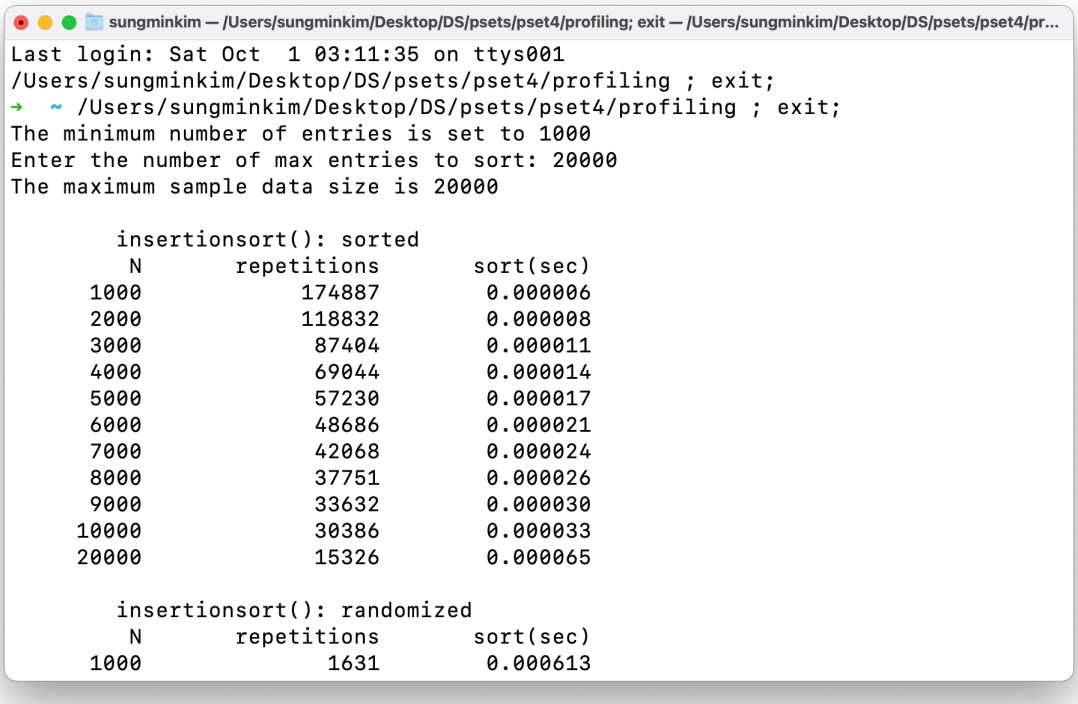
Pset4-Profiling

21900112 김성민

Profiling의 출력 스크린 캡쳐

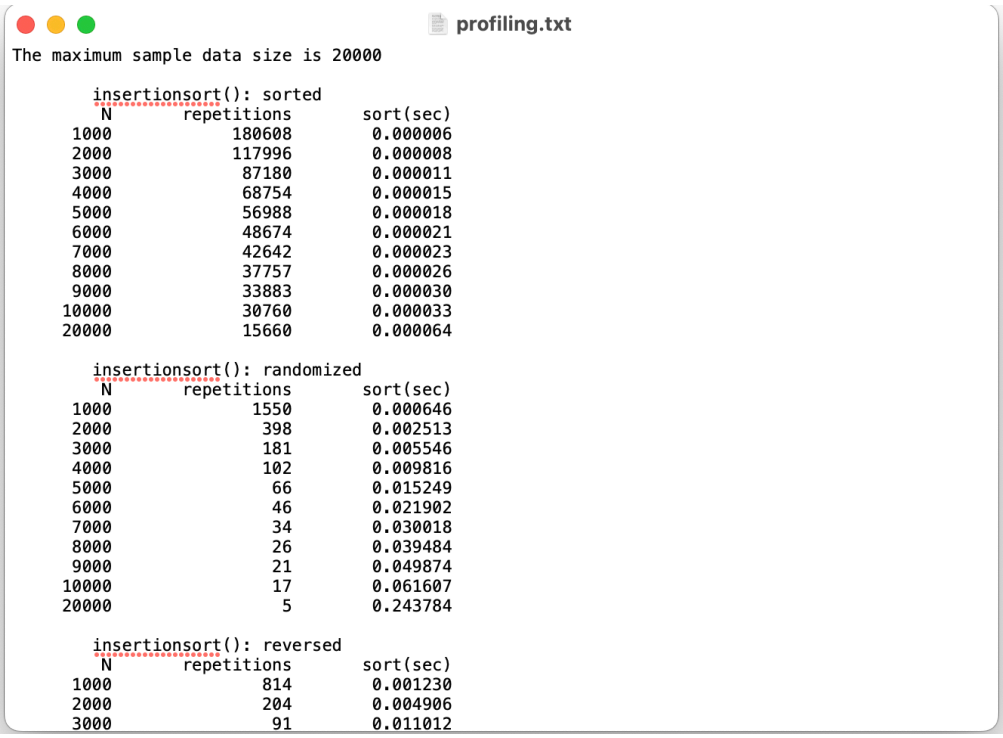


```
sungminkim — /Users/sungminkim/Desktop/DS/psets/pset4/profiling; exit — /Users/sungminkim/Desktop/DS/psets/pset4/pr...
Last login: Sat Oct  1 03:11:35 on ttys001
/Users/sungminkim/Desktop/DS/psets/pset4/profiling ; exit;
➜  ~ /Users/sungminkim/Desktop/DS/psets/pset4/profiling ; exit;
The minimum number of entries is set to 1000
Enter the number of max entries to sort: 20000
The maximum sample data size is 20000

        insertionsort(): sorted
          N       repetitions       sort(sec)
       1000            174887        0.000006
       2000            118832        0.000008
       3000             87404        0.000011
       4000             69044        0.000014
       5000             57230        0.000017
       6000             48686        0.000021
       7000             42068        0.000024
       8000             37751        0.000026
       9000             33632        0.000030
      10000             30386        0.000033
      20000             15326        0.000065

        insertionsort(): randomized
          N       repetitions       sort(sec)
       1000              1631        0.000613
```

