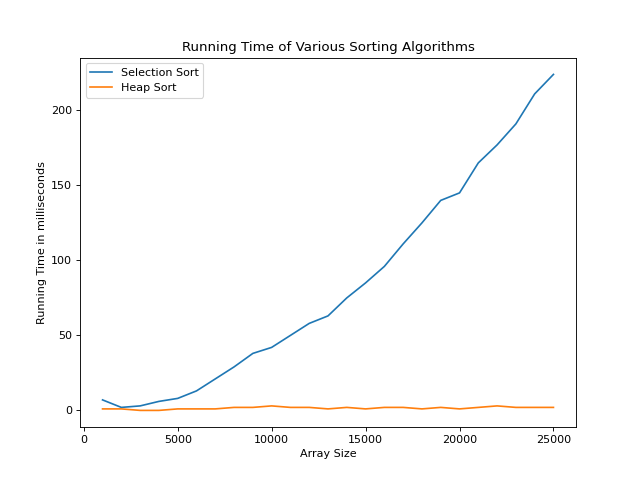
Result and Analysis of the comparison between heap sort and selection sort

Result:



The diagram demonstrates that the orange line is heap sort which is linearithmic, the blue line is selection sort which is quadratic. The result presented by diagram is corresponding to the theoretical prediction.

Analysis:

The time complexity of the selection sort:

1. The best case is O(n^2)

2. The average case is O(n^2)

3. The worst case is O(n^2)

The reason why these three cases have the same time complexity is that whatever the initial array is, they always need to compare to each other so that they can find the smallest one. Therefore, the total number of comparison is 1+2+3+….+arr.length-1, so the time complexity is O(n^2).

The time complexity of the heap sort:

1. The best case is O(nlogn)

2. The average case is O(nlogn)

3. The worst case is O(nlogn)

Heap sort can be divided into two parts which are building a max heap and then heapify it. If the node is not correspond to the attribute of the max heap, then it has to compare with its parent. Here will take lots of time to exchange and compare. Suppose there are n elements then the height of the tree is h=logn. For the i level node it has to do h-i times operation if necessary. Therefore, the worst case of is that nodes in each level need to do operation which is the sum from i=1 to h-1, where the formula is (h-i)\*2^i. Even if the heap is initially the max heap means that the best case for creating max heap, it still takes time to heapify which will be O(nlogn) because the head will be exchanged with the tail elements and the tail elements must not be the biggest one. Therefore, the tail element will go down after exchanging with the head element. Repeatedly, it will take O(nlogn) times, the analysis is same with creating the max heap.