Andy Gregoire

Mike McCall

CIS 575

Assignment 5

Code submitted separately.

## Methods and Results

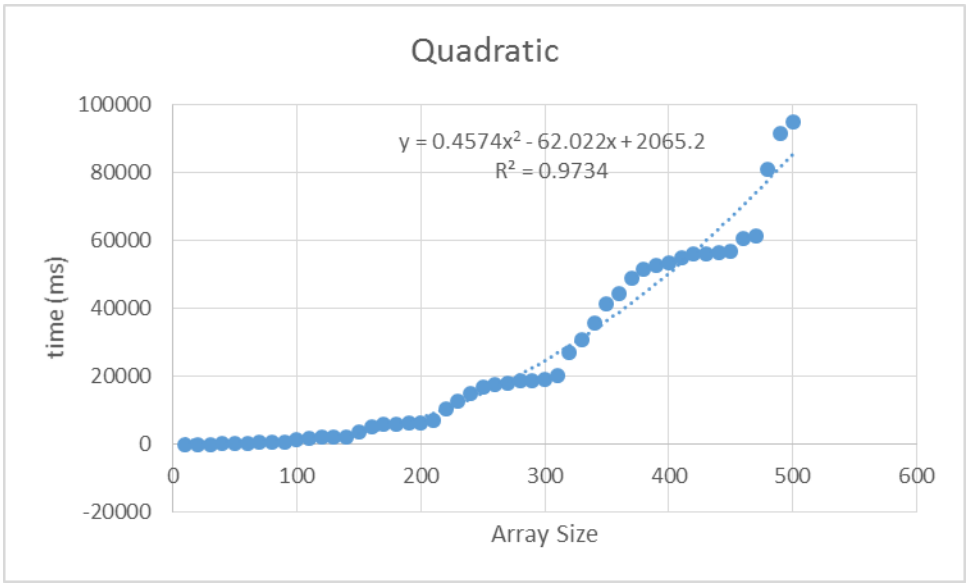
We timed sorting arrays 500 times incrementing array size by 10 from 10 to 500. Each sort at a certain array size was executed on random values. The time only accounts for the actual sorting, not the population of arrays. For example, the tenth row of the table below denotes that it took 1230 milliseconds to sort 500 arrays of size 100 with newly generated random values for each array. This randomization helps keep results close to average runtimes.

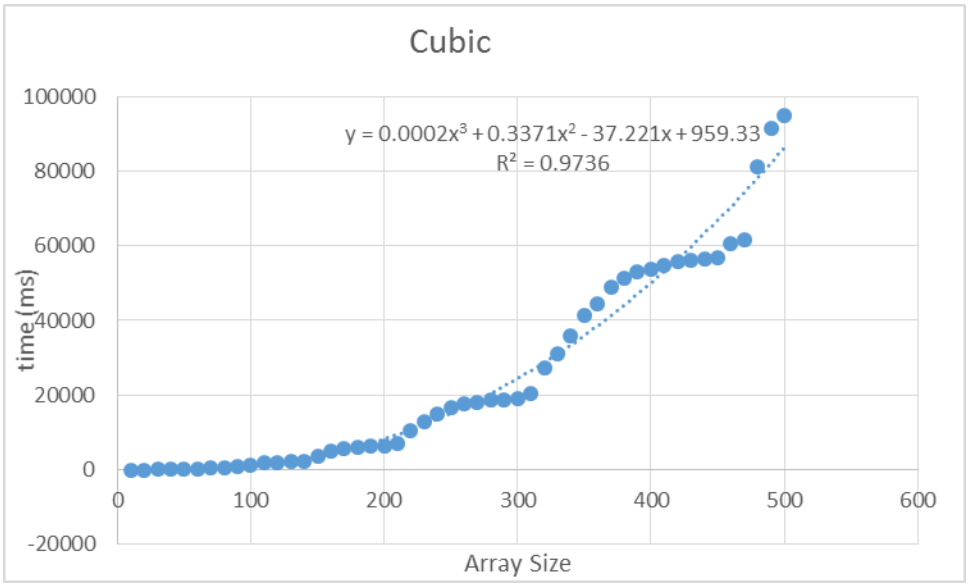
| Array Length | Run time (ms) |
|---|---|
| 10 | 0 |
| 20 | 20 |
| 30 | 50 |
| 40 | 80 |
| 50 | 210 |
| 60 | 250 |
| 70 | 540 |
| 80 | 680 |
| 90 | 720 |
| 100 | 1230 |
| 110 | 1840 |
| 120 | 2020 |
| 130 | 2110 |
| 140 | 2330 |
| 150 | 3670 |
| 160 | 5000 |
| 170 | 5800 |
| 180 | 6020 |
| 190 | 6280 |
| 200 | 6360 |
| 210 | 6920 |
| 220 | 10370 |
| 230 | 12720 |
| 240 | 14900 |

| | |
|---|---|
| 250 | 16760 |
| 260 | 17700 |
| 270 | 17980 |
| 280 | 18730 |
| 290 | 18870 |
| 300 | 19000 |
| 310 | 20380 |
| 320 | 27130 |
| 330 | 30990 |
| 340 | 35860 |
| 350 | 41190 |
| 360 | 44490 |
| 370 | 48900 |
| 380 | 51430 |
| 390 | 52870 |
| 400 | 53620 |
| 410 | 54880 |
| 420 | 55920 |
| 430 | 56170 |
| 440 | 56500 |
| 450 | 56660 |
| 460 | 60610 |
| 470 | 61550 |
| 480 | 81140 |
| 490 | 91590 |
| 500 | **94990** |

Analysis

We did our regression analysis using Excel, so we were limited to whole number polynomial regression. We did best fit lines of order 2 and 3. The cubic polynomial had a slightly better fit, based on the R^2 value given. This correlates with expected results, because the theoretical k is log(1.5, 3) = 2.7.



Floors <-> Ceilings

If we swap floors with ceilings and ceilings with floors, the array will not sort completely. Consider the array A = {5, 2, 6, 3}.

The first recursive call will sort the first two elements. The second recursive call will sort the last two elements. The third will do the first two again. The lack of overlap leaves the sorting incomplete. The array ends up as A = {2, 5, 3, 6}.