알고리즘
프로그래밍 실습 2

인천대학교

알고리즘 | 지도교수 채진석

송병준 | 201701562

알고리즘

프로그래밍 실습 2

송병준 | 201701562
2019년 9월 26일

1. 과제

다음 정렬 알고리즘을 구현 및 실행시간 비교:

- 선택 정렬
- 삽입 정렬
- 버블 정렬
- 쉘 정렬
- 퀵 정렬

2. 환경

운영체제: macOS 10.14.6 (Darwin Kernel Version 18.7.0 x86_64)
편집기: Atom 1.40.1
런타임: Python 2.7.10

3. 수행

3.1 기초 정렬 알고리즘 실행시간 비교

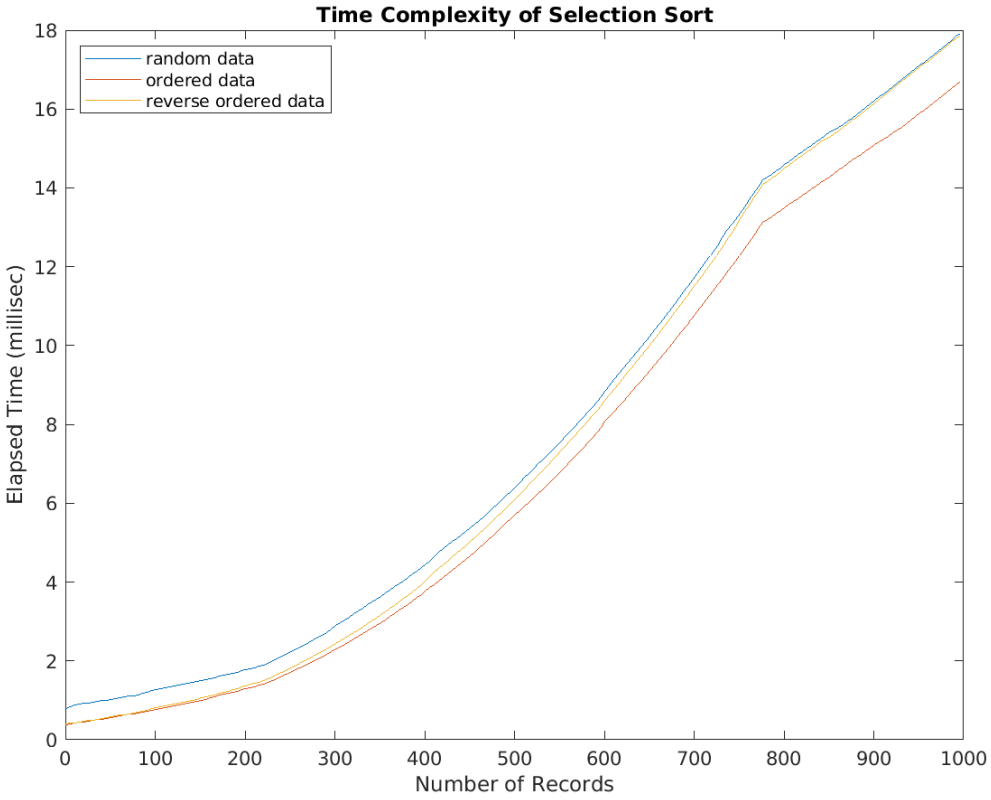
다음 정렬 알고리즘을 Python으로 구현하고, 데이터 크기에 따른 실행 시간을 측정, 비교한다:

