--end] 로

pivot을 정해 pivot을 기준으로 범위를 나누는 것이 핵심

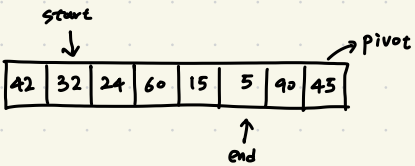
1)



⇒ 42 < 45 → start++

90 > 45 → end--

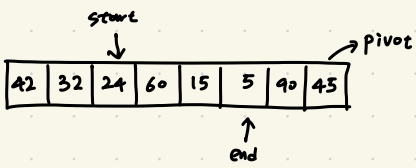
2)



⇒ 32 < 45 → start++

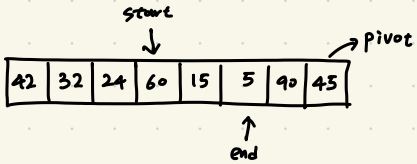
5 < 45 → end 유지

3)



⇒ 24 < 45 → start++

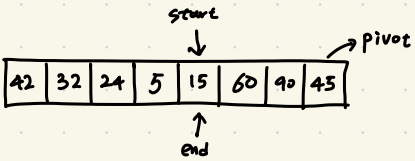
4)



⇒ 60 > 45 and 5 < 45

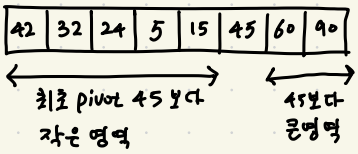
→ 60과 5 swap → start++, end--

5)



⇒ 15 < 45 → 15의 오른쪽에 데이터 삽입

6)



각 영역에 대해서 pivot을 재지정 후

알고리즘을 실행하면 정렬이 완성됨

⇒ 최악의 경우에는 복잡도가 $O(n^2)$ 이 되는경우가 있음

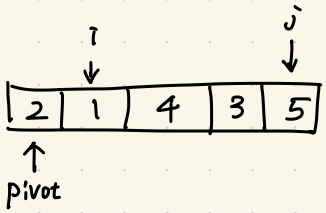
Sudo

ex) | 0 | 4 | 1 | 2 | 3 | 5 | = A = new int[6]

N=5 K=2

pivot = A.length / 2 = 3

A[pivot] $\overset{swap}{\longleftrightarrow}$ A[1]



1) A[i] < A[pivot] ⇒ i++

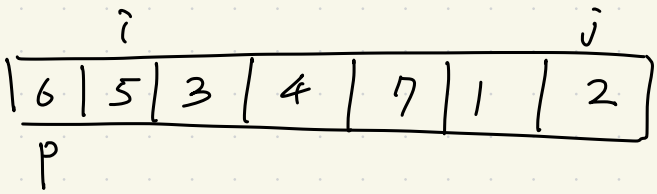
   A[j] > A[pivot] ⇒ j--

2) A[i] > A[pivot] and A[j] < A[pivot] ⇒ A[i] $\overset{swap}{\longleftrightarrow}$ A[j], i++, j--

3) i == j ⇒ if(A[i] < A[pivot]) ⇒ A[i-1] 에 A[pivot] 추가

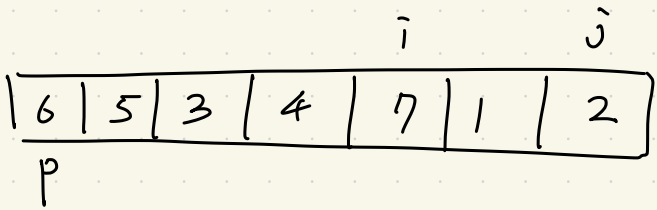
   if (A[i] > A[pivot]) ⇒ A[i+1] 에 A[pivot] 추가

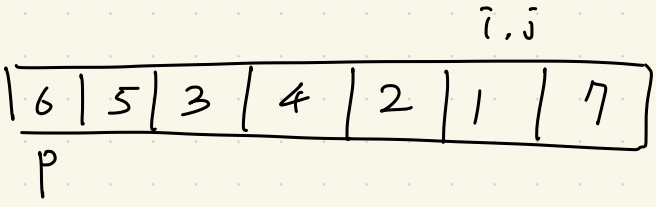
4) 1) ~ 3) 반복



start = 0
end = 6

pivot = A[0]



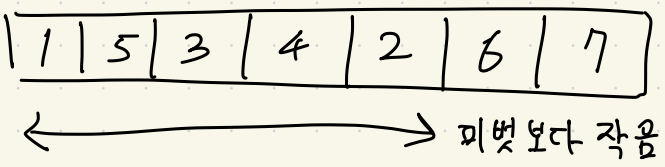


A[0] = A[j] = A[5]

A[5] = pivot = A[0]

pivot

⇒ A[start]와 A[j] 바꿈



return 5 ⇒ 리턴되는 피벗 값