Search/Sort Writeup

Jamie Moseley 2017

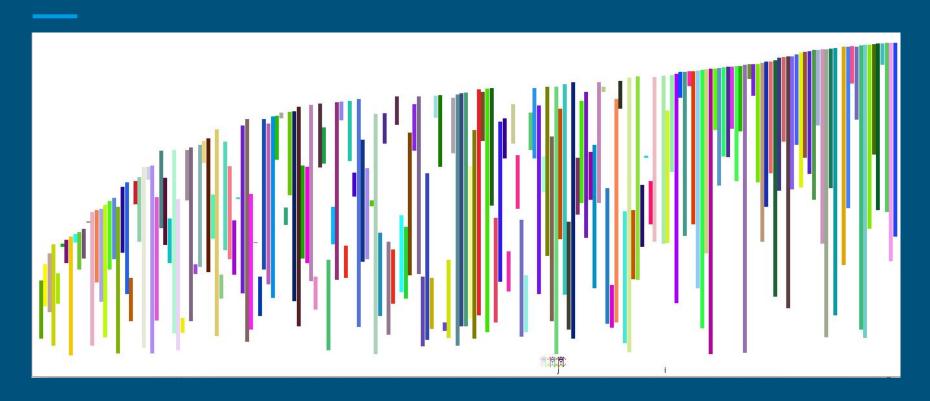
Bubble Sort - Overview

The Bubble sort works by comparing each bar with its neighbor and swapping them if they are out of order. One left-right pass of N cycles of this guarantees that the largest bar is at the far right. This bar is then considered to be in its final location and that position is no longer considered for the next pass.

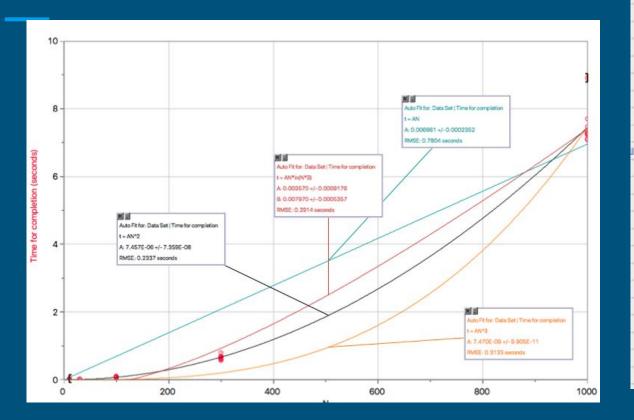
This "bubble" of finalized bars grows from right to left as the program progresses, until the entire list is in the bubble.

Some versions of the bubble sort go the opposite direction, growing the bubble on the left (smaller) size.

Bubble Sort - in progress



Bubble Sort - Analysis



	Data Set			Dat	a Set
	N	t		N	t
_		(seconds)			(seconds)
1	10	0.041	26	100	0.074
2	10	0.008	27	100	0.073
3	10	0.005	28	100	0.066
4	10	0.001	29	100	0.072
5	10	0.009	30	100	0.088
6	10	0.001	31	300	0.623
7	10	0.007	32	300	0.805
8	10	0.001	33	300	0.599
9	10	0.001	34	300	0.698
10	10	0.002	35	300	0.668
11	30	0.014	36	300	0.686
12	30	0.013	37	300	0.643
13	30	0.015	38	300	0.677
14	30	0.022	39	300	0.566
15	30	0.012	40	300	0.62
16	30	0.013	41	1000	8.911
17	30	0.009	42	1000	7.29
18	30	0.016	43	1000	7.269
19	30	0.017	44	1000	7.222
20	30	0.011	45	1000	7.256
21	100	0.104	46	1000	7.116
22	100	0.075	47	1000	7.708
23	100	0.067	48	1000	7.463
24	100	0.078	49	1000	7.261
25	100	0.067	50	1000	7.083

Bubble Sort - conclusions

- Bubble sort is $O(N^2)$ that had the best fit, and it matches theory, in that it functions like a set of nested loops over N and (on average) N/2.
- The strength of the Bubble Sort is that it is very easy to write.
- The weakness of the Bubble Sort is that it is slow and inefficient. Pieces get compared and moved many, many times before being "locked" in their final location.
- In the worst case, (an initially anti-ordered list), every bar will be moved (N-1) times before finding its final position.

Selection Sort - Overview

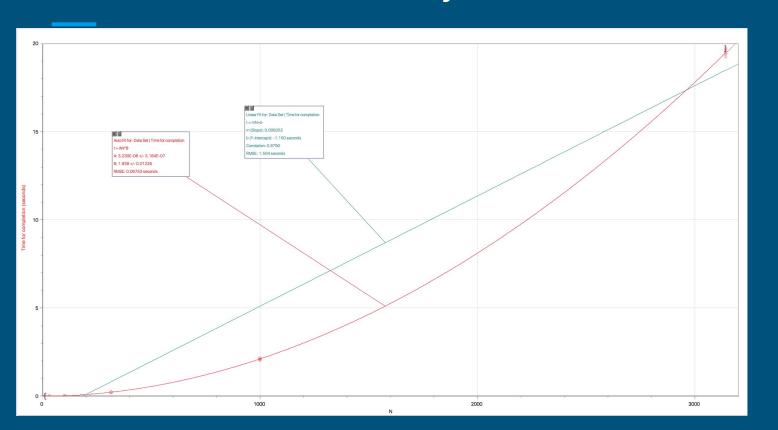
The Selection sort works by comparing the first value of a list to all others, until it finds one that is smaller. If it finds one that is smaller, it becomes the new value that is compared to the others. At the end, the smallest value takes the selected space and the program goes again for the next space. This value is considered to be in its final position and that position is no longer considered for the next passes.

