CS 165

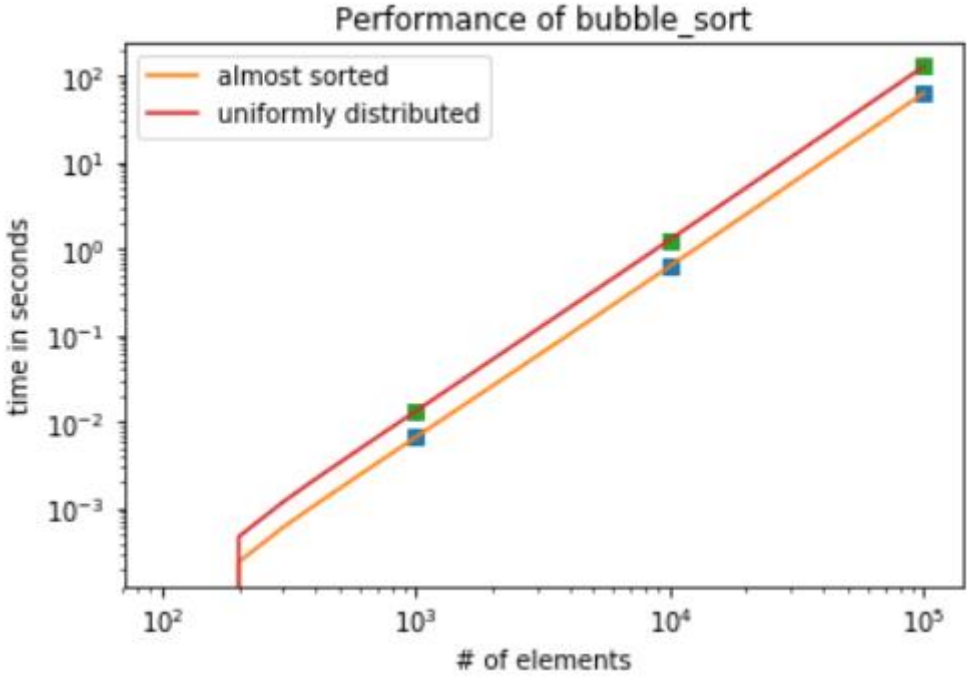
Project 1 Report

Wenzhuo Wang

14444481

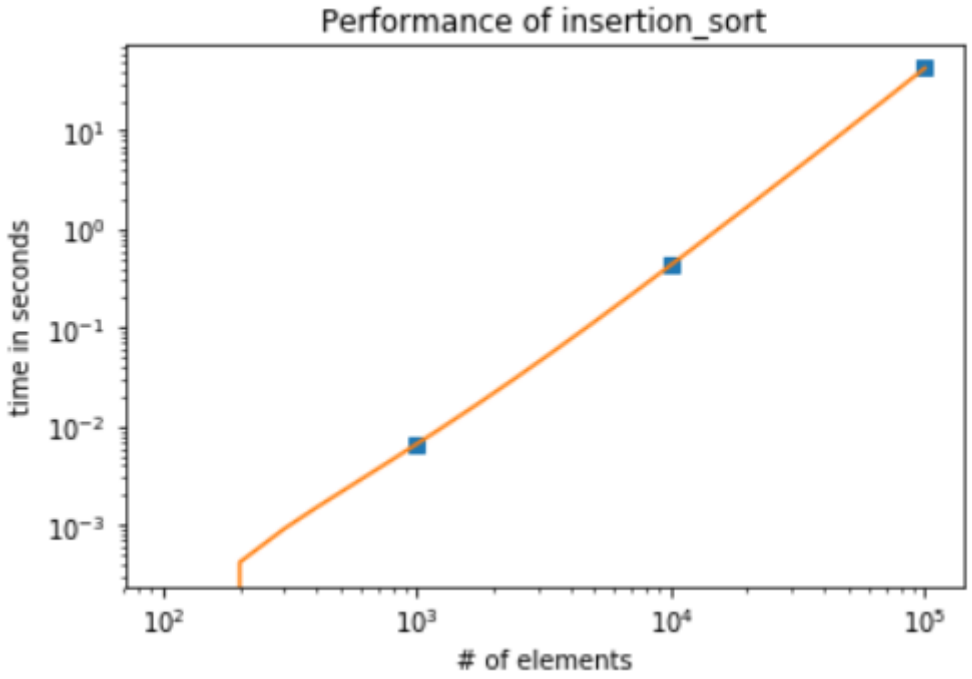
1) Bubble sort

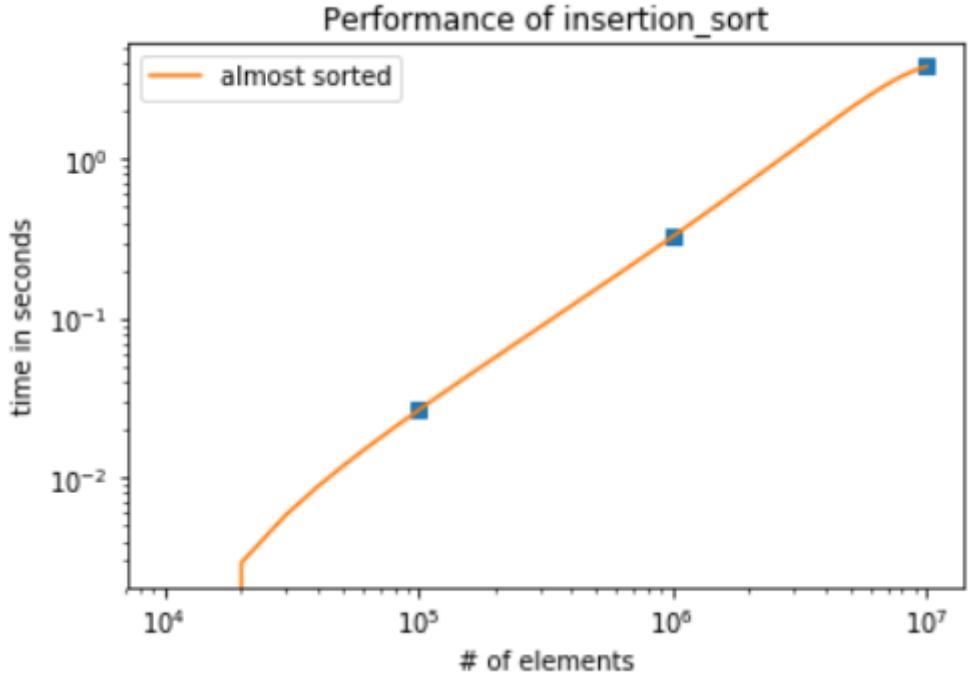
In bubble sort, starting from each element, the algorithm repeats the following steps: Compare the current element with the next one. If the next one is greater than current one, swap them. Then move to the next element. The time complexity of bubble sort is (n-1)+(n-2)+…+3+2+1=n(n-1)/2. In Big-Oh notation, the complexity of this algorithm is calculated as O(n^2). Because bubble sort will always do such n(n-1)/2 comparison consistently, both uniformly distributed permutation and almost sorted distribution require nearly equal time to finish sorting. In the following graph, we can see that when n gets enlarged by 10, the O(n^2) complexity correspondingly increases by 100 = (10^2) times, which proves the above analysis.



2) Insertion sort

In insertion sort, for each element, repeat the following steps: compare it with its previous element, if the previous one is greater than current one, swap them. The swapping does not stop until the previous one is smaller than the current one. In the worst case, each element will need to swap with all the element in front of it. It will take n(n-1)/2 operation to finish the sort, which is O(n^2) complexity. In general case, the time complexity of insertion sort is still O(n^2). However, in almost-sorted permutations, at most 4*log(n) elements are at the wrong position. Most elements will only compare with the right most element of the sorted sub-array. In the first graph of a uniform distribution case, we can see that the time increases 100 times when the number of elements increase 10 times. In the second graph of an almost-sorted case, the time and number of elements increase at the same rate, which is 10 times per increment.

