LAB 06

ALGORITHM:

The strategy used was to build a maxheap of the given list and extract the maximum of the heap k times. Once a max heap is produced it is certain that within the first k levels of the maxheap the k largest numbers are present. By considering the previously mentioned fact the maxheap produced was reduced to k levels (if the number of items present within k levels of a heap is less than the total size of the heap) before extracting k largest items from the heap. Thus, the algorithm efficiency increases drastically.

TIME COMPLEXITY:

Assume $k \lll n \quad (2^\wedge k < n)$

Building a MaxHeap – $O(n)$

Reducing the heap to k levels - $= \Theta(1)$

Extracting max – $O\left(\lg\left(2^k\right)\right) = O(k)$. The number elements in k levels of the heap = $2^k - 1 \approx 2^k$

There k such extractions.

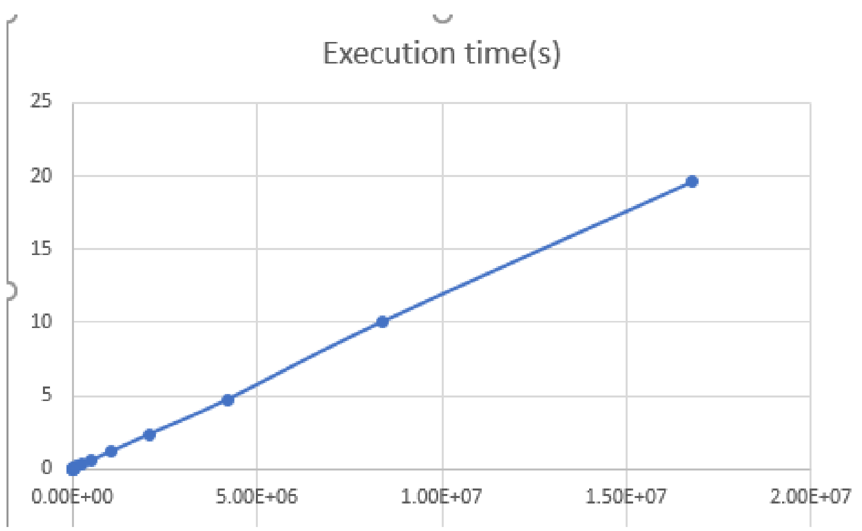
Extracting k largest items $= kO(k) = O(k^2)$

$$T(n) = O(n) + O(k^2) + \Theta(1) \approx O(n)$$

TESTING:

The testing was done using worst case data.(i.e A list where the k largest numbers are at the end). The sample sizes was doubled each time while keeping k fixed and the time was measured for extracting the k largest items.

Sample size	Execution time(s)
2^10	0
2^11	0
2^12	0.00999999
2^13	0.019999981
2^14	0.019999981
2^15	0.0400002
2^16	0.079999924
2^17	0.150000095
2^18	0.289999962
2^19	0.579999924
2^20	1.180000067
2^21	2.379999876
2^22	4.726000071
2^23	10.05100012
2^24	19.625



As shown by the graph the execution time is directly propotional to the sample size which is the same as predicted by the theoretical analyisis.