This effectively acts as selecting a place and looking for a value that fits.

Selection Sort - in progress



Selection Sort - Analysis



	Data Set		
	N	N t	
		(seconds)	
_1	10	0	
2	10	0.003	
3	10	0	
4	10	0.001	
5	10	0.001	
6	31	0.005	
_ 7	31	0.001	
8	31	0.007	
9	31	0.002	
10	31	0.002	
_11	100	0.027	
12	100	0.031	
13	100	0.025	
14	100	0.026	
15	100	0.027	
16	314	0.223	
17	314	0.222	
18	314	0.215	
19	314	0.221	
20	314	0.229	
21	1000	2.143	
22	1000	2.156	
23	1000	2.14	
24	1000	2.058	
25	1000	2.084	
26	3142	19.71	
27	3142	19.562	
28	3142	19.246	
29	3142	19.421	
30	3142	19.43	

Selection Sort - conclusions

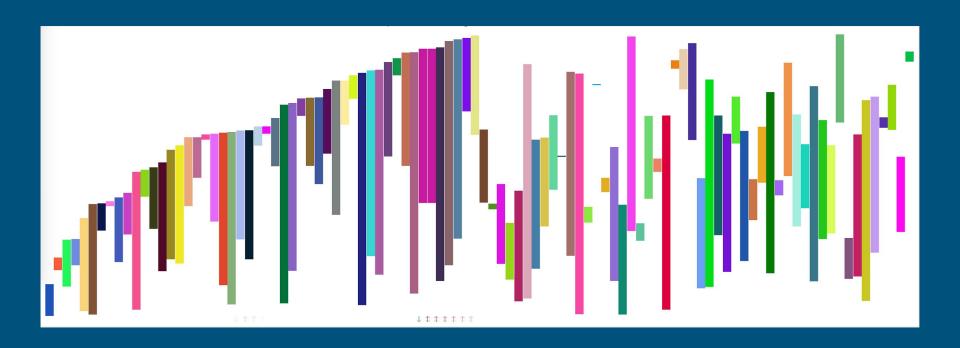
- Selection sort is O(N²) that had the best fit, and matches theory, in that it functions like a set of nested loops over N
- The strength of Selection Sort is that it is easy to write, but faster than something like bubble sort
- The weakness of Selection Sort is that it is slow.
- There isn't a worst case or best case for this sort, as it will check every bar
 in the list that hasn't been sorted by the algorithm, even if it is already in it's
 final position before the sorting begins.

Insertion Sort - Overview

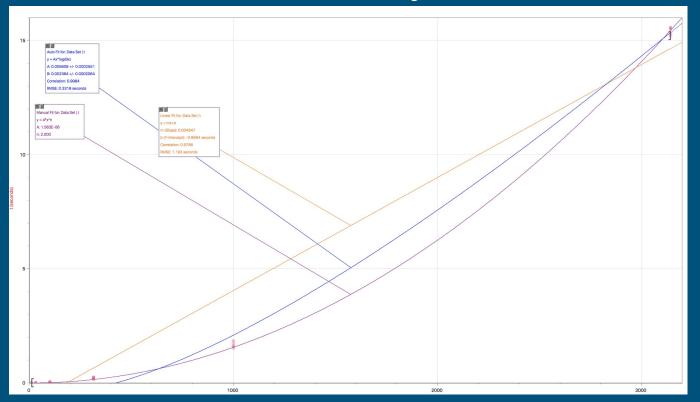
The insertion sort works by splitting the list into two sublists, with one being sorted and the other not. The program compares the value being sorted with the highest value, and keeps shifting lower until it finds a value lower than the value being sorted. This way, it knows everything to the left is lower, and to the right is higher.

Is effectively acts as selecting a value and looking for a place that fits. This is the way humans sort.

Insertion Sort - in progress



Insertion Sort - Analysis



	Data Set		
	N	N t	
		(seconds)	
1	10	0.008	
2	10	0.003	
3	10	0.001	
4	10	0.001	
5	10	0.003	
6	31	0.021	
7	31	0.003	
8	31	0.003	
9	31	0.006	
10	31	0.001	
11	100	0.072	
12	100	0.041	
13	100	0.021	
14	100	0.023	
15	100	0.02	
16	314	0.253	
17	314	0.219	
18	314	0.235	
19	314	0.159	
20	314	0.202	
21	1000	1.634	
22	1000	1.529	
23	1000	1.842	
24	1000	1.747	
25	1000	1.598	
26	3142	15.205	
27	3142	15.503	
28	3142	15.571	
29	3142	15.535	
30	3142	15.299	

