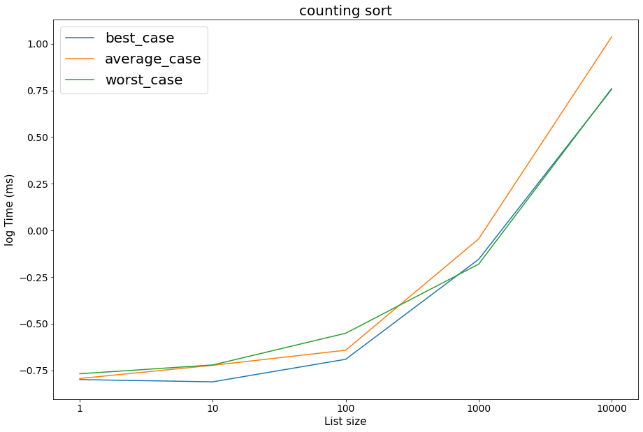
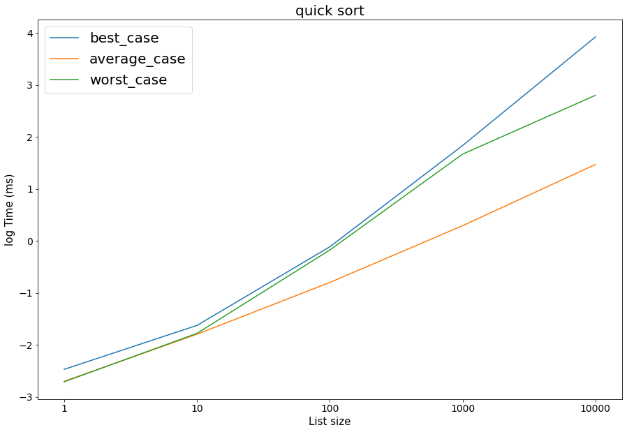
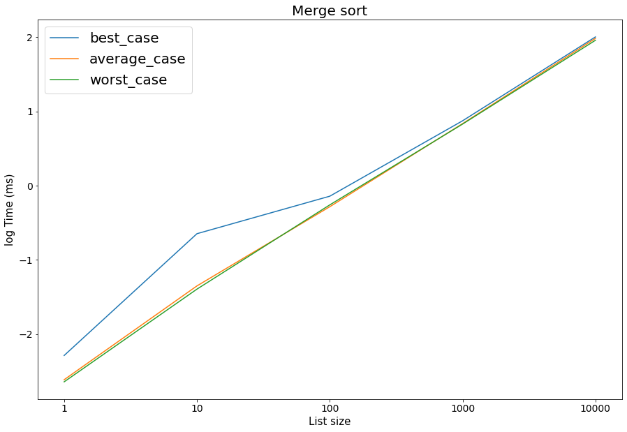
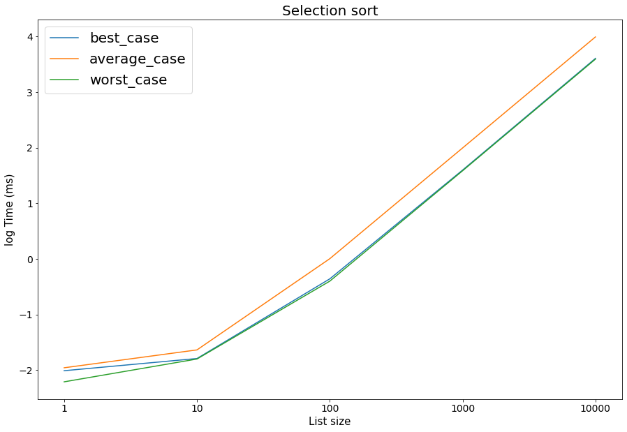
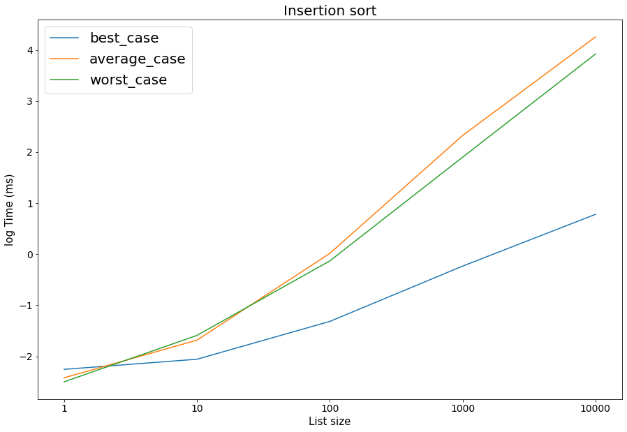
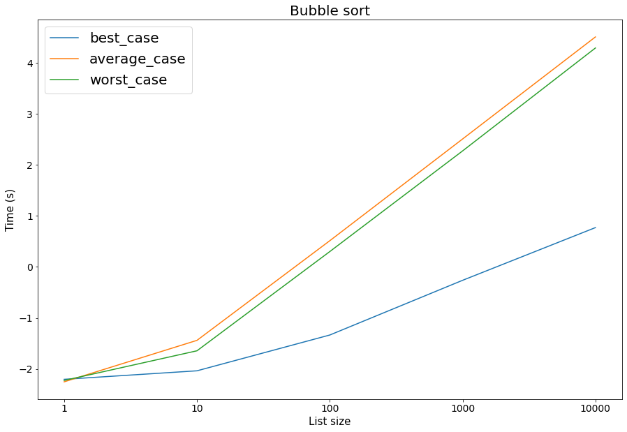
#1

