1. 퀵 정렬

말 그래도 빠른 정렬이라고 할 수 있다. 시간복잡도는 O(nlogn)으로 앞의 굼뱅이들보다는 빠르다고 할 수 있다. 이 정렬은 분할과 정복을 사용한다.

분할과 정복은 하나의 문제를 빠개서 해결하고 합치는 개념인데 이 정렬이 그렇다

특정한 값을 기준으로 왼쪽에는 작은 값 오른쪽에는 큰 값을 배치시 킨 후

왼쪽에서 다시 정렬 오른쪽에서도 다시 정렬, 하여 합치는 경우이다.

이 경우에는 전체에서 한번 읽는 과정 o(n)과 n이 나눠진 경우 정렬을 수행하는 o(logn)이 합쳐져 평균적으로 O(nlogn)이 나오게된다.

피벗은 일반적으로 맨 앞을 정하는 경우에 많이 사용한다.

3 7 8 1 5 9 6 10 2 4

3을 앞에서는 큰 값을 찾고 끝에서는 차례로 작은 값을 찾는다

그러면 큰 값(7) 작은 값(2)가 나오게 될것이다. 그러면 이를 스와핑 해준다.

3 2 8 1 5 9 6 10 7 4

다시 또 찾는다 큰 값(8) 작은 값(1)이 나오게 되면 스와핑

3 2 1 8 5 9 6 10 7 4

또 찾는다. 큰 값(8)과 작은 값(1)인데 얻갈린 경우에는 피벗과 작은 값을 교체한다

1 2 3 8 5 9 6 10 7 4

이렇게 되면 피벗을 기준으로 왼쪽에는 작은 값 오른 쪽은 큰 값이 위치한다.

이러면 다시 왼쪽 따로 오른쪽 따로 정렬을 수행한다.

퀵정렬의 경우에는 평균이 O(nlogn)이고 최약의 경우는 O(N2)이 나온다.

최악의 경우는 바로 이미 정렬된 경우에 정렬을 시도하는 경우이다. 그러면 피벗을 찾는 과정에 1 2 3 4 5 6 7 8 9 10

일 때 1보다 큰 값 2는 찾지만 1보다 작은 값은 없기에 얻갈리며 스와핑이 이루어진다. 이를 계속 반복하면 10 + 9 + 8 \*\*\* +1 까지 이루어지며 결국은 좋지 않은 속도를 가지게 된다.

이럴 경우를 대비하여 피벗을 랜덤하게 선정하는 방법과, 정렬을 하기전에 배열을 엎어버리는 방법도 존재한다.

2. 병합정렬

이 친구도 분할과 정복을 사용한 정렬방법으로 무조건 O(nlogn)을 보장한다는 점이 있다.

왜냐면 무조건 절반으로 빠개기 때문이다. 피벗이 없이 무조건 반으로 빠갠다.

7 6 5 8 3 5 9 1

이 있으면

7 6 5 8 3 5 9 1

이렇게 빠개서 정렬을 수행한다.

이제는 각 자를 비교한다

2의 배수의 개수만큼 묶어서 비교한다.

7과 6을 비교할수 있게 배열을 만들고 순서대로 배열을 한다 비교하고 스와핑하면 6 7 이 될것이다. 이를 다른 배열에도 만든다.

6 7 5 8 3 5 1 9

을 각각 비교할 수 있게 배열을 할당한다.

6 7 5 8 을 비교하게 4개의 배열을 할당하고

6과 5를 비교한다 5가 작네, 7과 5를 비교한다 5가 작네 이러면 k[0]은 5가 들어간다.’

K[1]은 6 7 8을 비교한다. 6이 작으니 k[1]은 6이 들어간다. 이를 반복하면 5678이 된다.

이제 합쳐진 6 7 5 8 3 5 1 9를 또다시 배열에 쳐넣는다 그러면 정렬이 완료된다.

이렇게 하면 O(nlogn)을 보장한다. 그러면 퀵 정렬보다 좋은거 아닌가? 하겠지만

쳐넣을 공간을 만들기 위해 배열을 할당한다는 점에서 메모리적으로는 비효율적이라고 할 수 있다.