정렬-1

김수진

정렬을 하는 이유

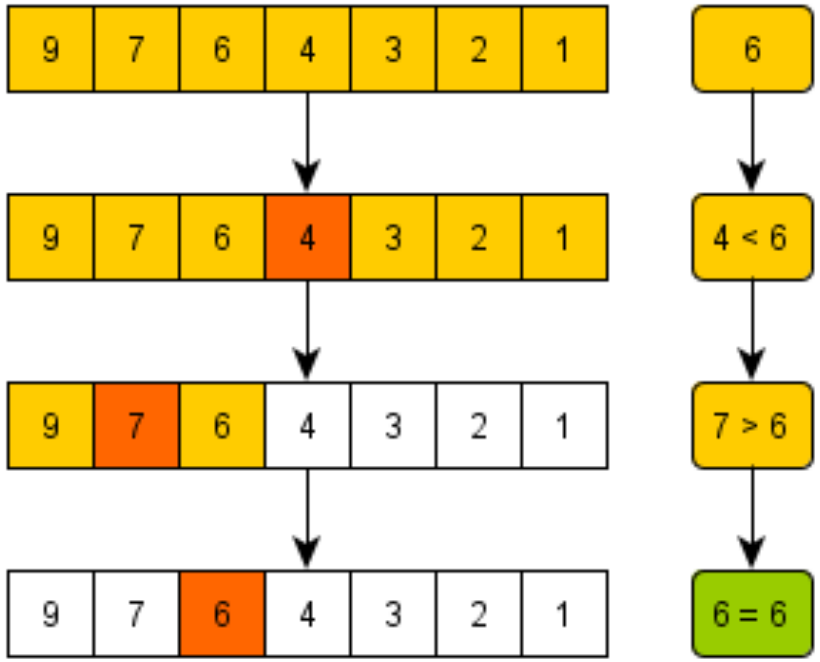
이진 탐색을
가능하게 하기 위해

이진탐색은 정렬이 되어 있어야함
그렇지않으면 순차탐색을 해야함
그럼 오래걸리니까

그리고 우리는 탐색하는 경우가 많음

이진탐색

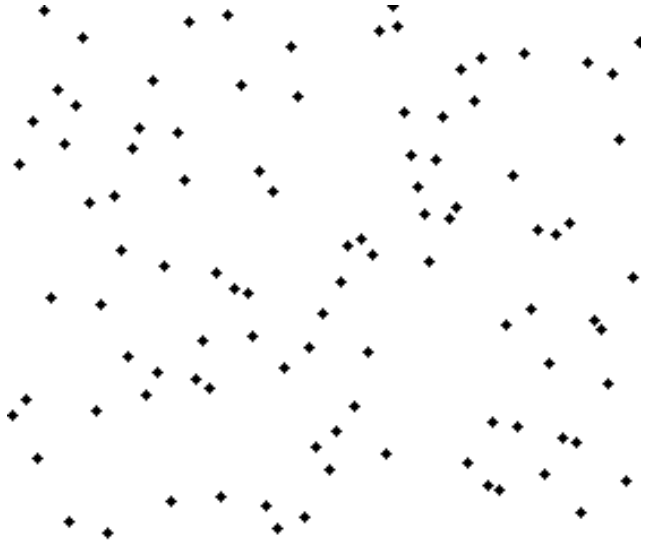
- 컴퓨터가 어떤 값을 집어 올리는 위치가 후보군의 가운데인 탐색 알고리즘
- 최악의 경우에도 logn의 성능



$T(N^2)$ 알고리즘

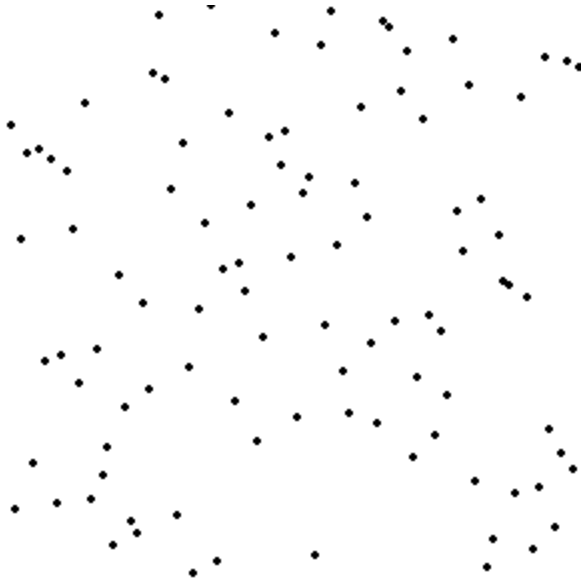
버블 삽입 선택

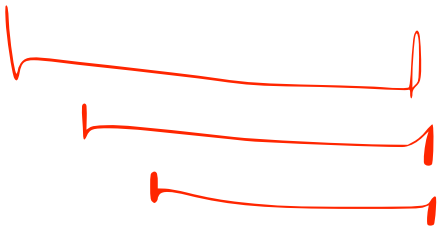
Bubble sort(stable sort)



