과목 : 알고리즘 (방영철)
학번 : 2014150010
이름 : 김홍준
보고서 : 영화 데이터의 정렬 분석

목차

---

1. 정렬 소개

- 개론

  - <빅 오 정의>

    $O(g(n))$ = { $f(n)$: there exist positive constants c and n_0 such that $0 \le f(n) \le cg(n)$ for all $n \ge n_0$ }

  - 삽입정렬, 합병정렬, 퀵 정렬, 힙 정렬, 기수 정렬은 데이터 정렬 알고리즘 종류의 일종. 각 정렬 알고리즘은 성능이 다르며, 구현 (logic) 또한 다름.

  - 이 중 성능 분석을 하면, O(n^2) 은 삽입 정렬, 그 외에 나머지 정렬 알고리즘은 O(nlgn) 임.

  - O(n^2)의 성능인 정렬 알고리즘 : 삽입정렬, 버블정렬, 선택정렬
    (삽입정렬만 다룰 예정)

  - O(nlgn)의 성능인 정렬 알고리즘 : 합병정렬, 퀵정렬, 힙정렬, 기수정렬

  - 대량의 데이터를 통해 각 정렬 알고리즘의 특성을 확인할 것이며, 성능 측정을 할 것임.

    .

1) 삽입정렬

- 시간복잡도가 O(n^2)인 알고리즘 중 하나임.

- 기존의 데이터 위치를 바꾸는 정렬 방식에 비해, 정렬을 하기 위해 특정 데이터를 적절한 위치에 삽입하는 방식. 즉 필요한 경우에만 위치를 바꿈. 또한 데이터를 접근하는 로직에 있어서 점진적임.

- 필요한 경우에만 위치를 바꾸고 비교 횟수를 효율적으로 줄이기에 O(n^2) 성능의 정렬 알고리즘 중에 가장 빠른 알고리즘[2]. 하지만 데이터의 수가 기하급수적인 경우에는 한계가 있으며 데이터의 초기 정렬 상태가 성능을 좌우함.

2) 합병정렬

- O(nlgn) 시간복잡도 알고리즘 중 하나.

- 일반적인 O(n^2) 정렬 알고리즘의 점진적 로직과 달리 분할정복 방식. 데이터들을 더이상 나눌 수 없을 때 까지 (1이 될 때 까지) 계속해서 쪼개고(divide) 2개씩 계속해서 비교를 하며 합치며 정렬을 함. 또한 정렬된 값들을 저장할 임시배열이 필요함.

- 평균 시간복잡도가 nlgn이 나오며 안정적이지만, 퀵 정렬에 비해 느린 경우가 있음. 하지만 임시 배열을 선언해야 하기

에 메모리 낭비가 초래됨. 동일한 정렬 기준인 데이터의 정렬의 전과 후가 동일 (stable)

3) 퀵 정렬

- O(nlgn) 시간복잡도 알고리즘 중 하나.
- 특정 값을 기준으로 큰 데이터와 작은 데이터를 반으로 나눔. 그 특정 값을 피벗이라 하고 그 피벗을 기준으로 왼쪽 집합과 오른쪽 집합으로 나누는 알고리즘.
- A연산을 실행 (A연산 = 인덱스 0번의 데이터를 피벗 값으로 두고, 점진적으로 데이터 방문 시 피벗 보다 작은 값이면 왼쪽 크면 오른쪽으로 swap.) 그 후 피벗의 위치가 결정이 되면 왼쪽 집합과 오른쪽 집합에 각각 똑같이 A연산을 실행. 즉 분할 정복 알고리즘의 종류 중 하나.
- 퀵 정렬은 최초의 데이터의 상태에 따라 성능이 결정됨
  - Worst case : O(n^2) -> A연산 시, 데이터가 모두 swap이 발생하는 경우.
  - Average case : O(nlgn) -> 분할 정복이 가능한 상태.
  - Best case : O(n) -> 데이터가 모두 정렬이 되어 있는 상태인 경우.
- 최악의 경우에는 분할 정복의 이점을 이용을 못함. 하지만 평균적으로 분할 정복의 이점으로 좋은 성능을 갖음. 일반적

으로 병합정렬에 비해 빠르지만 안정적이지 못함[1]. (병합정렬은 divide 비율이 일정함).