- 선택 정렬
- 삽입 정렬
- 버블 정렬
- 쉘 정렬

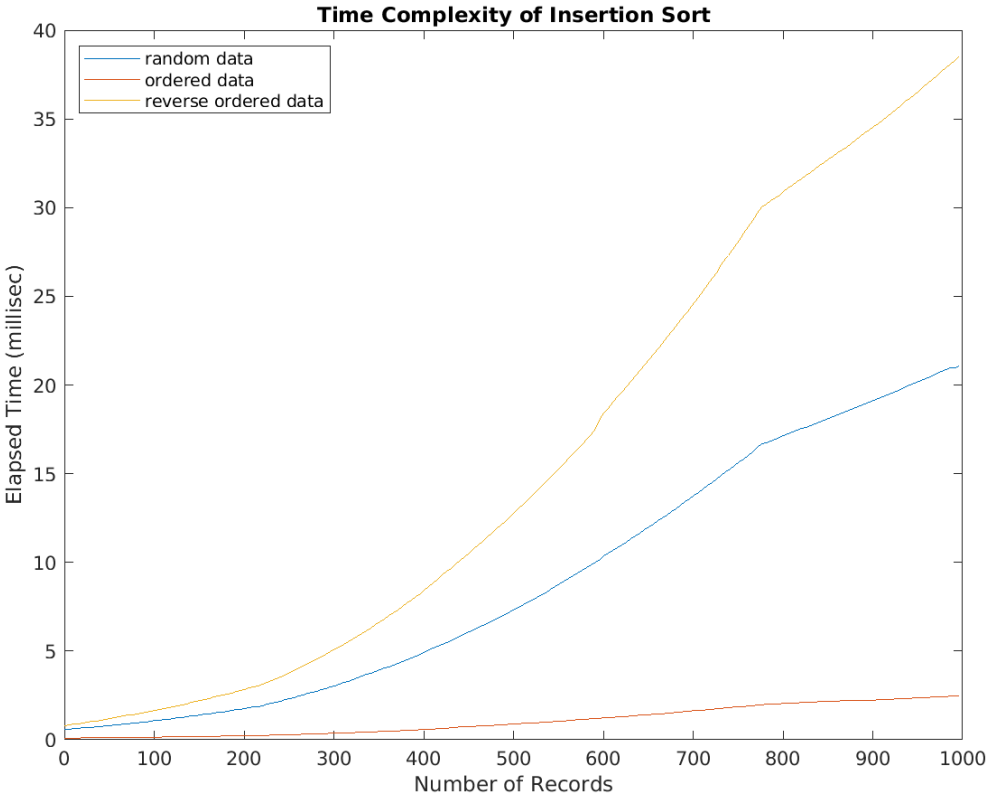
해당 알고리즘을 모두 구현한 뒤 테스트의 결과를 시각화하기 위해 MATLAB 코드를 출력하도록 프로그램을 작성하였다. statistics.py 내의 sorting_test 함수는 이름과 함수, 데이터 수의 범위를 인자로 받아 난수 데이터, 정렬된 데이터, 반대로 정렬된 데이터를 사용하여 실행 시간을 측정한 뒤 해당 결과를 2차원 선 그래프로 나타내는 MATLAB 소스 코드를 출력한다.

해당 함수를 사용해 1개부터 1000개까지의 데이터에 대한 위 알고리즘의 실행 시간을 측정한 결과는 다음과 같다.

