

# Sorting

2018년 6월 12일 화요일

오후 1:36

## • Bubble Sort

1 3 7 2 4 5

① 두 개 숫자씩 순서를 바꾸기를 반복.

-  $O(n^2)$ , practical하게 보면 효율적.

## • Selection Sort

1 3 7 2 4 5

① 하나를 select하고 다른 것들과 비교.

-  $O(n^2)$ .

## • Heap Sort

1 3 7 2 4 5

① last나 root를 교체하기를 반복.

-  $O(n \log n)$

## • Merge sort

1 3 7 2 4 5

① n:2로 덩어리를 나눔. (recursive 하게)

② merge가 중요. - 맨 앞을 비교해서 뒤로 보낸다.

-  $O(n \log n)$ , huge data로 sorting하는데 중요.

## • Quick Sort

1 3 7 2 4 5

① merge랑 비슷한데 나눌 때 하나를 기준으로 작은 것 left, 큰 것 right로 나눈다.

-  $O(n^2)$   $\rightarrow O(n \log n)$   $\rightarrow \Theta(n \log n)$ , 별로 안 빠른데 왜 quick인가?

-  $n^n$  정도로 worst case가 나옴.

-  $\frac{I(n)}{n+1} \leq 1 + 2 \log n$

- 이론적으로는 꽤 빠르지 않는데 캐시를 잘 활용하도록 설계해서 실제로는  $O(n \log n)$ 보다 빠르다.

① 1 3 7 2 4 5

Sorting이  $O(n \log n)$ 보다 빠를 수 있나?  $\rightarrow$  No.

## • AVL

- n개 insert에  $O(n \log n)$

- 이것 구현할 수 있으면 웬만한 알고리즘들 다 구현할 수 있음.

## • Two-Four Tree

- B Tree라는 게 있음.

- 모든 external의 height가 같음.

- internal node는 max child가 2 or 3 or 4개.

- 분기(카를로) search라는 게 있는데, 이것 Two-Four Tree로 구현함.

※ 시험.

- BST로 sorting하기. (어떻게?)