그림 알고리즘 별 수행 시간



Sort 알고리즘의 계산 복잡도 별 알고리즘은 다음 표1과 같다.

표 계산 복잡도 별 알고리즘 분류

|  |  |
| --- | --- |
| **계산 복잡도** | **Sort 알고리즘** |
|  | Bubble sort  Selection sort  Insertion sort (worst case)  Quick sort (worst case) |
|  | Insertion sort (best case) |
|  | Merge sort |

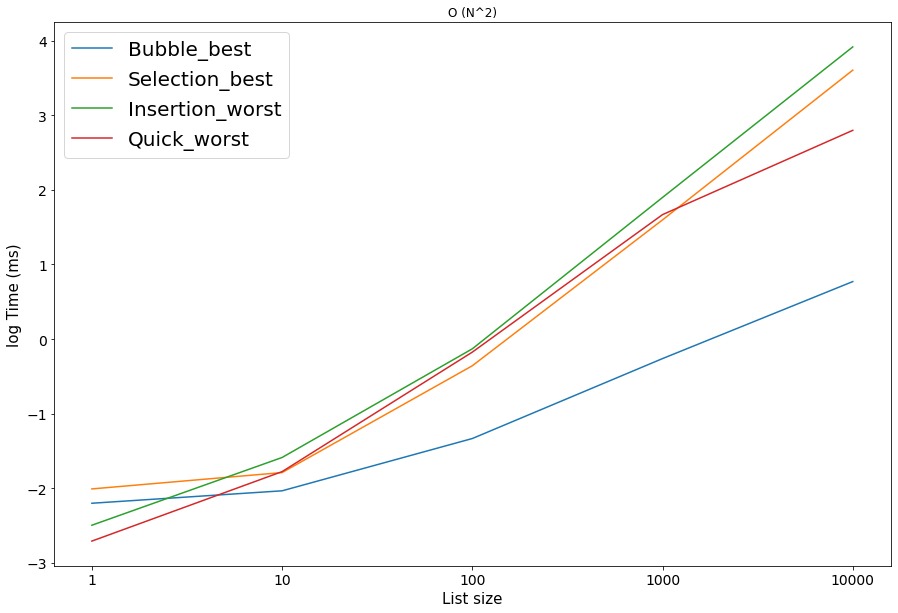
그림 2는 위의 case 중 계산 복잡도가 인 사례를 도시하였다.

그림 O(N^2) 복잡도를 갖는 case

과제의 1번 문항을 통해 big O complexity를 계산하고, 알고리즘 별 best case, average case와 worst case에 대해 실험을 수행했습니다.