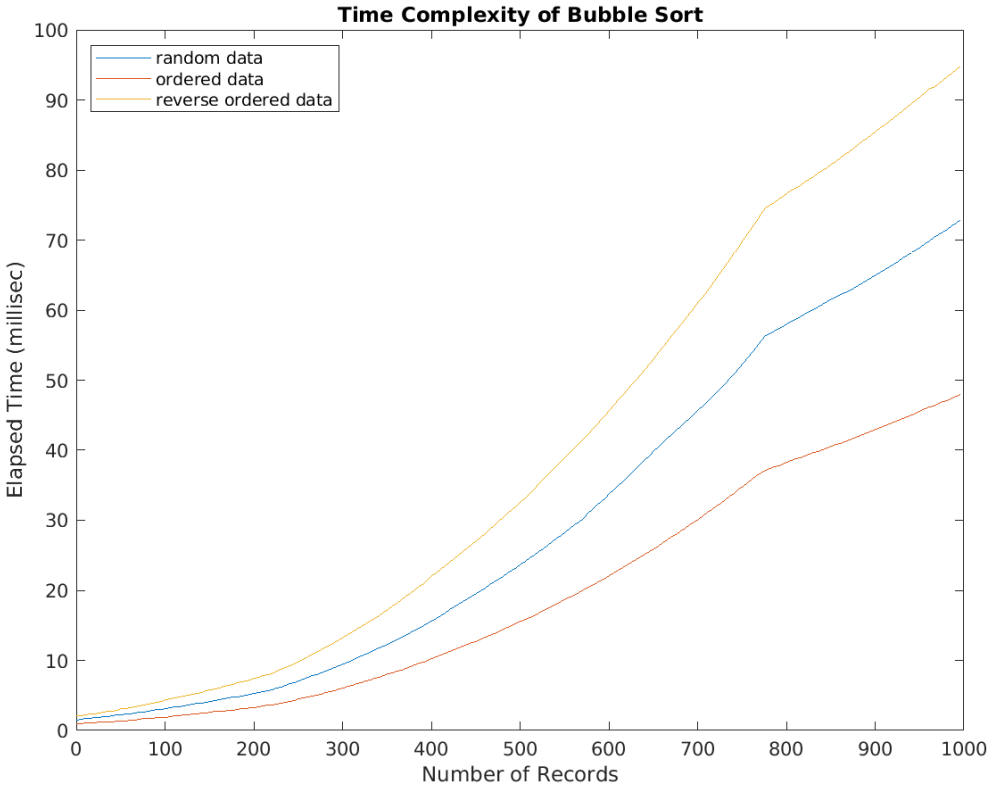
- 선택 정렬


Time Complexity of Selection Sort
random data
ordered data
reverse ordered data
Elapsed Time (millisec)
Number of Records
0
2
4
6
8
10
12
14
16
18
0
100
200
300
400
500
600
700
800
900
1000

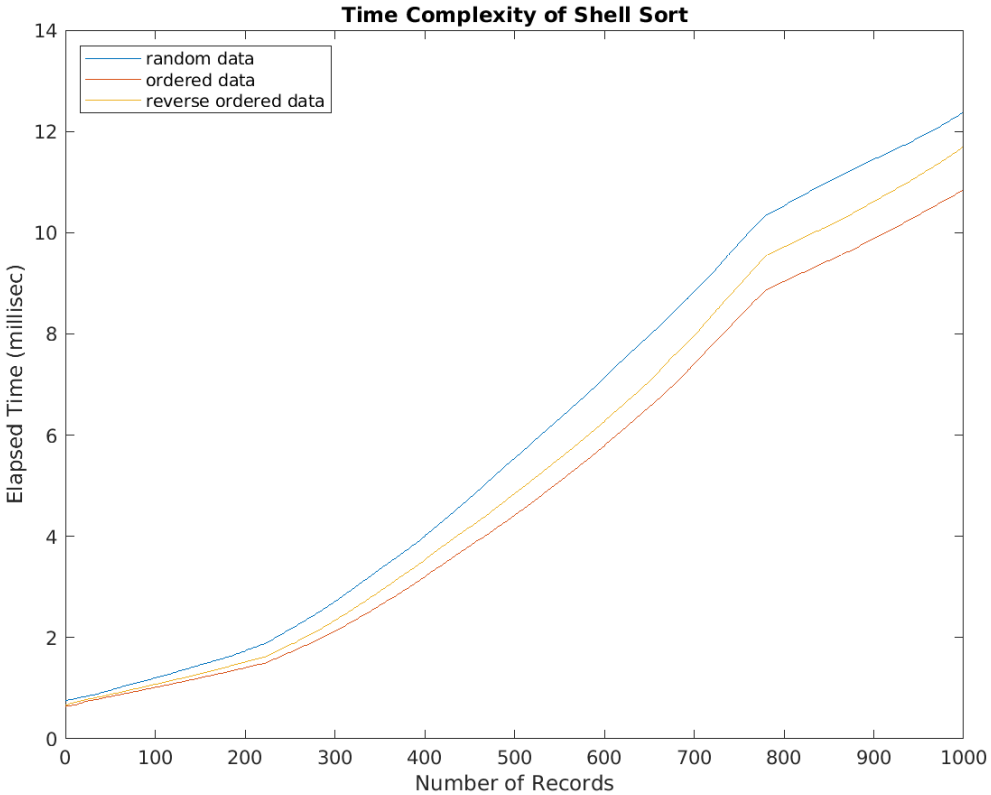
- 삽입 정렬


Time Complexity of Insertion Sort
random data
ordered data
reverse ordered data
Elapsed Time (millisec)
Number of Records
0
5
10
15
20
25
30
35
40
0
100
200
300
400
500
600
700
800
900
1000

- 버블 정렬


Time Complexity of Bubble Sort
random data
ordered data
reverse ordered data
Elapsed Time (millisec)
Number of Records
0
10
20
30
40
50
60
70
80
90
100
0
100
200
300
400
500
600
700
800
900
1000

- 쉘 정렬


Time Complexity of Shell Sort
random data
ordered data
reverse ordered data
Elapsed Time (millisec)
Number of Records
0
2
4
6
8
10
12
14
0
100
200
300
400
500
600
700
800
900
1000