- 또한 동일한 정렬 기준을 가진 것의 순서가 다를 수 있음. 동일한 정렬 기준인 데이터의 정렬의 전과 후가 동일(stable)

4) 힙 정렬

- O(nlgn) 시간복잡도 알고리즘 중 하나.

- 데이터를 표현하고자 할 때 데이터를 이어붙이는 것을 트리라 하고 자식이 두개인 경우를 이진트리라 함. 이는 각 정점에 최대 두개까지 자식을 갖음. 이진트리 중 왼쪽에서 오른쪽으로 차근차근 들어가는 구조를 완전 이진트리라 하며, 힙은 완전 이진트리를 이용하여 데이터를 빠르게 접근할 수 있음.

- 힙 정렬 과정

  - Max-Heapify (lgn) : 최대 힙에서, 트리 안에 특정 노드 때문에 힙이 붕괴 되는 경우, 그 노드의 자식 노드 중 큰 값과 swap을 해줌. 이 작업의 수행 시간은 O(lgn)

  - Build-Max-Heap (nlgn) : 힙 구조가 아닌 이진트리를 힙 구조로 만드는 것으로, 노드의 갯수 만큼 Max-Heapify 를 함 , 즉 O(n) * O(lgn)
  - Heapsort (nlgn) : 최종적으로 힙 정렬을 함. 즉 O(nlgn)

- 힙 정렬은 처음 힙을 구성하는데 시간이 오래 걸림. 하지만 구성이 된 이후 원소를 하나씩 접근 하는 과정이 매우 빠름

- 또한 부가적인 메모리가 필요가 없지만, 동일한 정렬 기준을 가진 것의 순서가 다를 수도 있음. (unstable)

5) 기수정렬

- O(nlgn) 시간복잡도 알고리즘 중 하나.

- 데이터의 개수가 많더라도 특정 범위 조건이 존재하는 경우에 한에서 매우 빠른 알고리즘. 그 데이터의 크기를 기준으로 갯수를 새는 것.

- 기존의 알고리즘 방식 (점진적, 분할정복)은 위치 값을 바꿔가며 정렬하는것에 반면, 기수정렬은 크기를 기준으로 데이터를 새주기만 함.

- 차례차례 데이터를 접근만 하기에 시간복잡도는 O(dn) 임. 하지만 매우 특수한 경우에만 사용됨. (d는 가장 큰 데이터 자리수)

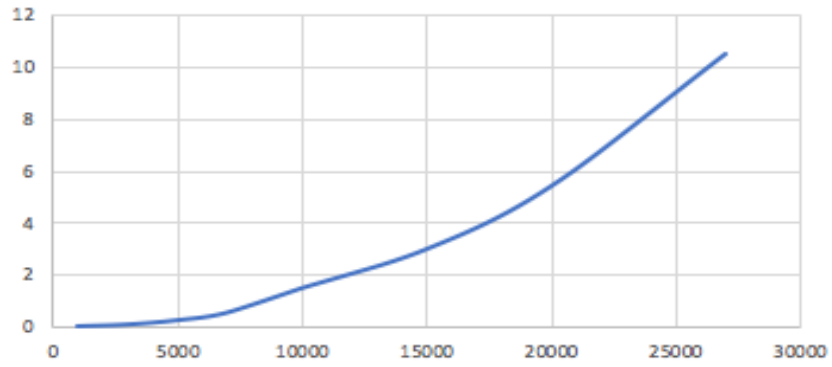
- 2. 알고리즘 성능 분석

- 영화 데이터를 통해 진행, 데이터의 속성은 “ID”, “제목”, “년도”, “장르” 로 구성
- 년도, 제목 순으로 정렬.
- 제목 정렬은 년도가 정렬된 데이터를 갖고 정렬.
- 각 알고리즘의 성능 측정.
- 각 알고리즘의 성능 측정 결과 비교

