정렬

- 데이터를 특정한 기준에 따라서 순서대로 나열.
  (데이터를 가공할때 사용)
  가공된 데이터는 탐색이 용이하다.
  - 탐색의 전처리 과정

① 선택 정렬 (Selection Sort)

(매번 가장 작은 것을 선택)

반복.
1. 정렬되지 않은 데이터 중에서 가장 작은 값 선택.
2. 정렬되지 않은 데이터 중에서 맨 앞 데이터와 swap

- 시간 복잡도

  N-1 번 작업을 해야한다.

  가장 작은 수를 찾기 위한 연산 $N + (N-1) + (N-2) + \cdots + 2$
  (1회 2회 ... N-1회)

  => $O(N^2)$

② 삽입 정렬 (Insertion Sort)

(데이터를 하나씩 확인, 적절한 위치에 삽입)

- 필요한 경우에만 정렬하기 때문에 정렬되어 있을 수록 효율 ↑

1. 첫번째 데이터는 정렬되었다고 가정
반복.
2. 정렬되지 않은 데이터 중에서 가장 앞 데이터를 선택.
3. 정렬된 데이터에서 선택된 데이터가 삽입될 위치 탐색. → 삽입

- 시간 복잡도

  최선의 경우 $O(N)$

  Worst : $O(N^2)$

③ 퀵 정렬

- 대부분의 프로그래밍 언어에서 정렬 라이브러리의 근간.

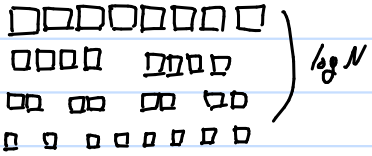
(기준 데이터를 설정하고 그 기준으로 큰 데이터와 작은 데이터의 위치를 바꾼다.)

- 피벗.
  (여러 분할 방식이 존재하지만, 대표적인 two 분할 사용)
  Hoare Partition

1. 리스트의 첫번째 데이터를 (Pivot)
반복 2. 왼쪽에서부터 Pivot보다 큰 값을, 오른쪽에서부터 Pivot보다 작은 값을 찾아서 교환
3. 중간 지점에 Pivot을 위치 시킨 후, Left 리스트 (작은 값), Right 리스트 (큰 값)을 재귀적으로 반복.

— 시간 복잡도

평균적으로 $O(N \cdot \log N)$.

최선의 경우 : 항상 절반으로 나뉘어지는 경우 →  $\log N$

최악의 경우 : 이미 정렬되어 있는 경우

④ 계수 정렬 (Count Sort)

(특정 조건이 부합할 경우 매우 빠르다)

시간복잡도

데이터의 개수 N.

데이터의 최대값 K (100만 이하인데 효과적)

=> 최악 : $O(N+K)$

공간 복잡도 : $O(N+K)$

1. 데이터의 개수를 Count 한 뒤로 리스트 준비. (메모리 공간 : $K+1$)
2. 데이터들을 모두 Count.
3. 작은 값부터 Count 수 만큼 출력 => 정렬된 결과.

⑤ Radix 정렬과 함께 가장 빠른 정렬.