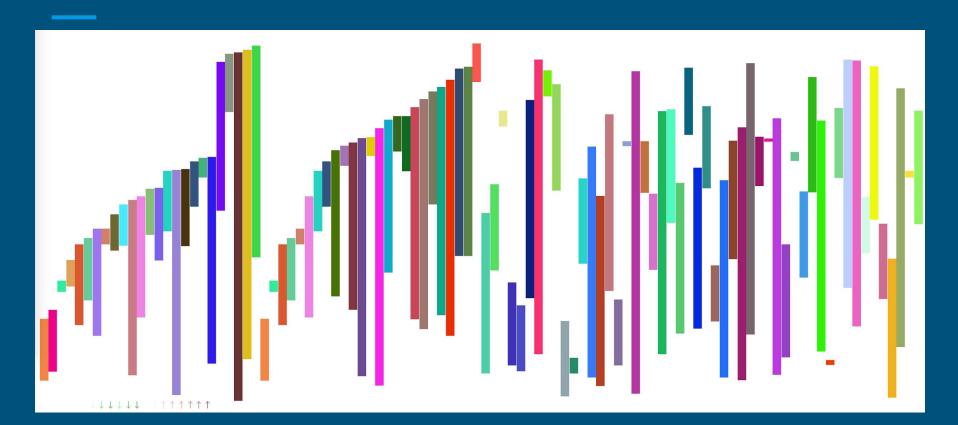
Insertion Sort - conclusions

- Insertion sort is $O(N^2)$ that had the best fit, and matches the theory in that it functions like a set of nested loops over N
- The strength of Insertion Sort is that it is easy to write, but faster than something like bubble sort or selection sort.
- The weakness of Insertion Sort is that it's slow
- The worst case for this sort is O(N²) but the best is O(N). This is because it
 only takes one step to realise if an item is already sorted relative to its
 neighbors. This means it will only take one pass to check a sorted list.

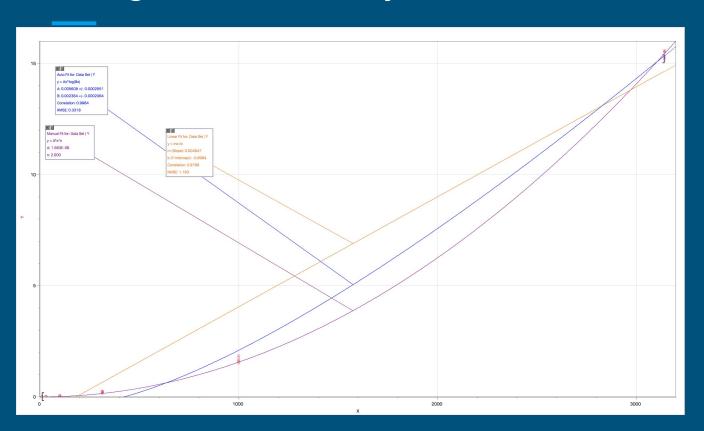
Merge Sort - Overview

The Merge Sort works by recursively breaking down a list at its center point until it has lists of one. Once this is done, it builds back up by comparing the first value of the smaller sorted lists into one bigger list over and over until it comes back to the top.