1) 삽입정렬 - 성능분석

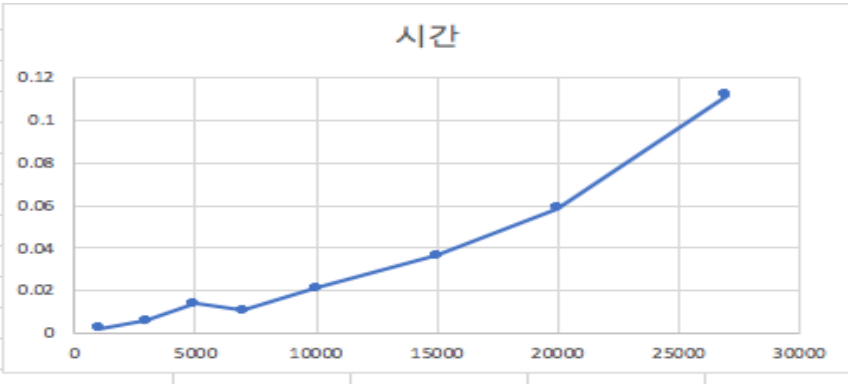
년도 정렬 (x 축 : 데이터 량 , y 축 : 시간 (초))

데이터량	시간
1000	0.015764
3000	0.081946
5000	0.245156
7000	0.538541
10000	1.482132
15000	2.984526
20000	5.409784
27000	10.512957



년도 정렬 후 제목 정렬(x 축 : 데이터 량 , y 축 : 시간 (초))

데이터 수	시간
1000	0.002012
3000	0.005766
5000	0.013834
7000	0.010704
10000	0.020912
15000	0.036309
20000	0.058453
27000	0.111868



- 년도 정렬 시 결과 값을 통해 삽입 정렬 시 초기 데이터가 정렬되지 못했음을 알 수 있음.

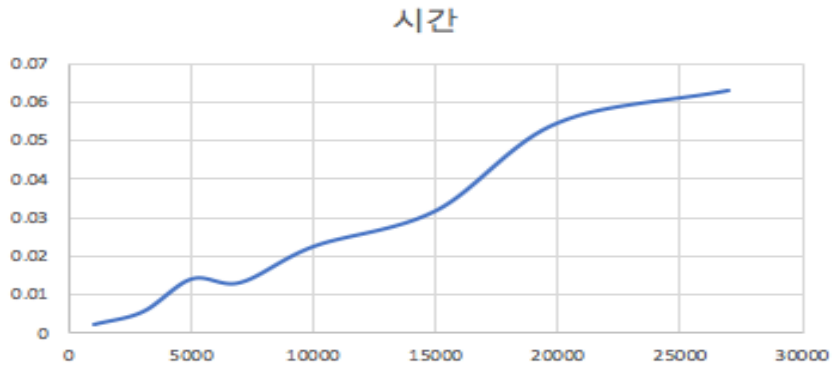
- 점진적 방식인 삽입정렬의 Average-case 는 O(n^2) 으로써 정렬 데이터가 늘어날 수록 시간도 y=n^2 의 형태의 그래프 모양과 비슷하게 그려짐.

- 제목 정렬 결과 값과 년도 정렬 결과 값을 비교 해보면, 제목 정렬이 월등한 성능을 확인할 수 있음. 이는 정렬된 상태에 따라 성능이 달라지는 삽입 정렬의 특징을 확인할 수 있음.