위의 실행 결과는 executable file인 profiling을 실행시킨 결과이다. 20000을 입력하여 샘플이 20000개일 경우의 insertion, merge, quick sort의 결과를 sorted, randomized, reversed 순서대로 보여준다.



```
profiling.txt
The maximum sample data size is 20000

        insertionsort(): sorted
          N       repetitions       sort(sec)
       1000            180608        0.000006
       2000            117996        0.000008
       3000             87180        0.000011
       4000             68754        0.000015
       5000             56988        0.000018
       6000             48674        0.000021
       7000             42642        0.000023
       8000             37757        0.000026
       9000             33883        0.000030
      10000             30760        0.000033
      20000             15660        0.000064

        insertionsort(): randomized
          N       repetitions       sort(sec)
       1000              1550        0.000646
       2000               398        0.002513
       3000               181        0.005546
       4000               102        0.009816
       5000                66        0.015249
       6000                46        0.021902
       7000                34        0.030018
       8000                26        0.039484
       9000                21        0.049874
      10000                17        0.061607
      20000                 5        0.243784

        insertionsort(): reversed
          N       repetitions       sort(sec)
       1000               814        0.001230
       2000               204        0.004906
       3000                91        0.011012
```

위의 화면은 ./profiling 20000 > profiling.txt 커맨드를 실행하여 직접 사용자에게 바로 보여주는 것이 아니라 다시 볼 수 있도록 profiling.txt파일에 저장해준 결과이다. 이 profiling.txt파일의 결과값들을 이용하여 다음 과정들을 수행할 것이다.

	$T(N) \approx a\,N^b$, a = 4.99×10^{-9} b = 0.955 insertionsort - Best		$T(N) \approx a\,N^b$, a = 0.713×10^{-9} b = 1.984 insertionsort - Average		$T(N) \approx a\,N^b$, a = 1.22×10^{-9} b = 2.000 insertionsort - Worst	
N	10,000	Time for Million	10,000	Time for Million	10,000	Time for Million
Time	0.000033	Estimated: 0.002679 sec	0.061607	Estimated: 9.52 min	0.122223	Estimated: 20.333 min
N	20,000		20,000		20,000	
Time	0.000064	Measured: 0.004087 sec	0.243784	Measured: 12.978 min	0.488999	Measured: 25.966 min

	$T(N) \approx a\,N^b$, a = 45.13×10^{-9} b = 1.101 Average quicksort **O(N log N): randomized**	
N	10,000	Time for Million
Time	0.001144	Estimated: 0.182165 sec
N	20,000	
Time	0.002455	Measured: 0.223406 sec