Merge Sort - in progress



Merge Sort - Analysis



N t (seconds) 1 10 0.043 2 10 0.002 3 10 0.002 3 10 0.001 5 10 0.001 6 30 0.002 7 30 0.002 8 30 0.002 10 30 0.015 11 100 0.018 12 100 0.008 13 100 0.016 14 100 0.009 15 100 0.007 16 314 0.065 17 314 0.078 18 314 0.076 19 314 0.078 19 314 0.073 21 1000 0.207 22 1000 0.27 23 1000 0.315 26 3142 0.574 27 3142 0.549 28 3142 0.593 29 3142 0.593 29 3142 0.593 29 3142 0.593 29 3142 0.593 31 10000 1.81 31 10000 1.868 32 10000 1.81 33 10000 1.924 34 10000 2.036 35 10000 1.914 34 10000 2.036 36 31415 6.75 37 31415 6.477 39 31415 6.477 39 31415 6.477 39 31415 6.437 41 100000 27.453 42 100000 27.453 44 100000 27.453 44 100000 27.453 44 100000 27.453 44 100000 27.453 44 100000 27.453 44 100000 27.453 44 100000 27.453 44 100000 27.453 45 100000 28.868 44 100000 27.036 45 100000 27.036		Date	
1 10 0.043 2 10 0.002 3 10 0.031 4 10 0.001 5 10 0.001 5 10 0.001 6 30 0.002 7 30 0.002 8 30 0.001 9 30 0.002 10 30 0.013 11 100 0.018 12 100 0.008 13 100 0.016 14 100 0.009 15 100 0.007 16 314 0.065 17 314 0.065 17 314 0.078 18 314 0.078 19 314 0.078 20 314 0.073 21 1000 0.207 22 1000 0.27 23 1000 0.311 24 1000 0.315 25 1000 0.153 26 3142 0.57 27 3142 0.549 28 3142 0.57 27 3142 0.549 28 3142 0.57 37 31415 0.57 37 31415 6.554 40 31415 6.437 41 10000 2.036 36 31415 6.477 39 31415 6.437 41 10000 27.036		N	t
2 10 0.002 3 10 0.0031 4 10 0.001 5 10 0.001 6 30 0.002 7 30 0.002 8 30 0.001 9 30 0.002 10 30 0.015 11 1 00 0.008 13 100 0.016 14 100 0.009 15 100 0.007 16 314 0.065 17 314 0.078 18 314 0.078 19 314 0.054 20 314 0.073 21 1000 0.207 22 1000 0.315 26 3142 0.593 26 3142 0.593 27 3142 0.593 28 3142 0.47 30 3142 0.593 29 3142 0.47 30 3142 0.593 29 3142 0.47 30 3142 0.593 29 3142 0.47 30 3142 0.593 31 10000 1.868 32 10000 1.81 33 10000 1.924 34 10000 2.036 35 10000 1.924 34 10000 2.036 36 31415 6.75 37 31415 6.437 41 100000 27.036			(seconds)
3 10 0.031 4 10 0.001 5 10 0.001 6 30 0.002 7 30 0.002 8 30 0.002 8 30 0.002 10 30 0.015 11 100 0.018 12 100 0.008 13 100 0.016 14 100 0.009 15 100 0.007 16 314 0.065 17 314 0.078 18 314 0.078 19 314 0.073 21 1000 0.207 22 1000 0.27 23 1000 0.311 24 1000 0.315 25 1000 0.313 24 1000 0.315 25 1000 0.313 24 1000 0.315 25 1000 0.153 26 3142 0.579 27 3142 0.593 29 3142 0.477 30 3142 0.593 29 3142 0.477 30 3142 0.549 28 3142 0.573 31 10000 1.888 32 10000 1.81 33 10000 1.924 34 10000 2.036 36 31415 6.75 37 31415 7.001 38 31415 6.554 40 31415 6.554 40 31415 6.554 41 100000 27.453 42 100000 28.868 44 100000 28.868 44 100000 28.868	1	10	0.043
4 10 0.001 5 10 0.001 6 30 0.002 7 30 0.002 8 30 0.002 10 30 0.015 11 100 0.018 12 100 0.008 13 100 0.016 14 100 0.009 15 100 0.007 16 314 0.065 17 314 0.078 18 314 0.078 19 314 0.054 20 314 0.054 20 314 0.073 21 1000 0.207 22 1000 0.27 23 1000 0.315 26 3142 0.57 27 3142 0.549 28 3142 0.57 27 3142 0.549 28 3142 0.57 27 3142 0.549 28 3142 0.57 27 3142 0.549 28 3142 0.57 31 1000 0.153 31 10000 1.868 32 10000 1.81 33 10000 1.924 34 10000 2.013 35 10000 1.81 33 10000 1.924 34 10000 2.013 35 10000 1.824 34 10000 2.013 35 10000 1.846 37 31415 6.75 37 31415 7.001 38 31415 6.554 40 31415 6.554 40 31415 6.554 40 31415 6.554 41 100000 27.453 42 100000 2.6631 43 100000 2.8688 44 100000 2.8688	2	10	0.002
5 10 0.001 6 30 0.002 7 30 0.002 8 30 0.002 8 30 0.001 9 30 0.002 10 30 0.015 11 100 0.018 12 100 0.008 13 100 0.016 14 100 0.009 15 100 0.007 16 314 0.065 17 314 0.078 18 314 0.078 19 314 0.073 21 1000 0.207 22 1000 0.27 23 1000 0.311 24 1000 0.315 25 1000 0.315 26 3142 0.57 27 3142 0.549 28 3142 0.57 27 3142 0.549 28 3142 0.57 27 3142 0.549 28 3142 0.57 30 3142 0.549 28 3142 0.57 31 10000 1.868 31 10000 1.81 33 10000 1.924 34 10000 2.036 36 31415 6.75 37 31415 6.477 39 31415 6.477 39 31415 6.437 41 100000 27.036 42 100000 28.868 44 100000 2.86681 44 100000 2.86681 44 100000 2.86681 44 100000 2.86681 44 100000 2.86681	3	10	0.031
6 30 0.002 7 30 0.002 8 30 0.001 9 30 0.002 10 30 0.002 11 10 0.018 12 100 0.018 13 100 0.016 14 100 0.009 15 100 0.007 16 314 0.065 17 314 0.076 19 314 0.076 19 314 0.076 20 314 0.073 21 1000 0.207 22 1000 0.27 23 1000 0.311 24 1000 0.315 25 1000 0.315 26 3142 0.57 27 3142 0.549 28 3142 0.57 27 3142 0.549 28 3142 0.57 31 1000 1.868 31 10000 1.818 33 10000 1.924 34 10000 2.036 36 31415 6.75 37 31415 6.477 39 31415 6.437 41 100000 27.453 42 100000 28.868 44 100000 2.86681 44 100000 2.86681 44 100000 2.86681 44 100000 2.86681 44 100000 2.8688	4	10	0.001
7 30 0.002 8 30 0.001 9 30 0.002 10 30 0.015 11 100 0.018 12 100 0.008 13 100 0.016 14 100 0.009 15 100 0.007 16 314 0.065 17 314 0.078 18 314 0.076 19 314 0.054 20 314 0.073 21 1000 0.27 22 1000 0.27 23 1000 0.315 26 3142 0.57 27 3142 0.593 26 3142 0.593 27 3142 0.593 28 3142 0.593 29 3142 0.47 30 3142 0.594 31 10000 1.81 33 10000 1.824 31 10000 1.83 33 10000 1.924 34 10000 2.036 36 31415 6.75 37 31415 6.437 41 100000 27.453 42 100000 28.868 44 100000 28.868 44 100000 28.868	5	10	0.001
