zachary_hightower_433_project_2_report

Write a program to estimate the running time of five sorting algorithms

See packaged source file and class file

Ranking

> Use your program to rank the five sorting algorithms from slowest to fastest for arrays of size 100.
> You will need to use 20000 as the number of arrays to get a reasonable compute time.
> Do the same thing for arrays of size 10000. Report your results. Is the ranking the same in both cases? Would you expect it to be the same for both array size? Explain.

Results Report

Selection
Array Size 100: Average sorting time: 0.010046312499999982 milliseconds

Array Size 10000: Average sorting time: 98.39844199999974 milliseconds

Insertion
Array Size 100: Average sorting time: 0.004192455499999999 milliseconds

Array Size 10000: Average sorting time: 44.36174018503507000154 milliseconds

Bubble
Array Size 100: Average sorting time: 0.016490720450000076 milliseconds

Array Size 10000: Average sorting time: 164.90818668503576000154 milliseconds

Merge
Array Size 100: Average sorting time: 0.005133646200000032 milliseconds

Array Size 10000: Average sorting time: 0.9233013761000044 milliseconds

Quick
Array Size 100: Average sorting time: 0.004477099199999974 milliseconds

Array Size 10000: Average sorting time: 0.875070358149998 milliseconds

Table of Rankings

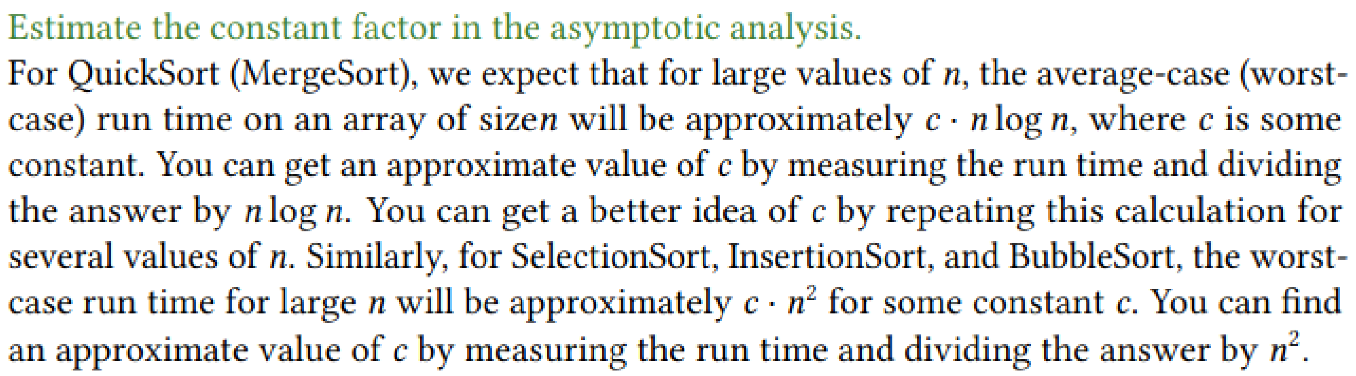
100	10000	
Insertion	Quick	First Place
Quick	Merge	Second Place
Merge	Insertion	Third Place
Selection	Selection	Fourth Place
Bubble	Bubble	Fifth Place

Analysis of Different Array Sizes

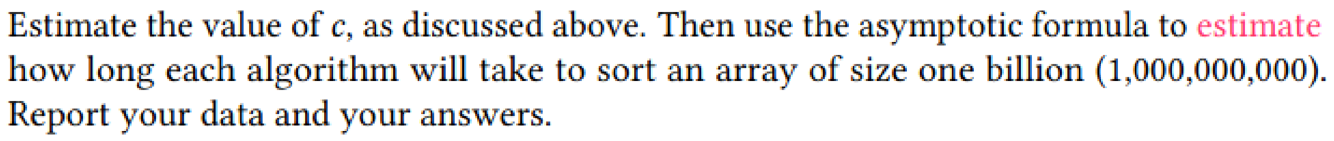
> Is the ranking the same in both cases?
> Would you expect it to be the same for both array size?
> Explain.

ANS: The rankings are not the same. Insertion worked better on the smaller size arrays, but when confronted with the arrays of a larger size its performance dropped heavily. This sort of thing makes sense, as we can expect that on smaller size arrays, the sorting methods might be comparable, but as we go higher and higher in array size, Merge and Quick sort will both perform better by about two orders of magnitude, when compared to Insertion, Selection, and Bubble sorts, within the constraints we examined.

