2020년 6월 19일 금요일 오전 12:50

3. 선택정렬과 삽입정렬

1) 선택 정렬 Selection Sort → 대표적으로 버블에 비교하는 정렬

→ (작은 값을 앞으로) 반복

ex - 4 2 3 9

첫 Index ← 2 4 3 9

2 3 4 9

각 위치에 어떤 값이 들어갈지 찾기

첫 번째 index부터 뒤에 있는 요소들과 비교하며 더 작은 요소 발견 시 자리 변경

(0번 Index의 값 찾고, 1번, 2번, 3번 ...)

2) 삽입 정렬 Insertion sort

각 값이 어떤 위치에 가야할지 탐색

→ Data는 적절한 위치에 삽입 정렬

→ 거의 정렬된 리스트에서 매우 빠름

but, 역순 정렬에서는 아주 느림

ex 2 4 7 1

→ → →

1 2 4 7

4. 알고리즘 평가법

→ 기준 ①: 시간 → 코드의 줄수

기준 ②: (저장) 공간

1) 시간 단순히 프로그램 실행시간 X

→ 시간복잡도로 판별

→ Data가 많아질 때 소요시간이 얼마나 규칙적 증가하냐

ex - Data의 수	알고리즘 A	알고리즘 B
10개	10	10
40개	40	400
1000개	1000	10000

시간복잡도가 크다

시간복잡도가 크다

→ 더 느린 알고리즘

* 시간복잡도 분석 시뮬레이션, 초, Σ …

• 점근표기법 (Big-O Notation)

→ n의 영향력이 가장 큰 부분의 차수만 기록

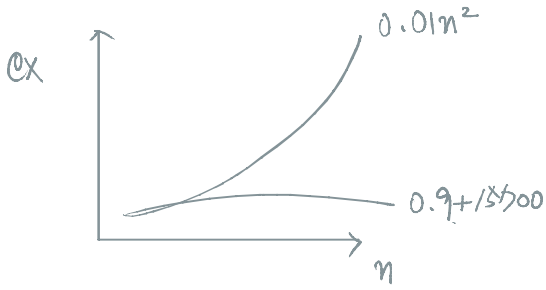
Data 크기 ←

↳ ∵ n이 작을 땐 큰 차이 없지만 커질 수록 큰 격차 만들어냄

ex_ 소요시간	Big-O
$2n+10$	$O(n)$
$2n^2+3n+1$	$O(n^2)$
$9n^3+5n^4$	$O(n^4)$
$20\lg n+9$	$O(\lg n)$

↓ 영향 적음

$\lg_2 n$



ex_ n	O(1)	O(n)	$O(n^2)$
[100	1초	1초	1초] 결과비교 ↙
200	1초	2초	4초
1000	1초	10초	100초
10000	1초	100초	10000초

* 상수는 무관, 실제 측정해봐야 함