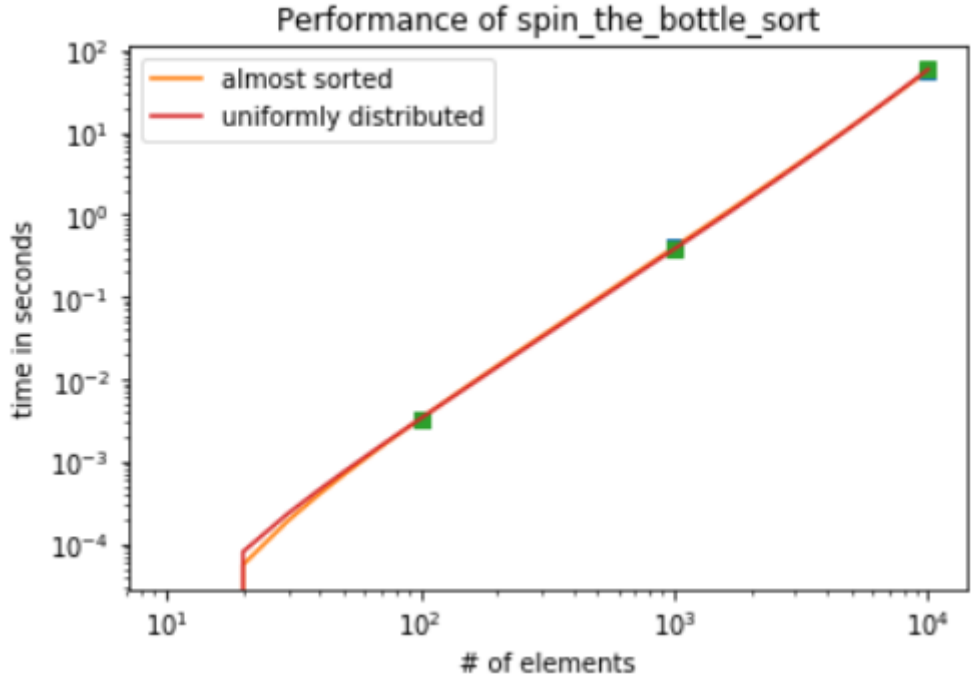




3) Spin the bottle sort

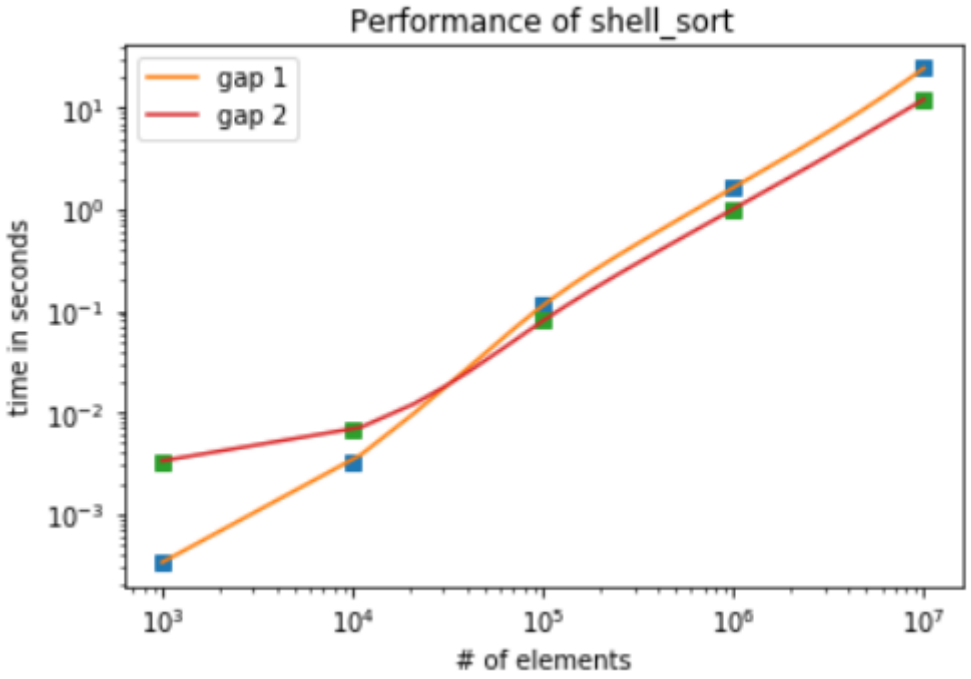
In spin the bottle sort, for each element, the algorithm will compare the element with another random element in the array. If the index of current element is smaller than that of the randomly selected element but the element value is greater than that of the random element, or the index of current element is greater than that of the randomly selected element but the element value is smaller than that of the random element, the two

