Project 2

*Shrey Jain*

# Problem Statement

We are required to analyze QuickSelect, deterministic (median of medians method) algorithm.

# Theoretical Analysis

* First, we will analyze the Time complexity of the algorithm. In this algorithm, an array A of size n and an integer of k is considered.
* Array ‘A’ is too large to find a median therefore we break it down into 5 groups which takes O(n) time. Upon dividing, each array takes O(1) time and since all arrays are of similar length it will take O(n) time overall.
* Finding medians of all the smaller arrays, n/5, also requires linear time which is O(n).
* Recursively finding the median of the medians, gives recurrence T(n/5). Recursively calling the algorithm on a smaller subarray (either T(3n/10) or T(7n/10)) as shown in the video lecture.
* This gives the recurrence relation as:
  + **T(n) = O(n) + T(n/5) + T(7n/10)**

And by substitution method T(n) becomes O(n), thus the total time complexity for the whole algorithm becomes **O(n).**

# Experimental Analysis [Github\_Code](https://github.com/shreyjaini2001/Project-2-Median-of-Medians-Algo)

* 1. **Program Listing**

def quickselect\_median\_of\_medians(array, lower, higher, k):

if higher - lower + 1 <= 5:

return sorted(array[lower:higher+1])[k-lower]

medians = []

for i in range(lower, higher + 1, 5):

group = sorted(array[i:min(i+5, higher + 1)])

medians.append(group[len(group) // 2])

median\_of\_medians\_value = quickselect\_median\_of\_medians(medians, 0, len(medians) - 1, len(medians) // 2)

pivot\_index = array.index(median\_of\_medians\_value)

array[pivot\_index], array[higher] = array[higher], array[pivot\_index]

pivot\_value = array[higher]

i = lower

for j in range(lower, higher):

if array[j] < pivot\_value:

array[i], array[j] = array[j], array[i]

i += 1

array[i], array[higher] = array[higher], array[i]

partition\_index = i

if k == partition\_index:

return array[k]

elif k < partition\_index:

return quickselect\_median\_of\_medians(array, lower, partition\_index - 1, k)

else:

return quickselect\_median\_of\_medians(array, partition\_index + 1, higher, k)

def quickselect(arr, k):

return quickselect\_median\_of\_medians(arr, 0, len(arr) - 1, k)

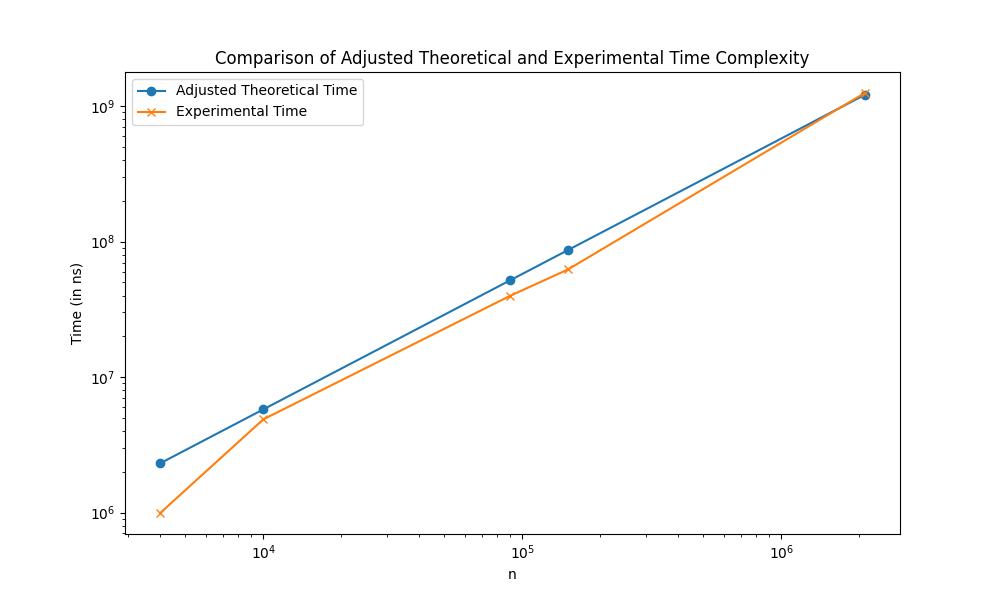
* 1. **Data Normalization Notes**

The average of Experimental results is 271760892.9 and the average of Theoretical results is 470800. Here we calculate the **Scaling factor = Avg. (Exp. result)/Avg. (Theo. result)** = **577.2321429.** It was observed that the Theoretical results were off by a factor of **577.2321429**. Therefore, to scale the values we multiply all theoretical results by the scaling constant.

* 1. **Output Numerical Data**



* 1. **Graph**

****

**3.5 Graph Observation**

The plot of experimental results seems to move away from our adjusted theoretical results for the values of n=4000, 90000, and 150000. On the other hand, the line plots of the experimental and adjusted theoretical results seem to almost completely overlap and intersect for the value of n=2100000. For the value of n=10000, the plot is close to each other but do not intersect.

# Conclusions

The convergence and intersection of the line plots for the experimental results and the adjusted theoretical results confirm that our theoretical asymptotic analysis, which determined the time complexity to be **O(n) for Quickselect, deterministic (median of medians) algorithm**, is accurate.