Lab 9 report

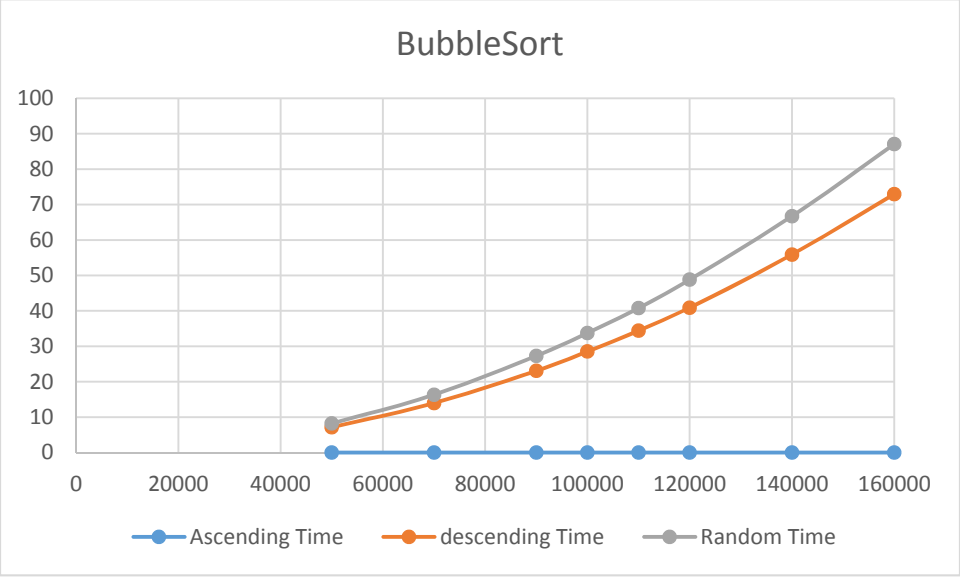
EECS 268

Instructor: John Gibbons

TA: Nimmakayala Surya

Author: Chen Long

BubbleSort

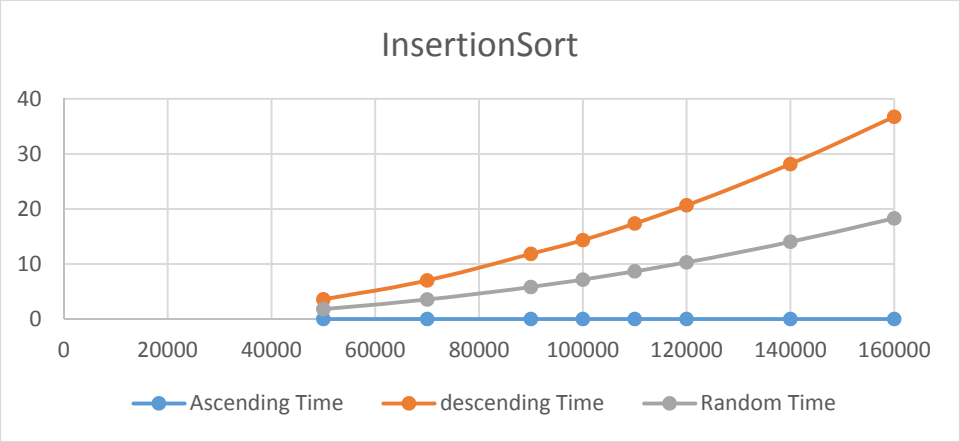


As we observed here, bubble sort is extremely good at sorting ordered array but it has the slowest processing speeding when doing the unsorted array. The reason for this is that bubble sort will go through the whole array n times, and for each time, it will go through the array n-i times in the worst case. We can also see that the trace of the graph looks like an n^2 function, which is one way of demonstrating that the big-O of bubble sort is n^2. The prediction of the time processing of 1000000 size array is 2500s for random order and 2000s for descending order.

SelectionSort

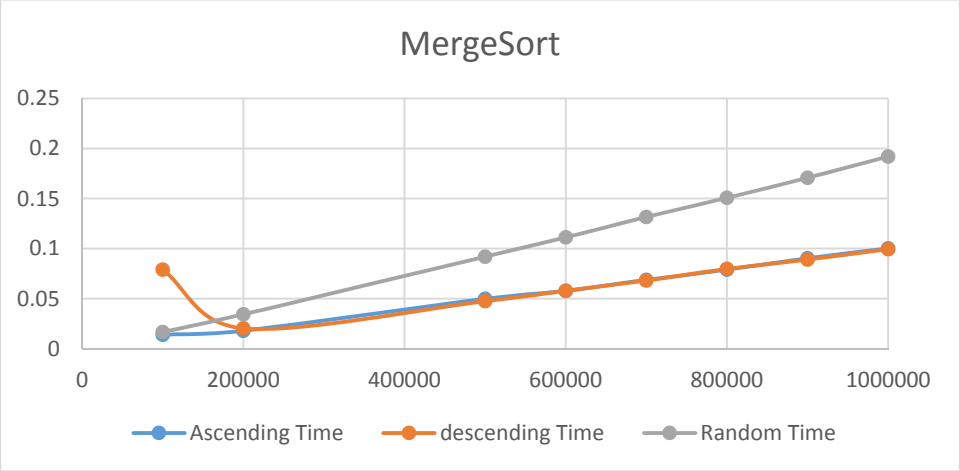
The character for selection sort is really obvious that for all the three different sorting, the time consumption is mostly the same, which proves the Big-O of selection sort of $O(n^2)$ for all cases. The reason for this is that selection sort will go over the whole array no matter what sorting order it is, therefore the processing is the same in all cases besides that sorted array doesn't need to switch the order, which is why ascending takes a little less time than other order. The prediction for 1000000 size is 950s for all cases.