- 두개의 인덱스를 비교하여 값을 정렬하는 방법
- 오름차순일 경우 두 인덱스끼리 비교하여 1바퀴 돌면 가장 큰 값이 맨 뒤에 저장됨.
- 최대 $n(n-1)/2$번 정렬을 수행, 즉 $O(n^2)$의 성능
- 이미 정렬된 자료에서 최선의 성능 O(n)

Selection sort(unstable sort)



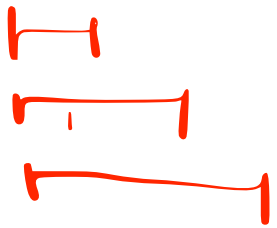


- 1번째부터 끝까지 훑어서 가장 작은 게 1번째 … n-1번째부터 끝까지 훑어서 가장 작은 게 n번째
- 최대 최소 모두 $n(n-1)/2$번 정렬을 수행, 즉 항상 $O(n^2)$
- 버블 정렬보다 2배정도 빠름.

계산복잡도는 같지만
실제로 계산 성능은 2배정도 빠름.
버블정렬은 매번 스왑함

Insert sort(non-stable sort)

6 5 3 1 8 7 2 4



- K번째 원소를 (1 ~ K-1)까지 비교해 적절한 위치에 끼어 넣음
- 평균적으로 $O(n^2)$의 성능
- 작은 값이 뒤에 몰려 있는 경우 최악의 성능
- 이미 정렬된 자료에서 삽입/삭제 시 다시 정렬하려고 할 때 최고의 성능 O(n)

O(N LOG N) 알고리즘

Merge Sort(stable sort)

6 5 3 1 8 7 2 4

log n만큼 자르고
n만큼 합병

- 마지막 한 개가 될 때 까지 자른 후 자른 순서를 역순으로 크기를 비교해 병합

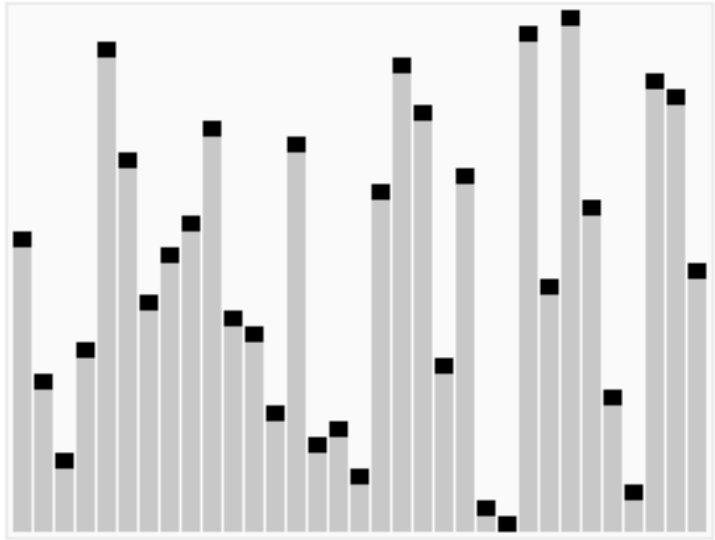
ex) A의 배열 크기 N1, B의 배열 크기 N2 -> O(N1+N2) = O(N)

A의 배열 분활 과정 N1/2, N1/4 … lgN

=> O(NlogN)

- 정렬을 위한 배열을 하나 더 생성 (즉 배열의 크기가 N이었다면 2N의 메모리 차지)
- 데이터 상태에 영향을 받지 않음(아무렇게나 분포되어도 성능이 같음)

Quick sort(non-stable sort)



- 알고리즘 방법

1. 기준이 되는 피봇을 뽑기
2. 피봇보다 작은 것을 다 앞으로 두고 마지막에 피봇을 그 뒤에 옮김.
3. 나누어진 영역에서 각각 다시 피봇을 잡고 정렬