2) 병합정렬 – 성능분석

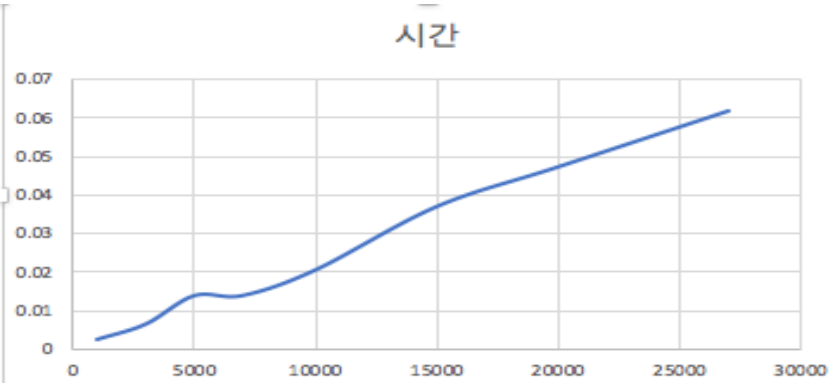
년도 정렬 (x 축 : 데이터 량 , y 축 : 시간 (초))

데이터량	시간
1000	0.002497
3000	0.005811
5000	0.014253
7000	0.013331
10000	0.022649
15000	0.031821
20000	0.054504
27000	0.062888



제목 정렬 (x 축 : 데이터 량 , y 축 : 시간 (초))

데이터 수	시간
1000	0.002568
3000	0.006552
5000	0.013929
7000	0.013974
10000	0.02063
15000	0.037095
20000	0.047383
27000	0.061876



- 제목 정렬의 수행 시간과 년도 정렬의 수행시간이 비슷함. 초기 데이터 상태에 의존하지 않고, O(nlgn) 성능을 보이는 정렬 알고리즘 중 가장 안정적인 결과 값을 확인 할 수 있음.

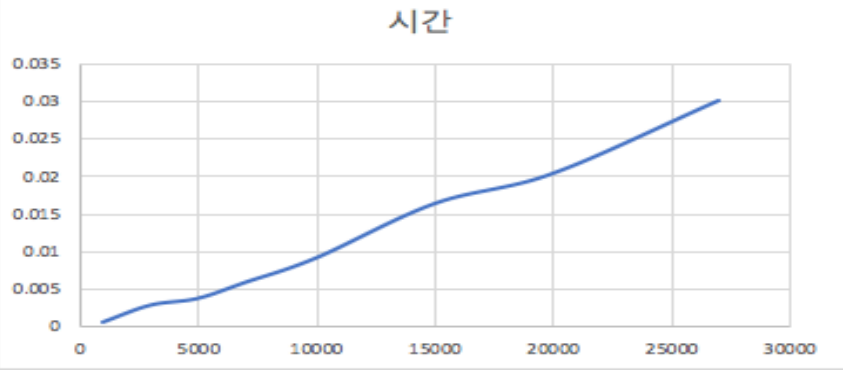
- 년도 정렬과 제목 정렬의 수행 결과를 통해 초기 데이터 상태 값에 영향을 받지 않음을 확인 가능.

- O(nlgn) 정렬 알고리즘 답게 O(n^2) 정렬 알고리즘인 삽입 정렬과의 월등한 성능 차이를 확인 할 수 있음.

3) 퀵 정렬 - 성능분석

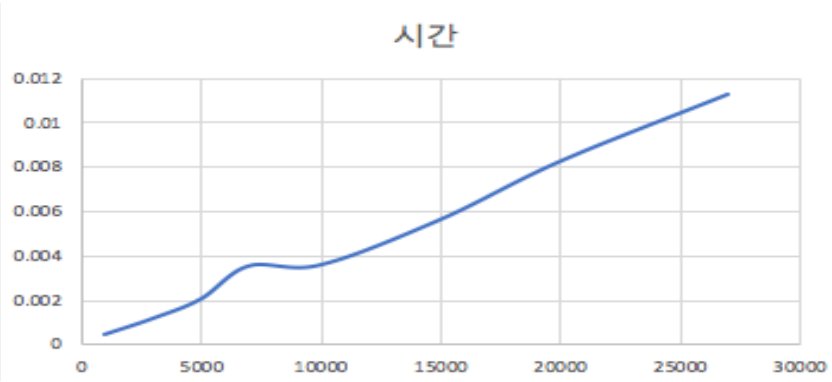
년도 정렬 (x 축 : 데이터 량 , y 축 : 시간 (초))