InsertionSort



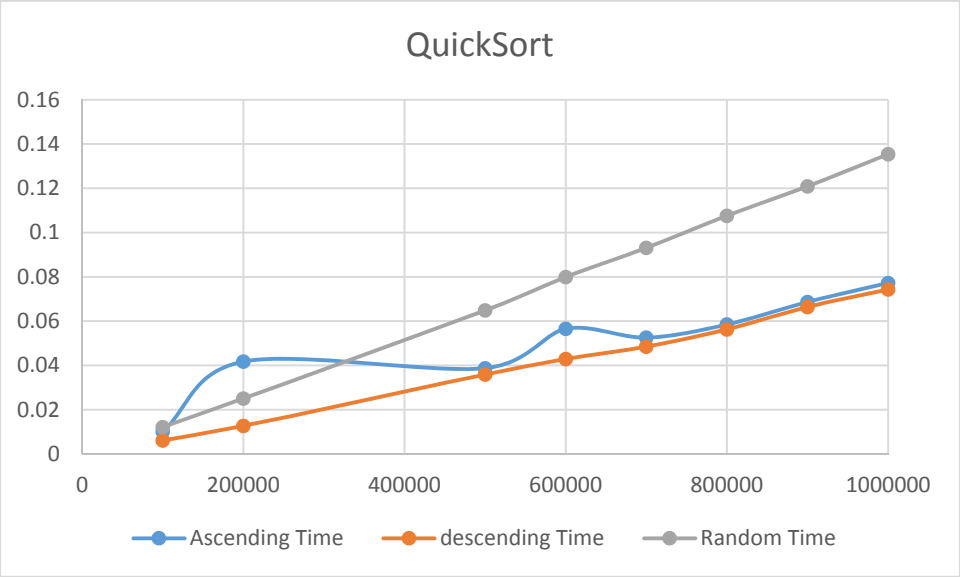
The time consume of insertion sort is a little close to bubble sort. Since for every spots the insertion sort is going through, it will compare with every spot at the left side of the array; therefore, in the worst case, it is going to compare and switch at every spot, which is going to consume a long time. The time consumption drops significantly when the order complexity drops a little bit. Even though the big-O of insertion in average case is $O(n^2)$, but it makes a lot of difference compared to the worst case. The prediction of 1000000 size is 1250s for descending, 550s for random order, and 0.005s for ascending order.

MergeSort



Merge sort consumes significant time when processing sorting compare to the previous sorting algorithm, the reason of this is that merge sort takes O(n) extra space complexity while the previous algorithm takes only O(1) extra space complexity, which means that merge sort takes significantly bigger memory than the previous three algorithm. Furthermore, we observed that the graph is like a linear line, however, the time complexity is not O(n) but O(nlog(n)), my explanation is that since it's a log function, the change of n should be way bigger than 1000000 to make the graph looks like a curve line, otherwise logn will be just like a constant which will make the graph looks like a linear function.

QuickSort



Quick sort is the fastest sorting algorithm we have observed so far, besides the sorted array. The space complexity is O(logn), which isn't too much space required. The reason why quick

sort is so fast is that there are a really big chance that the pivot value on the right side is so small to make the sorting so efficient. However, it will be really bad in a sorted array since the pivot value will always be the biggest if we choice the right side. That's why we observe a really slow time in ascending. We can also observe that the graph present a linear line, which demonstrate the time complexity is O(nlog(n)), the same thing happens to quick sort, the size of the array is too small that makes it looks like a linear function. For the ascending, since the worst case in quicksort is sorted array, which makes the time complexity to $O(n^2)$.

Raw Data

BubbleSort

Size	Ascending Time	descending Time	Random Time
50000	0.000128	7.1337	8.21215
70000	0.000565	13.9615	16.3163
90000	0.000622	23.0655	27.2964
100000	0.001122	28.5221	33.7526
110000	0.000315	34.4157	40.7654
120000	0.000361	40.8937	48.816
140000	0.000892	55.87	66.7031
160000	0.000892	72.9599	87.0639

Insertion

Size	Ascending Time	descending Time	Random Time
50000	0.000266	3.58334	1.80699
70000	0.000299	7.00662	3.54725
90000	0.001458	11.8515	5.83664
100000	0.000376	14.3417	7.16443
110000	0.000425	17.3655	8.66079
120000	0.000467	20.6628	10.2941
140000	0.000574	28.1422	14.0386
160000	0.000831	36.7567	18.287

Selection

Size	Ascending Time	descending Time	Random Time
50000	2.54671	2.71331	2.82801
70000	5.06032	5.24921	5.5443
90000	8.24548	8.70045	9.15644
100000	10.2025	10.7394	11.2832
110000	12.3337	13.0123	13.6774
120000	14.6599	15.5071	16.3153
140000	19.9604	21.0671	22.1775

160000	25.9836	27.5	28.941

Quicksort

Size	Ascending Time	descending Time	Random Time
100000	0.010137	0.00607	0.012129
200000	0.04171	0.01269	0.025043
500000	0.038732	0.035856	0.064863
600000	0.056539	0.042884	0.079927
700000	0.052549	0.048484	0.093104
800000	0.058482	0.056271	0.107604
900000	0.068633	0.066386	0.120893
1000000	0.077253	0.074298	0.135374

MergeSort

Size	Ascending Time	descending Time	Random Time
100000	0.013934	0.07896	0.016504
200000	0.017901	0.02023	0.034281
500000	0.049763	0.047472	0.091906
600000	0.057728	0.057792	0.111203
700000	0.068719	0.068129	0.131592
800000	0.07896	0.079661	0.150653
900000	0.090331	0.089008	0.170709
1000000	0.100231	0.099389	0.191976