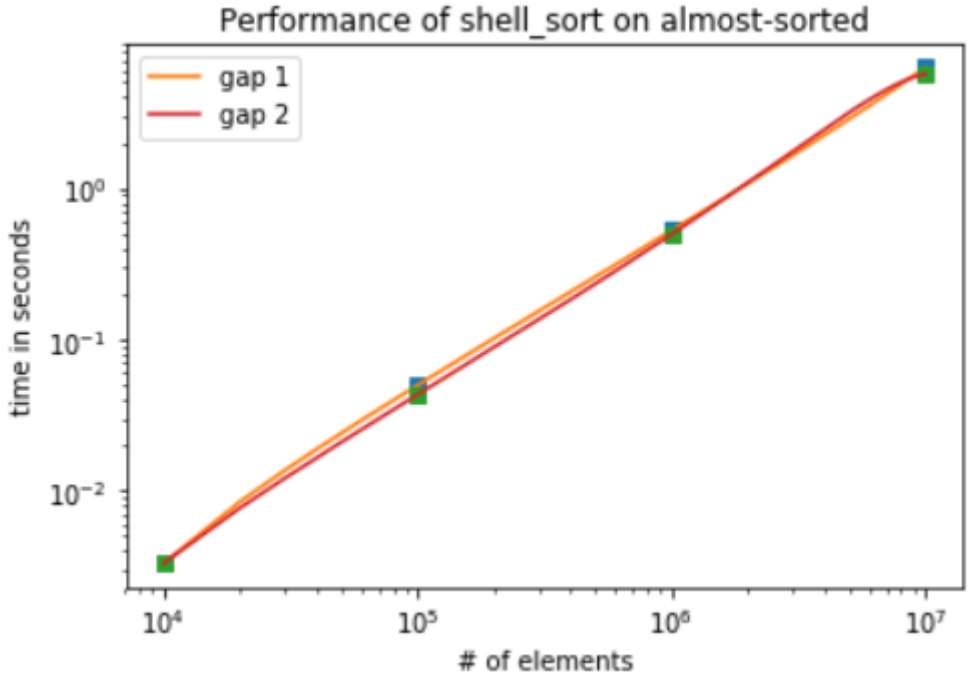
elements will be swapped. For each loop, the array will be verified on whether it is sorted. If it is sorted, the loop will terminate. The time complexity of spin the bottle sort is O(log(n)*n^2).



4) Shell sort

In shell sort, a gap set is required for index-specific sorting. The last element in the gap set should decrease to 1. The whole algorithm is similar to insertion sort. Starting from the large gap, each element will be compared with the element in front of it with the distance equal to the gap. For instance, if the gap set starts with 5, then the element at index 5 will be compared with the element at index 0. If the element at index 5 is smaller than the element at index 0, the two elements will be swapped. The element at index 6 will be compared with the element at index 1. With such gap, shell sort could create an almost-sorted array by a few steps. For both uniformly distributed permutation and almost sorted permutation, the time complexity increases by 10 times when the number of elements increases by 10 times.