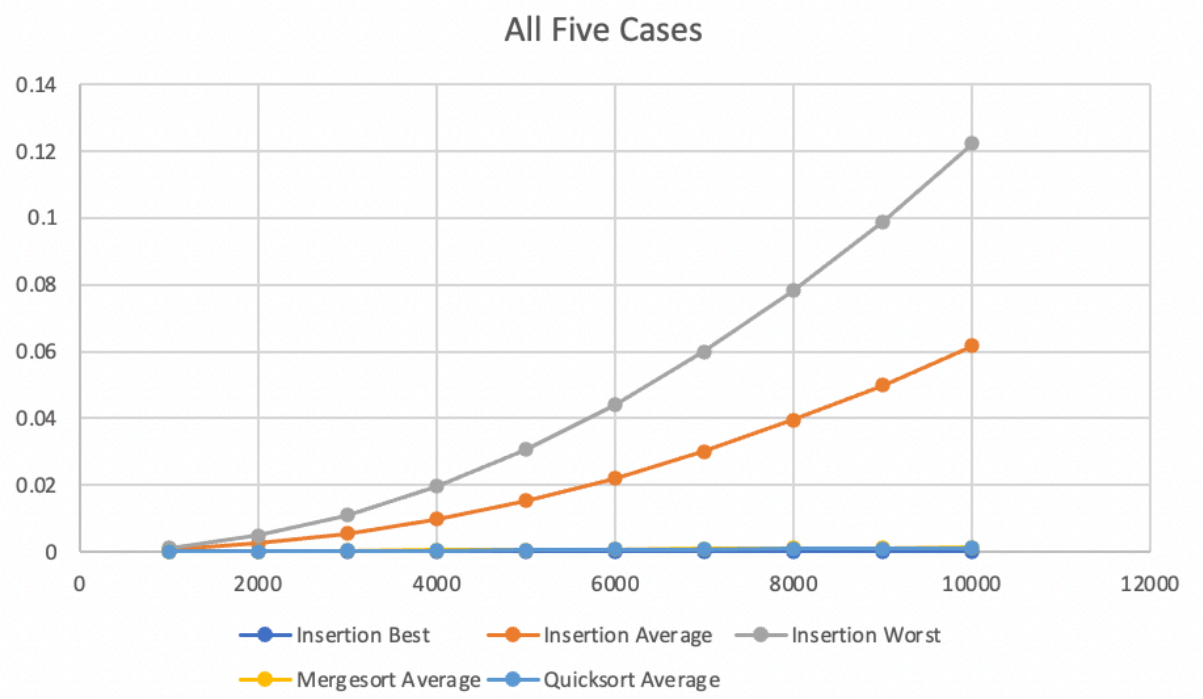
	$T(N) \approx a\,N^b$, a = 45.44×10^{-9} b = 1.117 Average mergesort **O(N log N): randomized**	
N	10,000	Time for Million
Time	0.001335	Estimated: 0.228790 sec
N	20,000	
Time	0.002896	Measured: 0.211921 sec

위에서 소개한 profiling.txt파일을 사용하여 insertion sort의 best, average, worst와 merge sort, quick sort의 average, 총 5가지 case의 시간을 예측하기 위해 a, b를 계산하고 이를 바탕으로 백만개의 샘플에 대한 estimated time을 구하고, 예측 시간을 driver파일을 사용하여 측정한 실제 시간과 비교하는 표를 완성하였다.