데이터량	시간
1000	0.000709
3000	0.002957
5000	0.003857
7000	0.005977
10000	0.009235
15000	0.016458
20000	0.020475
27000	0.03016



제목 정렬 (x 축 : 데이터 량 , y 축 : 시간 (초))

데이터량	시간
1000	0.0005
3000	0.001213
5000	0.002078
7000	0.003561
10000	0.003614
15000	0.005643
20000	0.008251
27000	0.011258



- 제목 정렬 시 년도 정렬 보다 성능이 더 좋음. 이를 통해 년도 정렬 프로세스 이후 정돈된 데이터를 갖고 프로세스를 수행 시, 성능이 더 좋음을 알 수 있음.

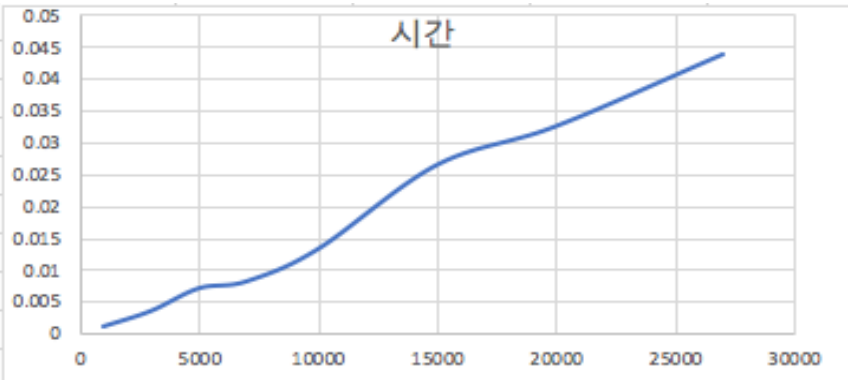
- 또한, 년도 정렬 이후에 제목정렬의 결과 값은 정렬 되어진 데이터에 성능이 좋아지는 퀵 정렬 다운 결과 값임을 확인 가능.

- O(nlgn) 정렬 알고리즘 답게 O(n^2)인 삽입정렬 성능 보다 월등한 성능을 보임.

4) 힙 정렬 – 성능분석

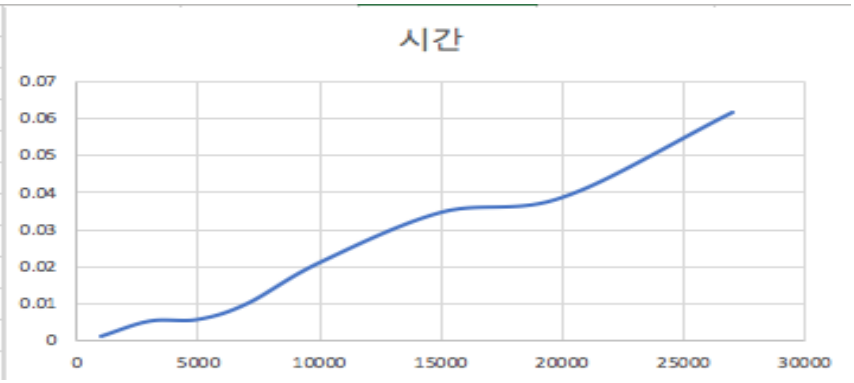
년도 정렬 (x 축 : 데이터 량 , y 축 : 시간 (초))

데이터량	시간
1000	0.001334
3000	0.00377
5000	0.00729
7000	0.00837
10000	0.013409
15000	0.026568
20000	0.032621
27000	0.043839



제목 정렬 (x 축 : 데이터 량 , y 축 : 시간 (초))

데이터량	시간
1000	0.001521
3000	0.005589
5000	0.006041
7000	0.010239
10000	0.021294
15000	0.034766
20000	0.038784
27000	0.061587



- 연산 결과 값을 통해 O(nlgn) 정렬 알고리즘 답게 좋은 성능을 확인할 수 있음.