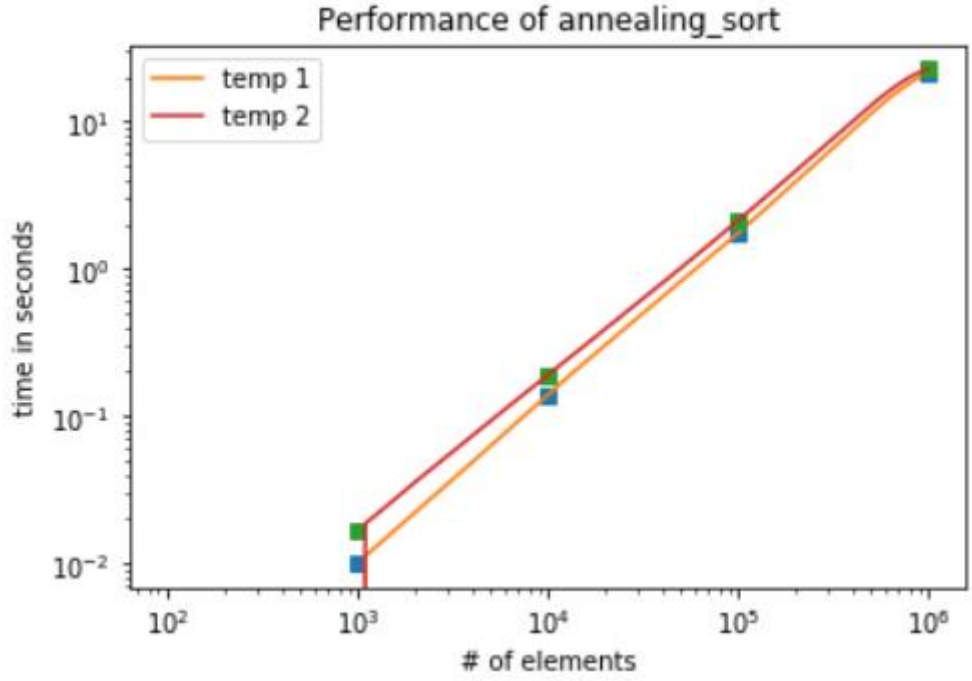


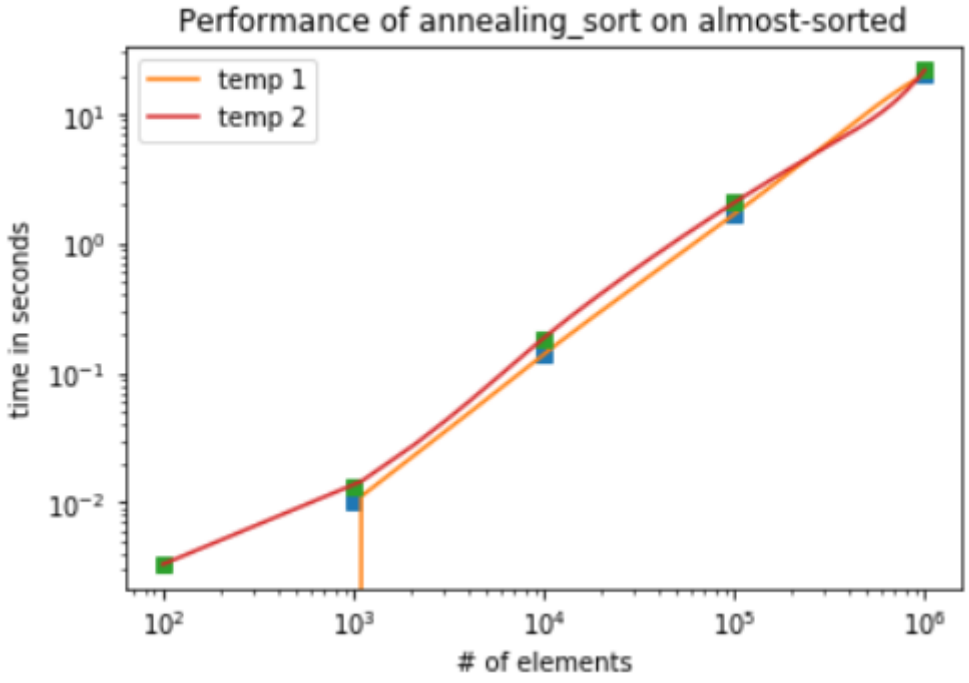


5) Annealing sort

In annealing sort, a temperature set and a repetition set are required. The elements in temperature set should keep decreasing till 0 (the last element in temperature set). The elements in repetition set should be greater than 0. Then the sort is based on each element in temperature set. For each temperature, it will have two loops. The first loop starts from the first element of the array to the second last one, and then randomly chooses an element in the range of temperature after the current element. If the chosen element is

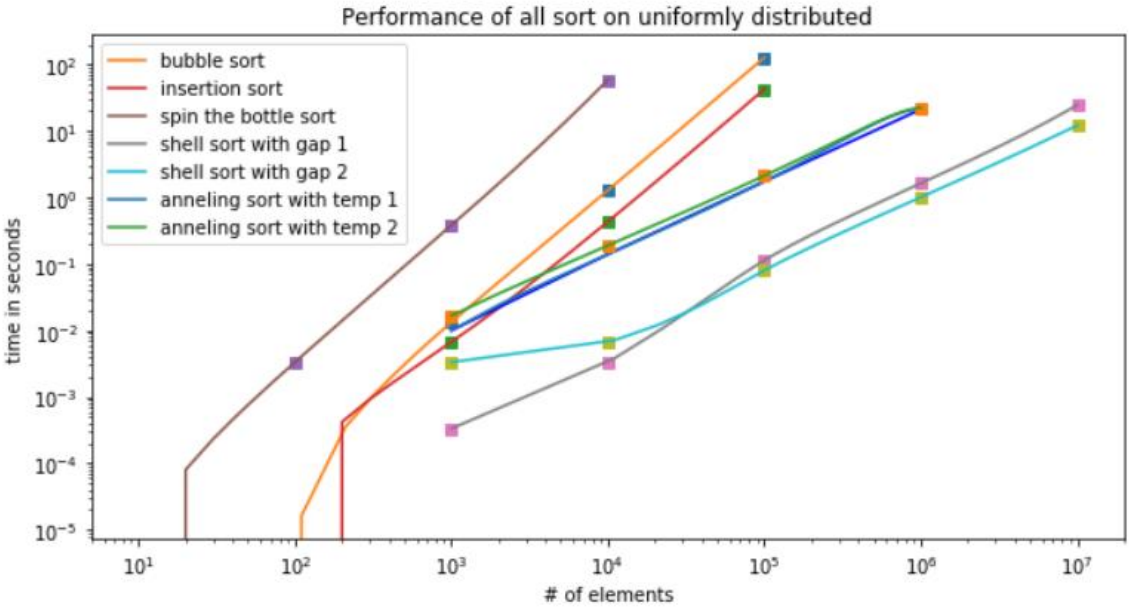
smaller than the current one, the two elements will be swapped. The second loop starts from the last element to the second one and then randomly chooses element in the range of temperature before the current one. If the chosen element is greater than the current one, the two elements will be swapped. From the graph, we can see that the two temperature sets require nearly equal time to finish the sorting. When the number of elements increases by 10 times, the time also increases by 10 times.





Graph and Conclusion

If the input is a uniformly distributed permutation, the spin the bottle sort has the worst time complexities among all algorithms analyzed above. The second slowest sort is bubble sort. Insertion sort is faster than bubble sort. Annealing sort with temperature set 2 is better than that with temperature set 1. The best sort algorithm is shell sort with gap set 2.



If the input is an almost-sorted distribution, the spin the bottle sort is still has the worst time complexities among all algorithms analyzed above. Bubble sort has the second worst time complexity. Annealing sort is faster than bubble sort. Shell sort is faster than annealing sort. The fastest sorting algorithm for almost-sorted distribution is insertion sort because its time complexity is linear and only requires a few operations.