8 30 0.001 9 30 0.002 10 30 0.015 11 100 0.018 12 100 0.008 13 100 0.016 14 100 0.009 15 100 0.007 16 314 0.054 17 314 0.073 18 314 0.074 19 314 0.073 21 1000 0.207 22 1000 0.27 23 1000 0.311 24 1000 0.315 25 1000 0.153 26 3142 0.574 28 3142 0.554 29 3142 0.549 29 3142 0.549 30 3142 0.554 31 10000 1.868 32 10000 1.868 32 100	6	30	0.002
9 30 0.002 10 30 0.015 11 100 0.018 12 100 0.008 13 100 0.016 14 100 0.009 15 100 0.007 16 314 0.065 17 314 0.078 18 314 0.078 19 314 0.054 20 314 0.073 21 1000 0.207 22 1000 0.27 23 1000 0.311 24 1000 0.315 25 1000 0.315 26 3142 0.57 27 3142 0.549 28 3142 0.593 29 3142 0.47 30 3142 0.549 28 3142 0.57 27 3142 0.549 38 3141 0.054 31 10000 1.868 32 10000 1.81 33 10000 1.924 34 10000 2.013 35 10000 1.924 34 10000 2.013 35 10000 3.315 36 31415 6.75 37 31415 7.001 38 31415 6.477 39 31415 6.554 40 31415 6.554 41 100000 27.453 42 100000 28.868 44 100000 28.868 44 100000 27.036	7	30	0.002
10 30 0.015 11 100 0.018 12 100 0.008 13 100 0.016 14 100 0.009 15 100 0.007 16 314 0.065 17 314 0.078 18 314 0.076 19 314 0.076 20 314 0.073 21 1000 0.207 22 1000 0.27 23 1000 0.311 24 1000 0.315 25 1000 0.315 26 3142 0.57 27 3142 0.549 28 3142 0.57 27 3142 0.549 28 3142 0.57 27 3142 0.549 28 3142 0.57 30 3142 0.549 31 10000 1.868 32 10000 1.81 33 10000 1.924 34 10000 2.036 36 31415 6.75 37 31415 6.75 37 31415 6.437 41 100000 27.453 42 100000 28.868 44 100000 28.868 44 100000 2.6631	8	30	0.001
11 100 0.018 12 100 0.008 13 100 0.016 14 100 0.009 15 100 0.007 16 314 0.065 17 314 0.076 19 314 0.076 20 314 0.073 21 1000 0.207 22 1000 0.27 23 1000 0.315 25 1000 0.315 26 3142 0.579 28 3142 0.579 28 3142 0.579 28 3142 0.579 31 1000 0.315 30 3142 0.549 31 10000 1.868 32 10000 1.818 33 10000 1.924 34 10000 2.036 36 31415 6.75 37 31415 6.477 39 31415 6.437 41 100000 27.453 42 100000 28.868 44 100000 27.453 42 100000 28.868 44 100000 27.036	9	30	0.002
12 100 0.008 13 100 0.016 14 100 0.009 15 100 0.007 16 314 0.065 17 314 0.078 18 314 0.078 19 314 0.074 20 314 0.073 21 1000 0.207 22 1000 0.27 23 1000 0.315 26 3142 0.57 27 3142 0.549 28 3142 0.57 27 3142 0.549 28 3142 0.57 31 1000 1.868 31 10000 1.81 33 10000 1.924 34 10000 2.036 36 31415 6.75 37 31415 6.437 41 100000 27.453 42 100000 28.868 44 100000 28.868 44 100000 27.036	10	30	0.015
13 100 0.016 14 100 0.009 15 100 0.007 16 314 0.065 17 314 0.078 18 314 0.078 19 314 0.054 20 314 0.073 21 1000 0.207 22 1000 0.315 25 1000 0.315 26 3142 0.57 27 3142 0.593 29 3142 0.47 30 3142 0.593 29 3142 0.47 30 3142 0.593 31 10000 1.868 32 10000 1.81 33 10000 1.924 34 10000 2.036 36 31415 6.75 37 31415 6.477 39 31415 6.437 41 100000 27.453 42 100000 28.868 44 100000 28.868	11	100	0.018
14 100 0.009 15 100 0.007 16 314 0.056 17 314 0.078 18 314 0.054 19 314 0.054 20 314 0.073 21 1000 0.207 22 1000 0.27 23 1000 0.315 26 3142 0.57 27 3142 0.593 29 3142 0.47 30 3142 0.554 31 10000 1.868 32 10000 1.868 32 10000 1.868 32 10000 1.924 34 10000 2.036 36 31415 6.75 37 31415 7.001 38 31415 6.477 39 31415 6.437 41 100000 27.453 42 <td>12</td> <td>100</td> <td>0.008</td>	12	100	0.008
15 100 0.007 16 314 0.065 17 314 0.078 18 314 0.078 19 314 0.054 20 314 0.054 20 314 0.073 21 1000 0.207 22 1000 0.27 23 1000 0.311 24 1000 0.315 25 1000 0.153 26 3142 0.57 27 3142 0.549 28 3142 0.593 29 3142 0.47 30 3142 0.554 31 10000 1.868 32 10000 1.81 33 10000 1.924 34 10000 2.013 35 10000 1.924 34 10000 2.036 36 31415 6.75 37 31415 7.001 38 31415 6.437 39 31415 6.554 40 31415 6.554 40 31415 6.554 41 100000 27.453 42 100000 28.868 44 100000 28.868	13	100	0.016
16 314 0.065 17 314 0.078 18 314 0.076 19 314 0.056 20 314 0.057 21 1000 0.207 22 1000 0.27 23 1000 0.315 25 1000 0.153 26 3142 0.57 27 3142 0.549 28 3142 0.57 27 3142 0.549 28 3142 0.47 30 3142 0.549 31 10000 1.868 32 1000 1.81 33 1000 1.924 34 10000 2.036 36 31415 6.75 37 31415 6.477 39 31415 6.437 41 100000 27.453 42 100000 28.868 44 100000 2.8686 44 100000 2.8686	14	100	0.009
17 314 0.078 18 314 0.076 19 314 0.076 20 314 0.073 21 1000 0.207 22 1000 0.27 23 1000 0.315 25 1000 0.153 26 3142 0.593 28 3142 0.593 29 3142 0.47 30 3142 0.593 31 10000 1.868 32 10000 1.81 33 10000 1.924 34 10000 2.036 36 31415 6.75 37 31415 6.477 39 31415 6.437 41 100000 27.453 42 100000 28.868 44 100000 28.868	15	100	0.007
18 314 0.076 19 314 0.054 20 314 0.073 21 1000 0.207 22 1000 0.27 23 1000 0.311 24 1000 0.315 25 1000 0.153 26 3142 0.57 27 3142 0.593 29 3142 0.47 30 3142 0.593 29 3142 0.47 30 1.868 32 10000 1.81 33 10000 1.924 34 10000 2.036 36 31415 6.75 37 31415 6.477 39 31415 6.477 39 31415 6.437 41 100000 27.453 42 100000 28.868 44 100000 28.868 44 100000 27.036	16	314	0.065