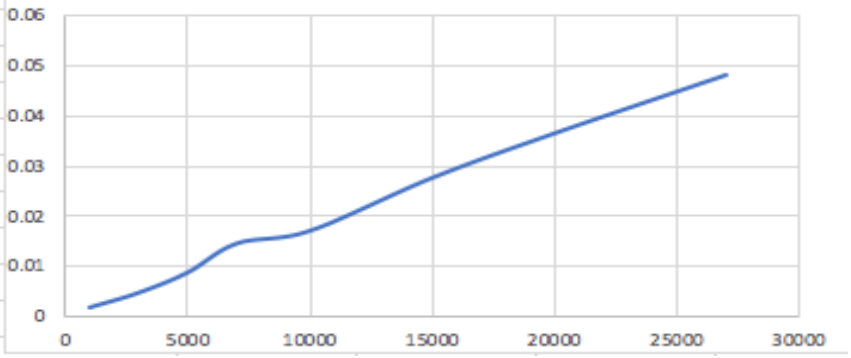
- 년도 정렬 결과 값과 제목 정렬 결과 값 두개를 비교 시 미세한 차이는 존재할 뿐, 결과 값이 비슷하게 나옴. 이를 통해 힙 정렬은 초기 정렬 상태에 영향을 받지 않다는 것을 확인 가능.

- 정렬 데이터의 수와 결과 값의 비율 관계를 통해 Worst, Best, Average 값이 동일하단 사실을 확인할 수 있음. 즉 Worst, Best, Average 값이 O(nlgn)으로 동일한 힙 정렬 결과 값을 확인 가능.

5) 기수 정렬 – 성능 분석

(x 축 : 데이터 량 , y 축 : 시간(초)) – 년도 정렬

데이터량	시간
1000	0.001816
3000	0.004763
5000	0.008803
7000	0.014577
10000	0.017141
15000	0.027724
20000	0.036616
27000	0.048266