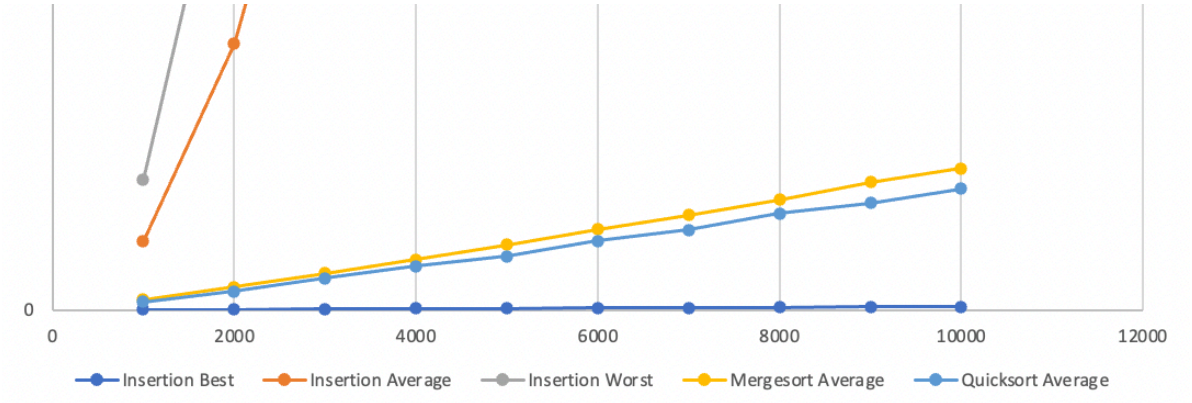
a, b를 계산하기 위해서는 샘플의 개수가 두배 차이나는 두 가지 케이스가 필요한데, 이 두 케이스를 각각 10000인 경우와 20000인 경우로 정하고 해당되는 시간을 profiling.txt에서 찾아서 작성하였다. 먼저, b를 계산하고자 한다. b는 $b = \log_2 \frac{T(2N)}{T(N)}$으로 계산할 수 있는데, 우선 insertion sort의 best case의 b를 먼저 구해보자. $T(10000) = 0.000033$, $T(20000) = 0.000064$임을 이용하여 b를 구하면 $b = \log_2 \frac{0.000064}{0.000033} = 0.955$임을 알 수 있다(유효숫자 3자리 적용). $T(N) \approx a \times N^b$이이므로 b값을 이용하여 a를 구하면 $a = T(N) \div N^b$이므로 $N = 10000$일 경우로 a를 계산하면 $a = 0.000033 \div 10000^{0.955} = 4.99 \times 10^{-9}$임을 알 수 있다. 즉, insertion sort의 best case에서는 $T(N) \approx 4.99 \times 10^{-9} \times N^{0.955}$가 된다. 이를 바탕으로 $N = 1000000$일 경우를 계산하면 0.002679초가 나오게 된다. 이런 식으로 5가지 case 모두 a와 b를 구할 수 있고, $N = 1000000$인 경우의 estimated time 또한 구할 수 있게 된다. 그리고 driver 파일을 이용하여 실제로 각 case를 실행시켰고, 그 결과를 표에 입력하였다. 결과를 보면 완전히 같지는 않고 약간의 오차가 존재하지만 어느 정도 유사성을 보임을 확인할 수 있다.

N	Insertion Best	Insertion Ave	Insertion Wor	Mergesort Av	Quicksort Average
1000	0.000006	0.000646	0.00123	0.0001	0.000077
2000	0.000008	0.002513	0.004906	0.000221	0.000179
3000	0.000011	0.005546	0.011012	0.000348	0.000302
4000	0.000015	0.009816	0.019569	0.000478	0.000413
5000	0.000018	0.015249	0.030564	0.000616	0.000508
6000	0.000021	0.021902	0.044001	0.000761	0.000653
7000	0.000023	0.030018	0.059905	0.000899	0.000755
8000	0.000026	0.039484	0.078199	0.001041	0.000913
9000	0.00003	0.049874	0.098939	0.001206	0.00101
10000	0.000033	0.061607	0.122223	0.001335	0.001144

위 표는 profiling을 실행한 결과를 별도의 추가 command를 통해 profiling.txt파일에 저장한 결과 중 우리가 필요한 데이터만 추출하여 excel에 plot한 표이다. 이 데이터를 바탕으로 다섯 가지 case에 대한 그래프를 그릴 것이다.



위 그래프는 표에 있던 다섯 가지 case의 데이터로 만든 그래프이다. 그래프를 보면 5 case들의 차이점을 시각적으로 더 잘 이해할 수 있다. 우선 insertion의 worst가 가장 시간이 오래 걸리는 것을 볼 수 있고, 그 다음에는 insertion average가 오래 걸림을 볼 수 있다. 나머지 세 경우는 이 상태에서 확인이 어려워서 그래프를 확대하여 보았다.



그래프를 확대하니 나머지 세 경우를 확인할 수 있게 되었다. Insertion best는 거의 x축에 붙어 있음을, 즉 시간 복잡도가 가장 낮다는 사실을 알 수 있다. 나머지 두 경우는 mergesort와 quicksort의 average case인데 이 둘 중에는 mergesort가 더 시간 복잡도가 높다는 것을 알 수 있다.

위와 같은 일련의 과정을 통해 5가지 case들에 대한 결론을 얻을 수 있었다. 이 다섯 경우 중 가장 빠른 경우는 이미 ascending sort되어 있는 sample을 insertion sort하는 경우라는 것을 알 수 있다. 또한 만약 merge, quick, insertion 모두 randomized된 sample을 sort하는 경우라면, quicksort를 사용하는 것이 시간 복잡도가 가장 낮은, 즉 가장 효율적인 방법임을 알 수 있다.