Constant factor Estimation and Asymptotic Analysis



Estimate the constant factor in the asymptotic analysis.
For QuickSort (MergeSort), we expect that for large values of n, the average-case (worst-case) run time on an array of size$n$ will be approximately $c \cdot n \log n$, where c is some constant. You can get an approximate value of c by measuring the run time and dividing the answer by $n \log n$. You can get a better idea of c by repeating this calculation for several values of n. Similarly, for SelectionSort, InsertionSort, and BubbleSort, the worst-case run time for large n will be approximately $c \cdot n^2$ for some constant c. You can find an approximate value of c by measuring the run time and dividing the answer by n^2.



Estimate the value of c, as discussed above. Then use the asymptotic formula to estimate how long each algorithm will take to sort an array of size one billion (1,000,000,000). Report your data and your answers.

Quick Merge

$$c \times n \, log \, n$$

We want to find c so we take in the result we achieved in Quicksort 0.004477099199999974 and we divide this run time by $n\ log\ n$

$$\frac{0.004477099199999974}{100 log 100} = c = 0.000022385$$

We can then check the work by using this formula

$$(0.000022385)\,(10000 \cdot\ log 10000) = 0.8954$$

From this, we can see that the constant factor is fairly accurate. We can then perform the same calculation to find a few separate constants

$$\frac{0.000020538 + 0.00002045 + 0.0000212 + 0.0000232 + 0.000022385}{5} = 0.0000215546$$

Which when used in the above calculation to find the time that the operation should take, yields $(0.0000215546)\,(10000 \cdot\ log 10000) = 0.862184$
Which appears to be a very good approximation of the time

Selection Insertion Bubble

$$c \times n^2$$

We want to find c so we take in the result we achieved in selection sort and divide it by the run time.

$$\frac{0.010046312499999982}{100^2} = 0.0000010046312499999982$$

We can then check the work by using this formula

$$(0.0000010046312499999982)(10000^2) = 100.463125$$

From this we can see that the constant factor is fairly accurate. We can then perform the calculations to find a few separate constants and find a tighter bound around the constant for these three sorts.

$$\frac{0.0000009839844199999974+0.0000010046312499999982+0.0000009730641900000107+0.0000009690763900000038+0.0000009801689249999976}{5}$$
$$= 0.000000982185035000001 54$$

Asymptotic analysis

As we move into higher and higher values of n the constant factor ceases to be of a real concern. The dominant factor in the growth of our analysis will be the term to the right, either $n\ log\ n$ in the case of Merge and Quick sort, or it will be n^2 in the case of bubble, insertion, and selection.

So if we want to analyze how the behavior increases as it grows, we can examine a performance ratio.

Quick sort

$$\frac{(0.000022385)\,(1000000000 \cdot\ log 1000000000)}{(0.000022385)\,(10000 \cdot\ log 10000)} = \frac{(1000000000 \cdot\ log 1000000000)}{(10000 \cdot\ log 10000)} = 225000$$

The constant factors cancel out and we are left with the performance of our $n\ log\ n$ terms. Now that we have the integer number from the performance ratio, we can simply look at the time in milliseconds that it took our Quick sort to run, and multiply that with the result of our previous formula, to get the answer of how long it would take for a 1 billion size array in quicksort.

$$225000 \times 0.875070358149998 = 196890.83058374955\ milliseconds$$

Merge Sort

For merge sort, we don't really need to repeat all the calculations, we can simply take the number given by our above performance ratio, since the growth of the two should be the same, and use that to estimate the time length of merge sort when used on a 1 billion size array.

$$225000 \times 0.9233013761000044 = 207742.80962250099\ milliseconds$$

Selection

$$\frac{(0.0000010046312499999982)(1000000000^2)}{(0.0000010046312499999982)(10000^2)} = \frac{(1000000000^2)}{(10000^2)} = 10000000000$$

$$10000000000 \times 98.39844199999974 = 983984419999.9974\ milliseconds$$

Bubble

$$10000000000 \times 164.9081866850357600154 = 1649081866850.3576000154\ milliseconds$$

Insertion

$$10000000000 \times 44.3617401850350700154 = 443617401850.3507000154\ milliseconds$$