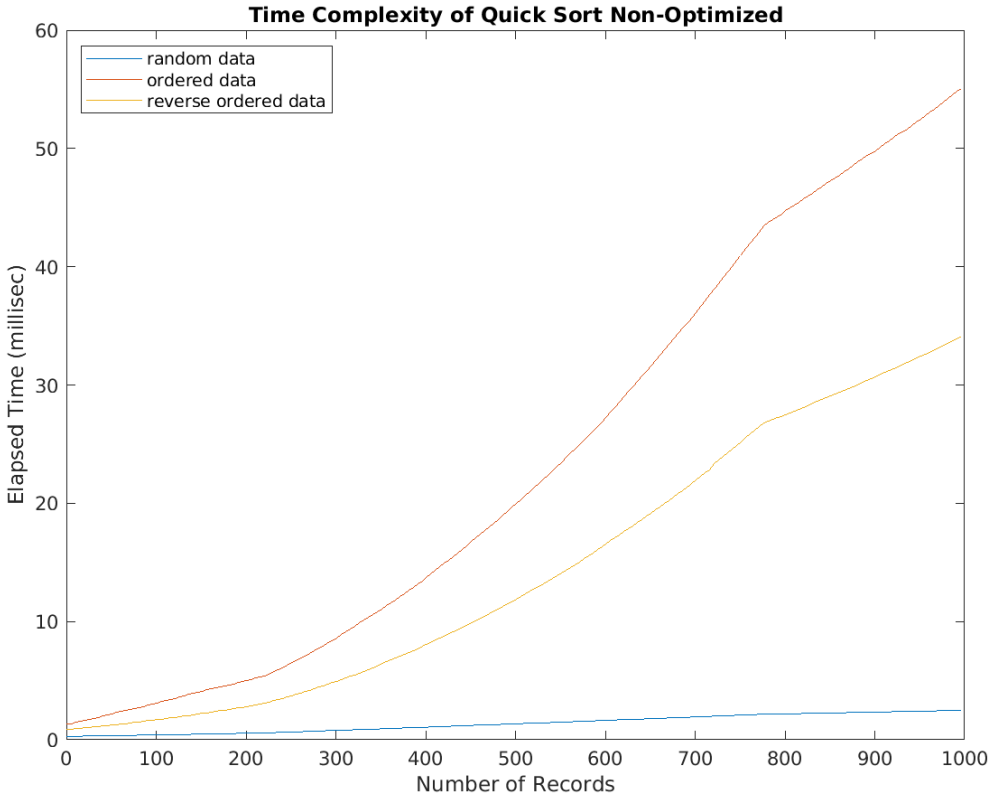
3.2 최적화된 퀵 정렬의 실행시간 비교

일반적인 퀵 정렬 구현에서 pivot은 가장 왼쪽 또는 가장 오른쪽 값으로 설정된다. 이러한 방식은 입력 데이터가 이미 정렬된 경우에 worst case의 시간 복잡도를 유발하게 된다. 이를 해결하기 위해 중간 index에 위치한 값을 최 좌측 또는 최 우측으로 보내도록 한다. 이렇게 보내진 값을 pivot으로 사용하게 되면 거의 혹은 완전히 정렬된 데이터를 다룰 때에 최악의 상황을 면할 수 있다.