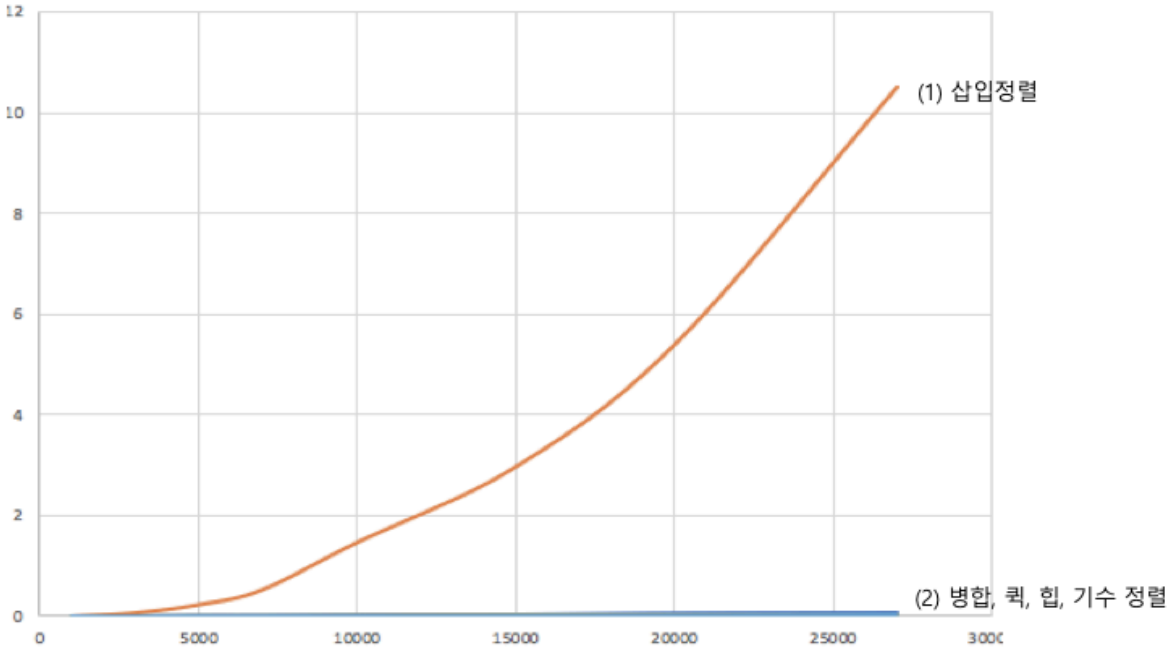


- 연산 결과 값을 통해 O(nlgn) 정렬 알고리즘 답게 좋은 성능을 확인할 수 있음.

6) 각 알고리즘 성능 비교 및 분석

- O(n^2) 와 O(nlgn) 비교

(x 축 : 데이터 량 , y 축 : 시간(초))



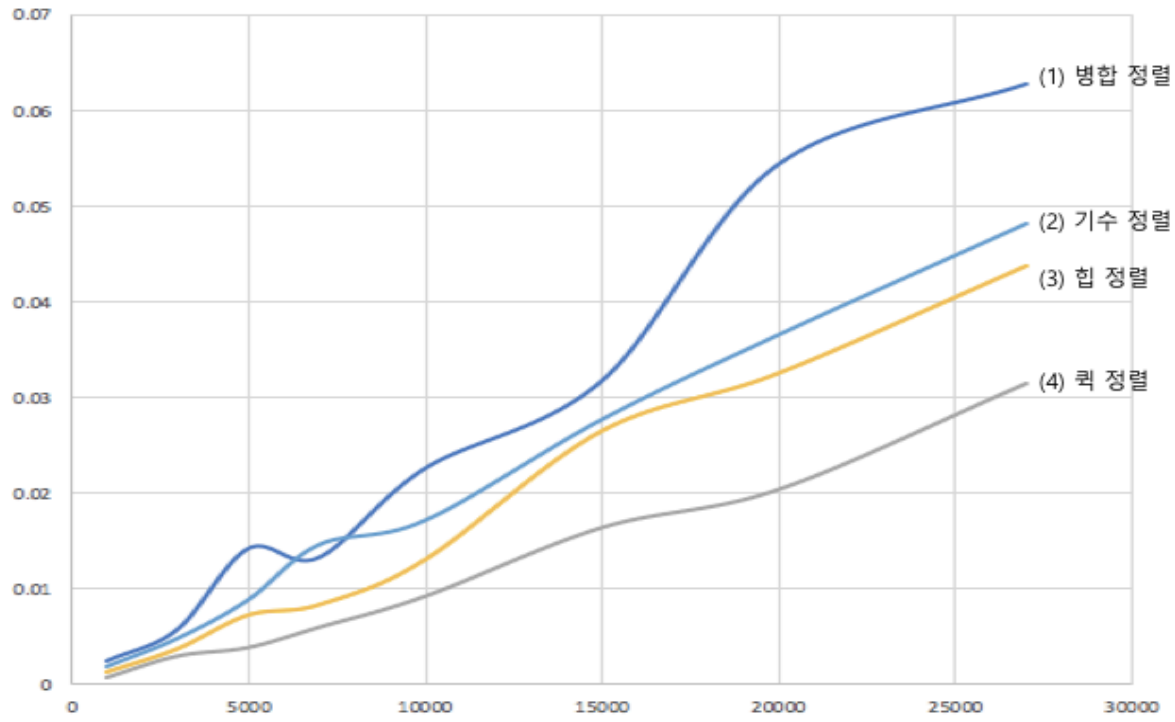
- 정렬 데이터의 수가 적은 경우에는 성능의 차이는 미세함.