19 314 0.054 20 314 0.073 21 1000 0.207 22 1000 0.27 23 1000 0.311 24 1000 0.315 25 1000 0.153 26 3142 0.57 27 3142 0.549 28 3142 0.593 3142 0.477 30 3142 0.477 30 1342 0.454 31 10000 1.868 32 10000 1.81 33 10000 1.924 34 10000 2.013 35 10000 2.036 36 31415 6.75 37 31415 7.001 38 31415 6.477 39 31415 6.437 41 100000 27.453 42 100000 28.868 44 100000 28.868 44 100000 27.036	17	314	0.078
20 314 0.073 21 1000 0.207 22 1000 0.27 23 1000 0.311 24 1000 0.315 25 1000 0.153 26 3142 0.57 27 3142 0.549 28 3142 0.593 29 3142 0.47 30 3142 0.554 31 10000 1.868 32 10000 1.81 33 10000 1.924 34 10000 2.013 35 10000 2.036 36 31415 6.75 37 31415 7.001 38 31415 6.437 41 100000 27.453 42 100000 28.868 44 100000 2.6631 43 100000 28.868 44 100000 27.036	18	314	0.076
21 1000 0.207 22 1000 0.27 23 1000 0.311 24 1000 0.315 25 1000 0.53 26 3142 0.57 27 3142 0.549 28 3142 0.47 30 3142 0.53 31 10000 1.868 32 10000 1.81 33 10000 2.036 36 31415 6.75 37 31415 6.437 41 100000 27.453 42 100000 28.868 44 100000 2.6631 43 100000 28.8688 44 100000 2.7.366	19	314	0.054
22 1000 0.27 23 1000 0.311 24 1000 0.315 25 1000 0.153 26 3142 0.57 27 3142 0.549 28 3142 0.47 30 3142 0.549 31 10000 1.868 32 10000 1.81 33 10000 2.036 36 31415 6.75 37 31415 6.477 39 31415 6.437 41 100000 27.453 42 100000 28.868 44 100000 28.868	20	314	0.073
23 1000 0.311 24 1000 0.315 25 1000 0.153 26 3142 0.57 27 3142 0.593 29 3142 0.47 30 3142 0.554 31 10000 1.868 32 10000 1.81 33 10000 2.036 36 31415 6.75 37 31415 6.477 39 31415 6.437 41 10000 27.453 42 100000 28.868 44 100000 27.036	21	1000	0.207
24 1000 0.315 25 1000 0.153 26 3142 0.57 27 3142 0.549 28 3142 0.593 29 3142 0.47 30 3142 0.554 31 10000 1.868 32 10000 1.81 33 10000 1.924 34 10000 2.013 35 10000 2.036 36 31415 6.75 37 31415 7.001 38 31415 6.477 39 31415 6.437 41 100000 27.453 42 100000 27.453 42 100000 28.868 44 100000 27.036	22	1000	0.27
25 1000 0.153 26 3142 0.57 27 3142 0.549 28 3142 0.593 29 3142 0.477 30 3142 0.554 31 10000 1.868 32 10000 1.81 33 10000 1.924 34 10000 2.013 35 10000 2.036 36 31415 6.75 37 31415 7.001 38 31415 6.477 39 31415 6.437 41 100000 27.453 42 100000 28.868 44 100000 27.036	23	1000	0.311
26 3142 0.57 27 3142 0.549 28 3142 0.593 29 3142 0.47 30 3142 0.554 31 10000 1.868 32 10000 1.81 33 10000 2.013 35 10000 2.036 36 31415 6.75 37 31415 7.001 38 31415 6.437 41 100000 27.453 42 100000 28.868 44 100000 28.868	24	1000	0.315
27 3142 0.549 28 3142 0.593 29 3142 0.47 30 3142 0.553 31 10000 1.868 32 10000 1.81 33 10000 2.013 35 10000 2.036 36 31415 6.75 37 31415 6.477 39 31415 6.437 41 100000 27.453 42 100000 28.868 44 100000 28.868	25	1000	0.153
28 3142 0.593 29 3142 0.47 30 3142 0.554 31 10000 1.868 32 10000 1.91 33 10000 2.036 36 31415 6.75 37 31415 6.477 39 31415 6.437 41 100000 27.453 42 100000 28.868 44 100000 27.036	26	3142	0.57
29 3142 0.47 30 3142 0.554 31 10000 1.868 32 10000 1.91 33 10000 2.036 36 31415 6.75 37 31415 7.001 38 31415 6.477 39 31415 6.554 40 31415 6.437 41 100000 27.453 42 100000 28.868 44 100000 27.036	27	3142	
30 3142 0,554 31 10000 1,868 32 10000 1,81 33 10000 1,924 34 10000 2,0136 36 31415 6,75 37 31415 7,001 38 31415 6,477 39 31415 6,437 41 100000 27,453 42 100000 27,453 44 100000 28,868 44 100000 27,036	28	3142	0.593
31 10000 1.868 32 10000 1.81 33 10000 1.924 34 10000 2.013 35 10000 2.036 36 31415 6.75 37 31415 7.001 38 31415 6.477 39 31415 6.554 40 31415 6.437 41 100000 27.453 42 100000 28.868 44 100000 27.036	29	3142	0.47
31 10000 1.868 32 10000 1.81 33 10000 1.924 34 10000 2.013 35 10000 2.036 36 31415 6.75 37 31415 7.001 38 31415 6.477 39 31415 6.554 40 31415 6.437 41 100000 27.453 42 100000 28.868 44 100000 27.036	30	3142	
32 10000 1.81 33 10000 1.924 34 10000 2.013 35 10000 2.036 36 31415 6.75 37 31415 7.001 38 31415 6.477 39 31415 6.437 41 100000 27.453 42 100000 28.868 44 100000 27.036	31	10000	
34 10000 2.013 35 10000 2.036 36 31415 6.75 37 31415 7.001 38 31415 6.477 39 31415 6.437 41 100000 27.453 42 100000 28.868 44 100000 27.036	32	10000	
34 10000 2.013 35 10000 2.036 36 31415 6.75 37 31415 7.001 38 31415 6.477 39 31415 6.437 41 10000 27.453 42 10000 26.631 43 10000 28.868 44 10000 27.036	33	10000	
36 31415 6.75 37 31415 7.001 38 31415 6.477 39 31415 6.554 40 31415 6.437 41 100000 27.453 42 100000 26.631 43 100000 27.036	34	10000	
36 31415 6.75 37 31415 7.001 38 31415 6.477 39 31415 6.554 40 31415 6.437 41 100000 27.453 42 100000 26.631 43 100000 28.868 44 100000 27.036	35	10000	2.036
37 31415 7.001 38 31415 6.477 39 31415 6.554 40 31415 6.437 41 100000 27.453 42 100000 26.631 43 100000 27.036			
38 31415 6.477 39 31415 6.554 40 31415 6.437 41 100000 27.453 42 100000 26.631 43 100000 28.868 44 100000 27.036	37		
39 31415 6.554 40 31415 6.437 41 100000 27.453 42 100000 26.631 43 100000 28.868 44 100000 27.036			
40 31415 6.437 41 100000 27.453 42 100000 26.631 43 100000 28.868 44 100000 27.036	39		
41 100000 27.453 42 100000 26.631 43 100000 28.868 44 100000 27.036	40		
42 100000 26.631 43 100000 28.868 44 100000 27.036			
43 100000 28.868 44 100000 27.036			
44 100000 27.036			