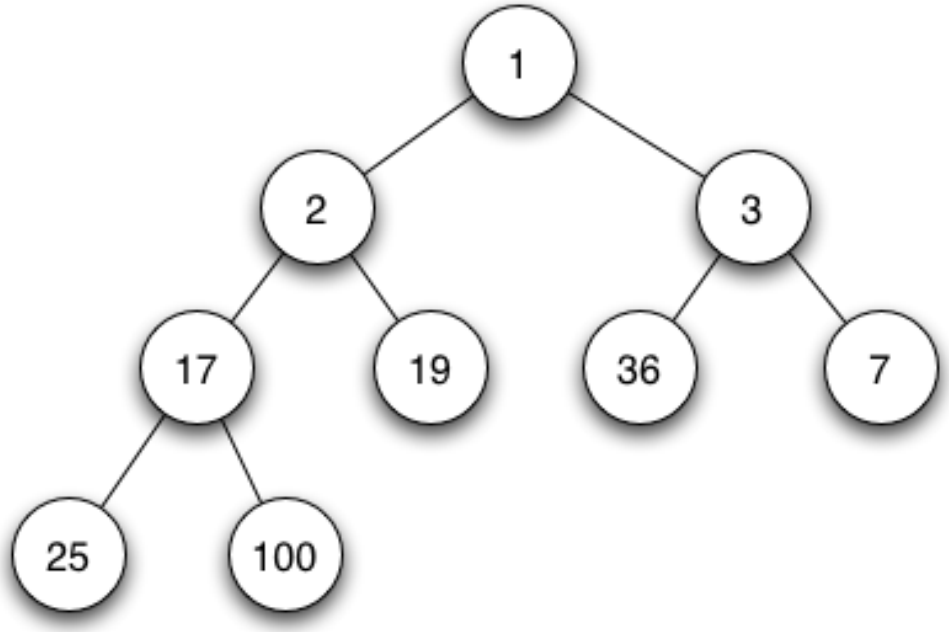
Quick sort(non-stable sort)

- 분할과 동시에 정렬을 하는 알고리즘
- 따로 메모리 크기가 필요하지 않다.
- 배열이 이미 정렬되어 있는 경우 분할이 N만큼 일어나 $O(n^2)$의 최악의 성능을 가짐
- 평균적인 상황에서는 어떤 알고리즘보다 최고의 성능을 가짐

OS에서 사용

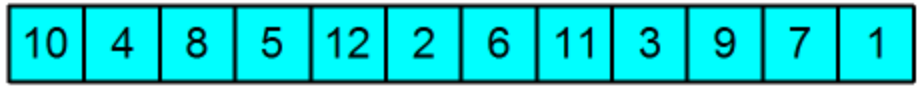
Heap sort(stable sort)

- Heap이란? = 완전 이진 트리
- Max heap과 Min heap이 있다.
- O(1)안에 최댓값(혹은 최솟값)을 찾을 수 있다.



Heap sort(stable sort)

- 장점 :
  - 추가 메모리가 전혀 필요하지 않음
  - 항상 O(nlogn)의 성능을 가짐
  - 퀵정렬 보다 일반적으로 빠르게 동작



메모리참조가 지역화가 되어있지 않아서
CPU의 캐시의 hit율이 높음
그래서 대부분의 컴퓨터아키텍처에 효율적

비교

- N = 1000(천)

```
랜덤 숫자 범위(1~n) : 1000


선택 정렬 수행시간 : 0.010000
삽입 정렬 수행시간 : 0.004000
버블 정렬 수행시간 : 0.009000
병합 정렬 수행시간 : 0.002000
퀵 정렬 수행시간   : 0.000000

--------------------------------
Process exited after 1.273 seconds with return value 0
계속하려면 아무 키나 누르십시오 . . .
```

비교

- N = 10000(만)

```
랜덤 숫자 범위(1~n) : 10000


선택 정렬 수행시간 : 0.896000
삽입 정렬 수행시간 : 0.329000
버블정렬 수행시간 : 0.891000
병합 정렬 수행시간 : 0.016000
퀵 정렬 수행시간   : 0.000000

--------------------------------
Process exited after 3.345 seconds with return value 0
계속하려면 아무 키나 누르십시오 . . .
```

비교

- N = 100000(10만)

```
랜덤 숫자 범위(1~n) : 100000


선택 정렬 수행시간 : 90.340000
삽입 정렬 수행시간 : 37.481000
버블 정렬 수행시간 : 91.816000
병합 정렬 수행시간 : 0.183000
퀵 정렬 수행시간   : 0.037000

--------------------------------
Process exited after 221.8 seconds with return value 0
계속하려면 아무 키나 누르십시오 . . .
```