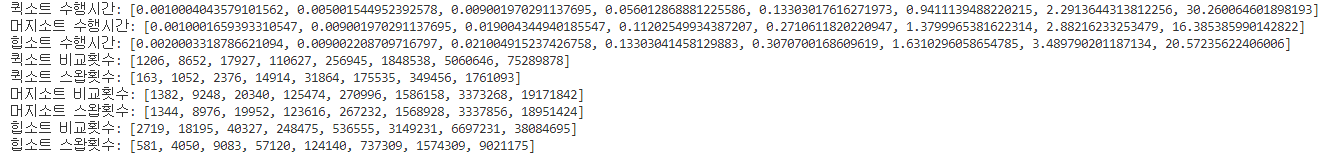
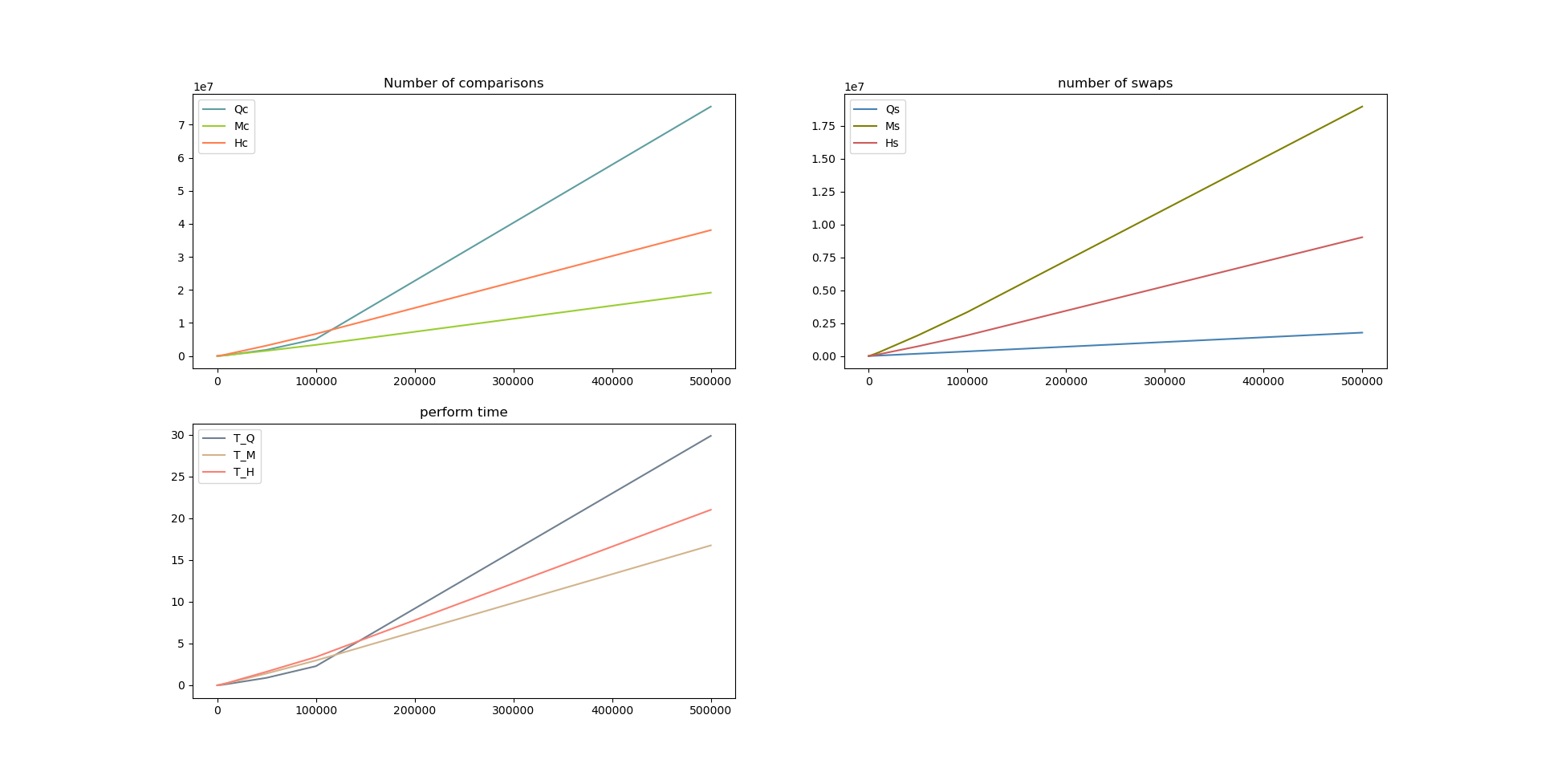
알고리즘 과제 #2

퀵 소트, 머지 소트, 힙 소트에서 비교 횟수와 스왑 횟수를 고려해 코드를 완성한 다음 수행시간을 비교했습니다. N의 값을 100, 500, 1000, 5000, 10000, 50000, 100000, 500000 이렇게 바꿔서 통계하여 그래프를 그렸습니다.