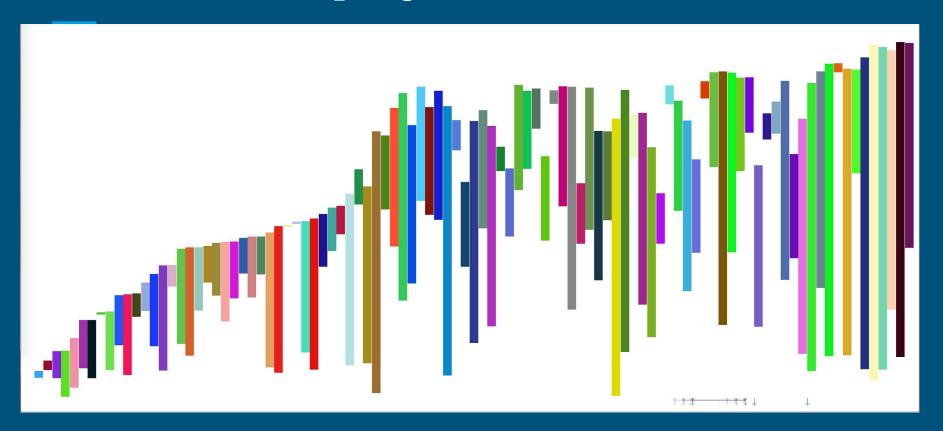
Merge Sort - conclusions

- Merge sort is O(nlogn) that had the best fit, and it matches theory.
- The strength of merge sort is its speed.
- The weakness of merge sort is that it takes a lot of memory to run, as it recurses and keep making subsets of the previous set as the function breaks down.
- There is no best or worst case for this sort.

