|  |  |
| --- | --- |
| Array Size | Comparisons |
| 10 | 63 |
| 50 | 2303 |
| 100 | 9603 |
| 150 | 21903 |
| 200 | 39203 |
| 550 | 300303 |
| 750 | 559503 |
| 900 | 806403 |
| 1000 | 996003 |
| 1500 | 2244003 |
| 3000 | 8988003 |
| 4000 | 15984003 |

**Bubble Sort:**

* Complexity: O(n^2)
  + Goes through every unsorted element, which starts at x, x times
  + Use of a nested for loop in algorithm creates a squared term
* Predictive Equation: x^2 – 4x + 3, where x is array size

|  |  |  |  |
| --- | --- | --- | --- |
| Array Size | Comparisons | Prediction | Difference |
| 492 | 240099 | 240099 | 0 |
| 4008 | 16048035 | 16048035 | 0 |
| 4796 | 22982435 | 22982435 | 0 |
| 277 | 75624 | 75624 | 0 |
| 6607 | 43626024 | 43626024 | 0 |
| 908 | 820835 | 820835 | 0 |
| 2569 | 6589488 | 6589488 | 0 |
| 5038 | 25361295 | 25361295 | 0 |
| 6160 | 37920963 | 37920963 | 0 |
| 565 | 316968 | 316968 | 0 |
| 6686 | 44675855 | 44675855 | 0 |
| 5105 | 26040608 | 26040608 | 0 |
| 1693 | 2859480 | 2859480 | 0 |
| 5987 | 35820224 | 35820224 | 0 |
| 5461 | 29800680 | 29800680 | 0 |
| 1441 | 2070720 | 2070720 | 0 |
| 4619 | 21316688 | 21316688 | 0 |
| 1647 | 2706024 | 2706024 | 0 |
| 6062 | 36723599 | 36723599 | 0 |
| 3815 | 14538968 | 14538968 | 0 |

**Data used to create predictive equation:**

**Predictions using equation from line of best fit:**

**Best case:**

* x = 1: no comparisons
* x = 2: one comparison

**Average Case:**

* x^2 comparisons

**Summary:**

* The bubble sort works well for smaller array sizes but is expensive when used on a large scale. It is only efficient on array sizes from 1 - 80,000. Because it needs to compare every two numbers every time it loops through, the process would take an impractical amount of time for sizes larger than 100,000. For reference, the quicksort would take less than twice the time the bubble sort takes to sort an array length 100 times as large.