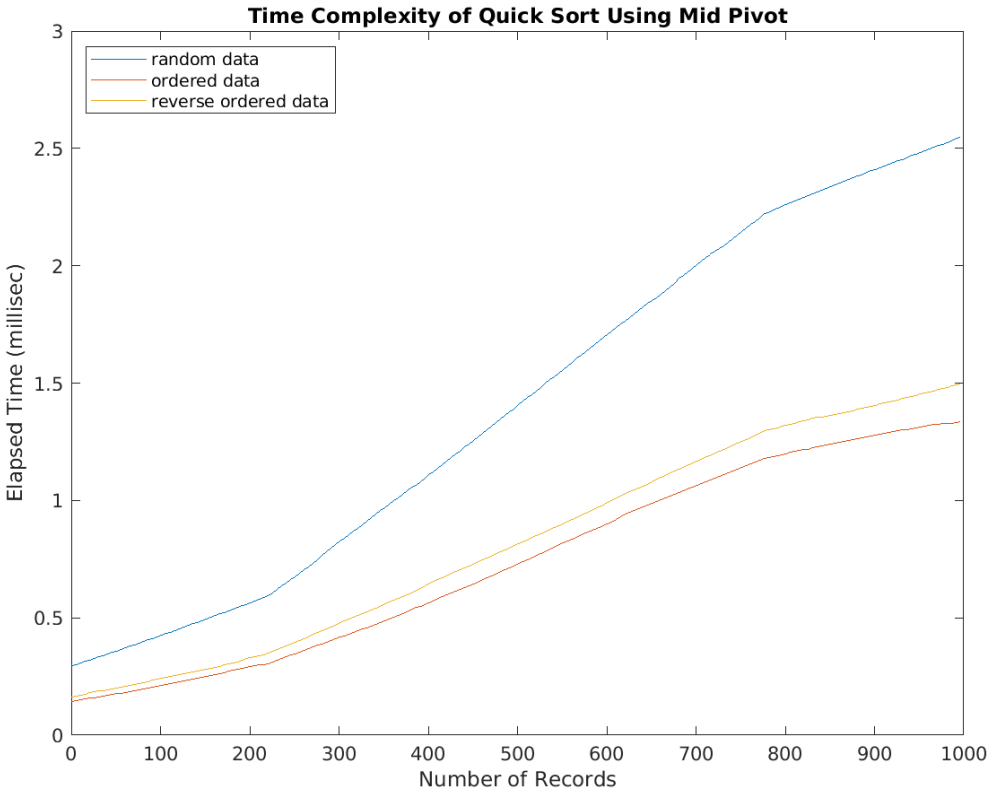
한편 퀵 정렬은 정렬할 원소의 갯수가 적을 때에는 다른 알고리즘에 비해 그리 빠르지 않다. 따라서 partition 후 해당 subarray의 크기가 작을 때에는 삽입 정렬을 사용하면 속도를 개선할 수 있다. 삽입 정렬을 사용하는 것이 빠른 가장 큰 원소의 갯수는 두 알고리즘의 실행 시간의 차이 10회 평균을 기준으로 하여 100회 테스트 평균을 도출한 결과 37이었다.

위 두 최적화를 적용한/적용하지 않은 퀵 정렬 알고리즘의 실행시간은 아래와 같다.