- 정렬 데이터의 수가 많은 경우에는 성능의 차이가 월등히 차이남.

- O(n^2) 와 O(nlgn) 의 비교를 통해 O(nlgn) 정렬 알고리즘의 월등한 성능 차이를 알 수 있음

● O(nlgn) 정렬 알고리즘 끼리 비교

(x 축 : 데이터 량 , y 축 : 시간(초))



- 성능이 좋은 순 : 퀵 정렬, 힙 정렬, 기수 정렬, 병합 정렬

- 정렬을 위해 임시 배열을 선언하는 정렬인 병합 정렬과 기수 정렬은 임시 배열을 선언하지 않은 퀵 정렬과 힙 정렬에 비해 성능이 안좋게 나옴.

- 이를 통해 임시 배열을 위해 메모리를 확보하는 과정 또한 프로세스 성능에 영향을 준다는 것을 확인 할 수 있음.

- O(nlgn)의 성능을 보이는 정렬 알고리즘 중 병합정렬이 가장 안정적인 알고리즘임 [2] 하지만 성능이 가장 안좋게 나온

결과를 통해 초기 상태 값이 어느정도 정렬이 되어 있음을 추측 가능.

- 즉, 초기 상태 값이 어느정도 정렬 되어 있었기에 , 퀵 정렬과 힙 정렬의 성능이 초기 상태 값에 영향을 받지 않는 합병정렬과 기수정렬에 비해 성능이 좋게 나왔음을 확인할 수 있음.

- 또한, 퀵 정렬이 가장 성능이 좋게 나온 결과를 통해, 일반적으로 O(nlgn) 의 성능을 보이는 정렬 알고리즘 중 퀵 정렬이 가장 성능이 좋은 것을 확인할 수 있음. [1]

- 힙 정렬의 그래프 모양을 통해 Worst-case, Best-case, Average-case 가 O(nlgn) 같다는 결과를 확인할 수 있음.

3. 결론

- 삽입정렬은 O(n^2) 정렬 알고리즘 중 성능이 좋은 알고리즘.[2] O(nlgn) 정렬 알고리즘에 비해서는 성능이 좋지 못하지만, 코드가 간단함. 또한 어느정도 정렬이 되어 있고, 적은 데이터 량을 정렬시킬 경우 적함.

- 병합정렬은 O(nlgn) 정렬 알고리즘 중 가장 안정적인 알고리즘 초기 데이터의 상태에 의존하지 않고 stable한 상태를 유지 해야하는 경우에 적합한 정렬 알고리즘. 또한 임시 메모리를 고려해야하기 때문에 메모리에 극단적으로 민감한 상황이면 부적합.

- 퀵 정렬은 데이터의 초기 정렬된 상태에 따라 성능의 차이가 있기에, O(nlgn) 정렬 알고리즘 성능 중 극닥적인 경우 가장 성능이 안좋을 수 있음. 하지만 O(nlgn) 정렬 알고리즘 중 일반적으로 가장 빠른 알고리즘[1]

- 힙 정렬은 초기에 힙 구조를 잡는데 시간이 걸림. 힙 구조가 깨진다 하더라도 힙 구조를 유지하는 조치를 수행하는 시간이 짧음. 특정 데이터에 접근하기 편하기에 VIP 관리와 같은 프로그램에 적합한 알고리즘.

4. Reference

[1] https://ko.wikipedia.org/wiki/%ED%80%B5_%EC%A0%95%EB%A0%AC
[2] https://ko.wikipedia.org/wiki/%EC%82%BD%EC%9E%85_%EC%A0%95%EB%A0%AC