그래프를 비교 횟수, 스왑 횟수, 수행 시간으로 나눴습니다. 먼저 비교 횟수를 나타낸 그래프를 보겠습니다. 보다시피 파란색 선을 가진 퀵소트 비교횟수가 가장 많음을 알 수 있습니다. n값이 100000일 때 까지는 비교 횟수가 머지 소트보다는 느리고 힙 소트보다는 빠르지만 n값이 100000을 넘어가면 힙 소트와 머지 소트보다 비교 횟수가 많아짐을 알 수 있습니다. 퀵 소트에 비해 머지 소트와 힙소트는 어떤 값을 전후로 비교 횟수가 크게 달라지는 게 없다는 것을 알 수 있습니다.

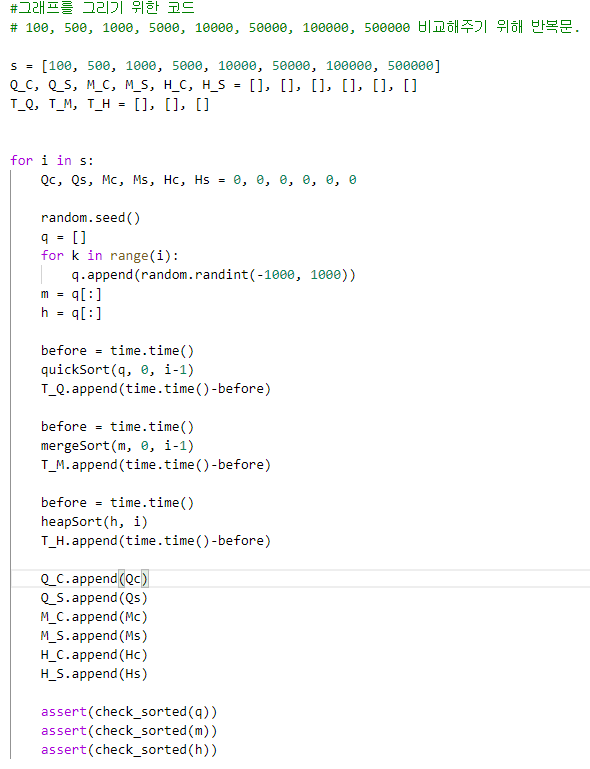
스왑 횟수는 머지 소트 > 힙 소트 > 퀵 소트 순으로 많습니다. 다음으로 수행 시간 그래프를 보겠습니다. 수행 시간 그래프도 비교 횟수 그래프와 비슷하게 퀵 소트가 n 값 100000을 기준으로 그래프의 모양이 크게 변하는 것을 볼 수 있습니다. 하지만 비교 횟수 그래프에서는 퀵 소트가 머지 소트보다 비교 횟수가 많지만 수행시간에서는 퀵소트가 더 빠름을 볼 수 있습니다.

통합적으로 본다면 n값이 100000 이하 일 때는 퀵 소트가 가장 좋은 효율을 보여주고 있음을 알 수 있습니다. 하지만 n값이 100000이 넘어가면 퀵 소트 보다 머지 소트와 힙 소트의 효율이 더 좋음을 알 수 있습니다.

각 정렬 알고리즘의 시간 복잡도를 알아보겠습니다. 퀵 소트는 가장 최선의 경우 O(nlogn), 평균적으로도 O(nlogn)입니다. 하지만 최악의 경우에는 O(n^2) 입니다. 머지 소트는 가장 최선의 경우, 최악의 경우, 평균적으로도 다 O(nlogn)입니다. 마지막으로 힙소트 또한 머지 소트처럼 세 경우에서의 시간복잡도 모두 O(nlogn)입니다.

정렬을 할 때 같은 값을 가진 값들이 정렬 후에 순서가 뒤바뀌게 된다면 그것을 unstable이라고 하며, 순서가 뒤바뀌지 않는다면 stable이라고 한다. 세 가지의 정렬 알고리즘이 stable 한지, unstable 한지 보겠습니다. 퀵 소트는 정렬 후에 같은 값들의 순서가 뒤바뀌어서 unstable 합니다. 머지 소트는 1/2 n을 크기로 하는 두 개의 서브로 쪼개집니다. 이 서브들이 최종적으로 크기가 1이 될 때까지 반복한다. 여기에서는 인덱스가 변화하지 않습니다. 그리하여 머지 소트는 stable 합니다. 힙 소트는 퀵 소트와 마찬가지로 unstable 하다.

그래프를 그리기 위해서 matplotlib를 임포트 했습니다. 그래프 코드는 아래와 같습니다.



세 정렬 알고리즘의 시간복잡도는 평균적으로 모두 O(nlogn)입니다. 하지만 여러가지 값을 넣었을 때 나타나는 그래프의 모양을 보면 똑같은 시간복잡도를 가져도 차이가 있음을 알 수 있습니다. 또한 위에서도 언급했듯이 n 값이 어떤 값 이하이면 머지 소트와 힙 소트보다는 효율이 좋지만 그 이상이면 머지 소트와 힙 소트보다 효율이 좋지 않음을 알 수 있습니다.