Chong Yiern

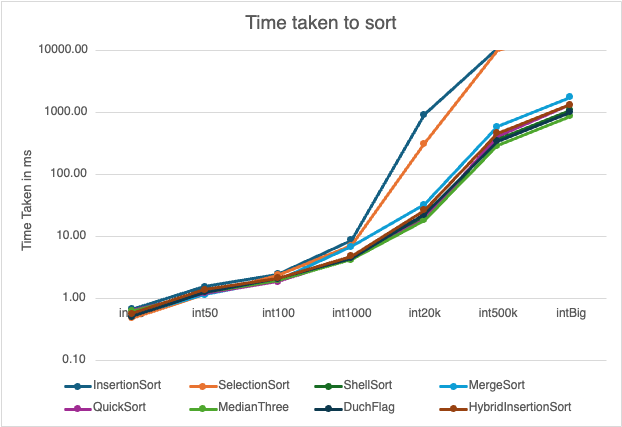
2901732C

**How code works:**

* runTimings() function runs the main timing for all the sorting algorithms
* runSpecificFile() function takes a string of the file name and does the whole loop for all sorting algorithms for only one file and gives the average time for each algorighm
* RunSpecificFunc() does the same but takes a function and only uses it for all the datasets.

**Sorting Algorithms time complexity comparisons:**

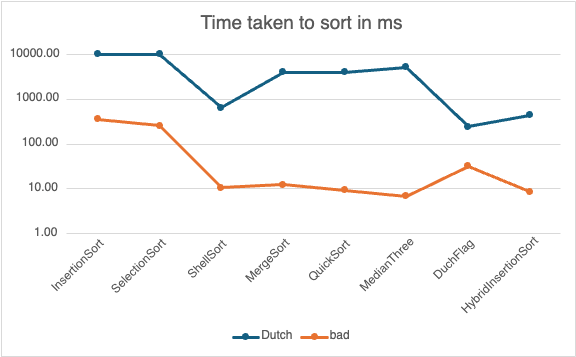
Before starting, a warmup loop is first started which runs the insertion sort 5 times in order cache functions and initialize variables. We then run each sorting algorithm with the same data set 10 times to find the average time taken with a cutoff time of 20,000ms to avoid long wait times. This is done 6 more times as there are 6 datasets.



Insertion and Selection sort both has a time complexity of O(n^2), which is why both have a similar time trend. The rest of the algorithms have a time complexity of O(n log n) which is evident in the graph. We can infer that a time complexity of O(n^2) does not do well with large data sets.

**Dutch and Bad data sets:**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Insertion Sort | Selection Sort | Shell Sort | Merge Sort | Quicksort | Median Three | Dutch Flag | Hybrid InsertionSort |
| Dutch | 10005.21 | 10005.95 | 653.49 | 3990.43 | 3979.81 | 5157.23 | 245.09 | 435.52 |
| bad | 357.010229 | 253.835854 | 10.48262 | 12.40835 | 9.317941 | 6.865191 | 32.222145 | 8.527941 |



Part 2

**Approach:**

Using the min-heap approach, we can find k most viewed videos only without any unneeded sorting time. When a new video enters the heap, it compares the least viewed video currently in the heap and replaces it if it has more views. It helps efficiently maintain the top k largest videos in the heap.

**Operates**:

Every element has to be processed individually to either be inserted into the heap or replace an element in the heap. Each operation on the heap takes O(log k) time as the heap is a balanced binary tree, insertions and extractions on the heap take log time in the size of the heap.

**Time Complexity**:

Since min heap goes through each element in array of size n, it is O(n) to traversers the whole array. And at each element, min heap starts an operation process which take O(log k), with k being the heap size. This makes the time complexity O(n log k) in any case.

**Java implantation:**

