COMP20010 Lab Six: Algorithm Analysis

James Peach

November 14, 2014

1 Part 1 Algorithm

1.1 Asymptotic run-time analysis

My algorithm runs in asymptotic time $O(n * log(n)$. The argument for this follows:

My algorithm has three parts:

1. load data into array
2. sort array
3. find 90t́h percentile value

The loading of the data from file $O(n)$.
Sorting the array with stdlib quicksort $O(n * log(n))$.
Finding the 90t́h percentile value $O(1)$.
this suggests that the dominating factor is the quicksort.

1.2 Experiments

The description of the experiments goes here:

I tested the algorithm with increasing number of n from 1 to $10,000,000$ in powers of 10.

1.3 Prediction

My estimate of the equation for the run-time of the algorithm is:

$$t(N) = 2.4 \times 10^{-8} * log(n) \tag{1}$$

Using this, the estimated time to find the ninetieth percentile of a file containing 60 million numbers is 25.79 seconds.

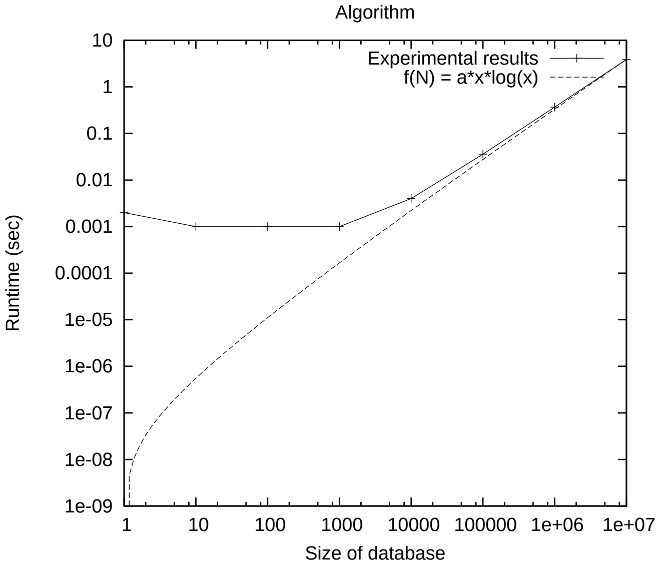

Figure 1: Real time vs count of numbers

2 Part 2 Algorithm

2.1 Asymptotic run-time analysis

My algorithm runs in asymptotic time $O(n)$.

The argument for this follows:

My algorithm has three parts:

1. load data into array
2. sort array
3. find 90t́h percentile value

The loading of the data from file $O(n)$.
Sorting the array with bucket sort $O(k + k)$.
Finding the 90t́h percentile value $O(1)$.
This suggests that the dominating factor is the sort.

2.2 Experiments

The description of the experiments goes here: same as above:

I tested the algorithm with increasing number of n from 1 to $10,000,000$ in powers of 10.

2.3 Prediction

My estimate of the equation for the run-time of the algorithm is:

$$t(N) = 1.46 * 10^{-7} * n \tag{2}$$

Using this, the estimated time to find the ninetieth percentile of a file containing 60 million numbers is 12.8 seconds.

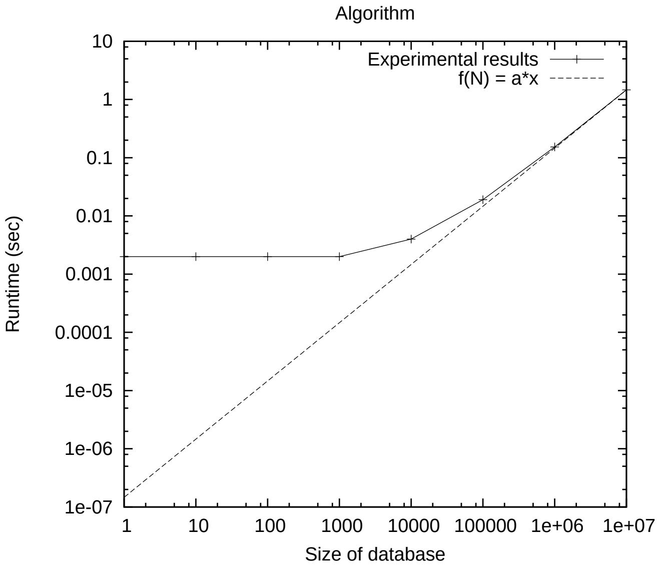

Figure 2: Real time vs count of numbers