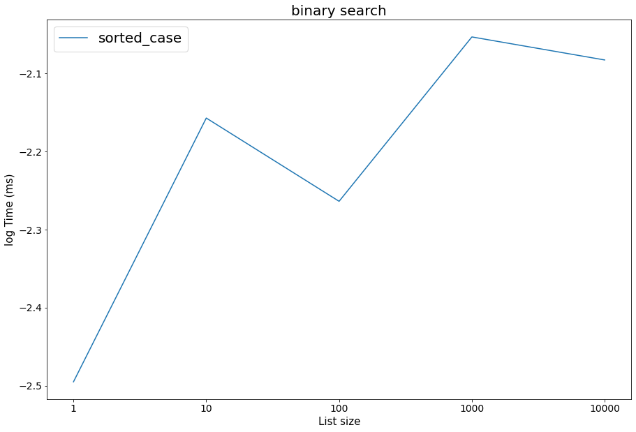
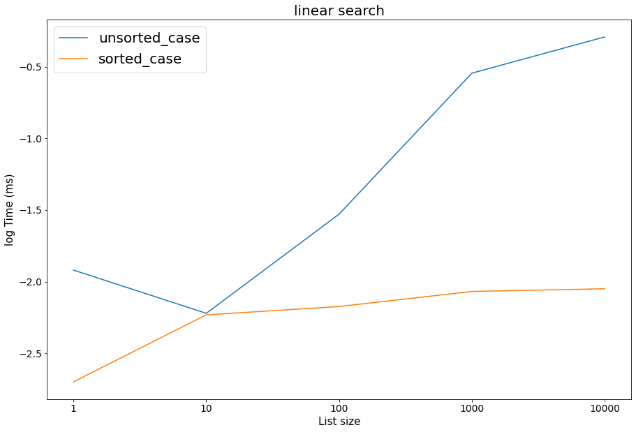
1. Best case와 worst case의 big O complexity가 동일한 bubble sort에서 best case와 worst case의 실제 수행시간의 차이가 매우 크게 발생한 것을 확인했습니다.
2. 그림 2에서도 마찬가지로 동일한 계산 복잡도의 case의 수행시간을 확인했을 때, 큰 차이를 보이는 경우를 확인했습니다.

Big O complexity는 데이터 크기에 따라 연산 수행 횟수의 변화 트렌드는 확인 할 수 있지만, 실제 수행 시간에는 각 연산에 소요되는 시간의 차이와 minor한 term의 영향이 있음을 확인했습니다.

이와 같은 실험에도 환경에 따라 영향을 받기 때문에, 여러 번 반복 실험을 통해 통계적으로 수행 시간을 추정하는 실험을 추가적으로 수행할 필요가 있어보입니다.

#2

그림 Search algorithm 실행시간



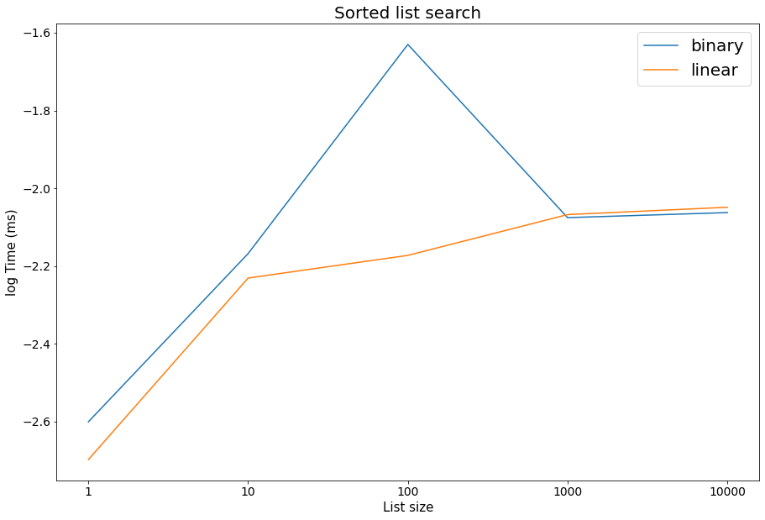
search 알고리즘 중 binary search는 이미 sorted된 list에 대해 탐색을 수행할 수 있기 때문에 sorted case에 대해서만 실험을 수행하였습니다. Linear case에 대해서 데이터의 크기가 어느 정도 큰 상황에서 정렬된 데이터의 소요시간이 압도적으로 적었습니다.

그림 sorted list의 search 알고리즘 실행 시간

정렬된 데이터에 대해 linear search의 복잡도는 이고, binary search의 복잡도는 입니다. 그러나 그림 4의 결과를 보면 100개의 데이터에서 binary search의 소요시간이 binary search에서 크게 소요된 것으로 확인되었고, 그 보다 더 큰 데이터에 대해서는 유사한 시간이 소요된 것을 확인했습니다. 이 이유는 linear search를 구현할 때, search 하고자하는 값을 찾은 후 알고리즘을 종료하는 방식으로 구현했기 때문에, 적은 데이터로 테스트를 수행할 때, 주어진 상황에 따라 소요시간의 편차가 크게 발생했을 것으로 판단됩니다.

#3

OS : Ubuntu 18.04.5 LTS

Cpu : intel Xeon® CPU @ 2.20GHz

Memory : 16GB