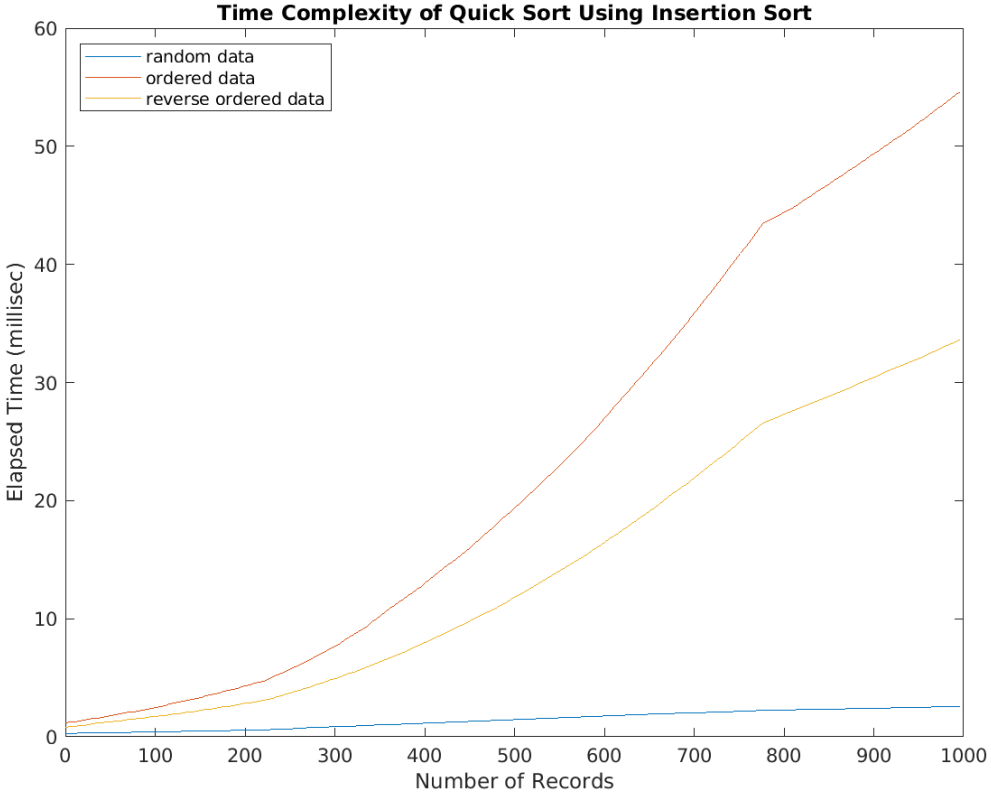
- **가장 비효율적인 퀵정렬**



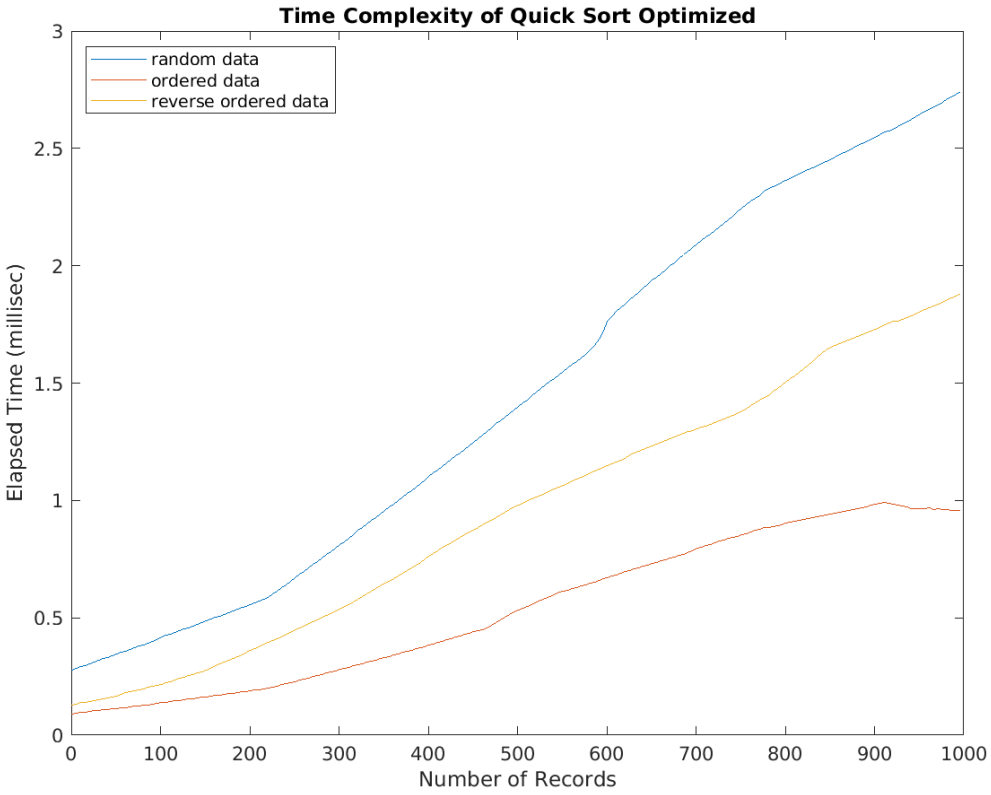
- 중간 원소를 pivot으로 설정한 퀵 정렬



- 작은 파일에 삽입 정렬을 적용한 퀵 정렬



- 모두 적용한 퀵 정렬



4. 결론

선택 정렬은 입력 데이터의 순서와 실행 시간 사이에 연관이 적었다. 삽입 정렬은 정렬된 데이터의 경우 쓰기 연산이 적어지므로 가장 빠르며, 이와 반대되는 역순 정렬 데이터에서 가장 느린 모습을 보였다. 버블 정렬 또한 정렬된 데이터에서 가장 빠르고 역순으로 정렬된 데이터에서 가장 느렸다. 쉘 정렬은 작은 양의 데이터에서는 삽입 정렬보다 느렸지만 큰 데이터에 대해서는 압도적으로 빨랐다.

퀵 정렬은 다른 정렬 알고리즘에 비해 압도적으로 빨랐는데, 이는 데이터의 접근이 지역적으로 일어나 캐싱이 용이하기 때문인 것으로 추정된다. 퀵 정렬의 최대 단점은 정렬된 데이터에서 최악 시간 복잡도를 보인다는 점인데, 이는 pivot을 적절히 설정해 주는 것으로 해결할 수 있다. 측정 결과 완전 정렬 데이터를 정렬시 실행 시간이 약 1/40로 단축되었다.

5. 기타

전체 소스 코드는 GitHub 저장소(https://github.com/potados99/algorithm)에 업로드되어 있습니다.