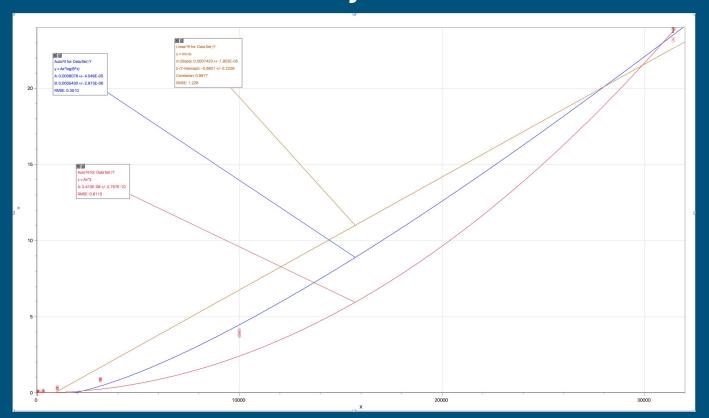
Quick Sort - Overview

The Quick sort works by taking the rightmost value, and calls it the pivot. Then, the algorithm as if there's an imaginary wall to the left of the first value. Now, by checking each value, the algorithm goes through and compares the value to the pivot. If the new value is less than the pivot, it goes to the left of the wall, switching places with the first value to the right of the wall, and by moving the wall up one place. If it's greater than the pivot, it stays in its place. At the very end, the pivot gets swapped with the value directly to the right of the wall. Now, everything to the left of the pivot is smaller than it and everything to the right of the pivot is larger than it. These two halves are then treated as new sortable lists and are then sorted the same way until the whole list is sorted.

Quick Sort - in progress



Quick Sort - Analysis



	Data Set	
	Х	Y
1	10	0.009
2	10	0.003
3	10	0.005
4	10	0.001
_ 5	10	0.005
6	30	0.005
_ 7	30	0.006
8	30	0.009
9	30	0.093
10	30	0.005
11	100	0.012
12	100	0.058
13	100	0.052
14	100	0.068
15	100	0.009
16	314	0.054
17	314	0.107
18	314	0.167
19	314	0.125
20	314	0.044
21	1000	0.405
22	1000	0.249
23	1000	0.298
24	1000	0.264
25	1000	0.264
26	3142	0.922
27	3142	0.868
28	3142	0.759
29	3142	0.898
30	3142	0.856
31	10000	4.128
32	10000	3.906
33	10000	3.804
34	10000	3.702
35	10000	4.022
36	31415	23.83
37	31415	23.908
38	31415	23.273
39	31415	23.115
40	31415	23.978

Quick Sort - conclusions

- Quick sort works most like an O(nlogn) that had the best fit. However, this can act like an $O(n^2)$ algorithm. It acts more like $O(n^2)$ the more the list is already sorted when the algorithm starts (worst case).
- The strength of Quick Sort is that it is faster than most other simple alternatives, while at the same time not taking any significant extra memory like Merge sort.
- The weakeness of quick sort is that it can potentially act as an O(n²)
 algorithm.

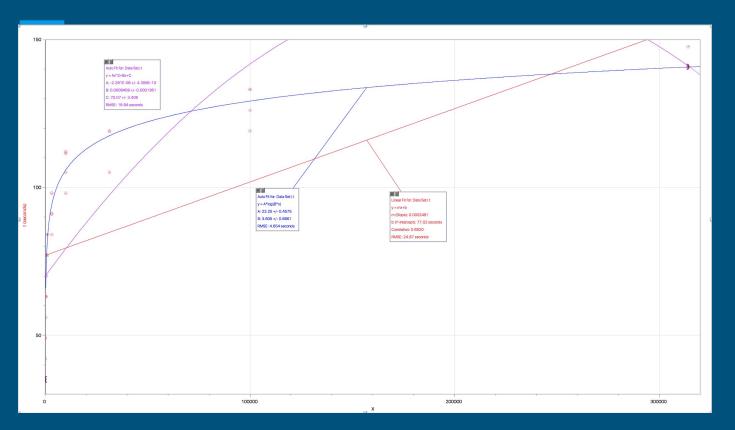
Binary Search - Overview

The binary search works by taking advantage of the nature of ordered lists. It starts in the middle, and checks if the value being searched for is less than or greater than the middle value. The algorithm then goes to the middle of the edge and middle and checks again. This happens over and over until there is only one value left. If the values match, it confirms that the value exists. If the values don't match, the value is not in the list.

Binary Search - in progress

There's not really a visual way to represent this:/

Binary Search - Analysis



	Data Set		
	X	t	
		(seconds)	
1	10	35.012	
2	10	35.004	
3	10	35.006	
4	10	35.019	
5	10	42.008	
6	30	49.012	
_ 7	30	42.009	
8	30	49.013	
9	30	49.009	
10	30	49.011	
11	100	56.011	
12	100	63.018	
_13	100	56.012	
14	100	63.015	
15	100	63.008	
16	314	70.027	
17	314	77.027	
18	314	63.016	
19	314	70.027	
20	314	77.018	
21	1000	77.016	
22	1000	84.031	
_23	1000	84.031	
24	1000	77.022	
25	1000	84.031	
26	3142	98.026	
27	3142	91.027	
28	3142	98.033	
29	3142	84.019	
30	3142	91.016	
31	10000	105.046	
32	10000	112.036	
33	10000	98.026	
34	10000	112.003	
35	10000	111.451	
36	31415	119.052	
37	31415	119.051	
38	31415	119.055	
39	31415	119.045	
40	31415	105.055	
41	100000	126.129	
42	100000	133.127	
43	100000	133.08	
44	100000	133.167	
45	100000	119.117	
46	314159	140.841	
47	314159	140.659	
48	314159	147.597	
49	314159	140.749	
50	314159	140.968	

Binary Search - conclusions

- Binary Search is O(logn) that had the best fit, and it matches theory.
- The strength of Binary search is that it's easy to code and really fast.
- There really isn't a weakness to this algorithm that I can think of.
- The O(logn) is the worst case, but the best case is potentially O(1).