# Project 2

*Hetal Lad*

## 1 Problem Statement

We are required to analyze the following program/code sample.

Given an unsorted array of numbers, the task is to determine the median value. The median is defined as the middle element in a sorted list of numbers. If the list has an even number of elements, the median is the average of the two middle elements.

A straightforward solution involves sorting the array and then selecting the middle element(s), which takes **O(n log n)** time due to sorting. However, your goal is to **design and implement a more efficient algorithm** using the **Divide and Conquer** approach that finds the median in **O(n)** time.

GITHUB LINK - <https://github.com/HetalLad/CSCI6212_project1>

## 2 Theoretical Analysis

*Intuition*

The key idea is to partition the array such that elements less than the pivot are on one side and those greater are on the other. Depending on the pivot’s position, we recursively search only in the relevant half until we reach the median position. This eliminates the need to sort the entire array. The pivot can be randomly chosen, but we can increase our chances of getting the best case time, by choosing the median as the pivot point.

*Pseudocode*

Function medianOfMedians( arr,k):

Medians=[]

For i in range of (0,len(arr)) in steps of 5:

Group=arr[ i:i+5]

Sort(group)

Median= group[length(group)//2]

Medians.append(median)

pivot=medianOfMedians(medians, length(medians//2)

low= all elements in arr less than pivot

high= all elements in arr higher than pivot

equal= all elements in arr equal to pivot

if k<length(lows):

recurse on the right half of the array

else if k < length(lows) + length(pivots):

return pivot //base case

else:

recurse on the left half of the array

*Time complexity*

Let’s calculate the time complexity part by part .

* Divide Step and Sorting : Since the array is split into 5 elements, sorting takes constant time that is O(n) time.
* Median of Medians : finding the median recursively for n/5 groups takes T(n/5) time.
* Partition Step: The array is rearranged for n elements present in the array so takes O(n) times
* Recursion on reduced array:

We get n/5 medians. Let’s say p is the median of medians. Thus half the medians (n/10) are less than p and the other half (n/10) are greater than p. In each group there 3 elements less than the median p.

So there are atleast 3n/10 elements lesser than p. This also holds true for the larger side.

Therefore, the recursion is for the rest of 7n/10 elements, T(7n/10).

Thus, the T(n)= T(n/5) + T(7n/10) + cO(n)

By Induction Hypothesis, we assume that for all m<n , T(m) <= Dm

Applying recurrence we get

T(n) <= cn + D. n/5 + D. 7n/10

= cn + 9/10 D.n

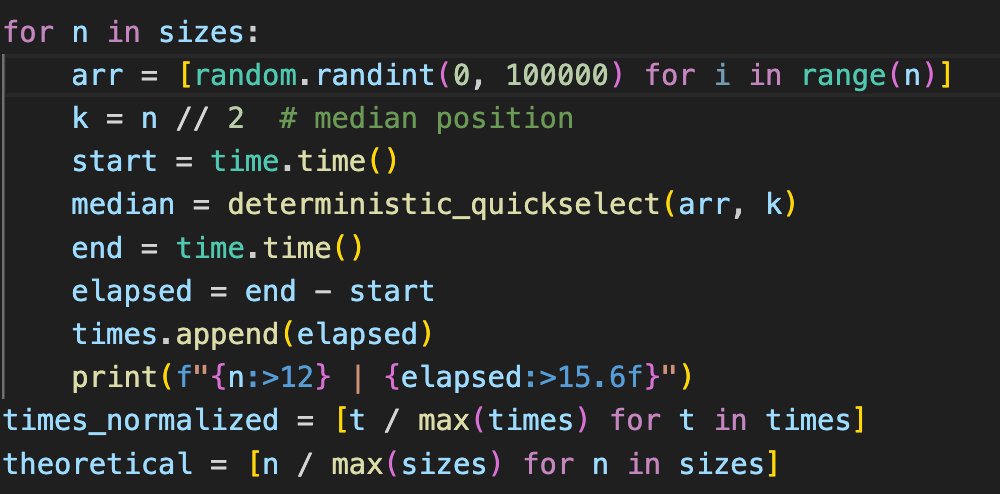
Let D=10 c

Therefore, T(n) <= 10 cn

Thus, T(n) = O(n)

## 3 Experimental Analysis

### 3.1 Program Listing



### 3.2 Data Normalization Notes

Do you normalize the values by some constant? How did you derive that constant?

Yes I have normalized my theoretical values to compare them fairly with my experiment results.

Experimentaltime\_normalized= time / max(time)

Theoretical= n/max (n).

This approach scales both the datasets to a [0,1] range providing a clear visualization of how the theoretical and experimental trends align.

### 3.3 Output Numerical Data

|  |  |  |
| --- | --- | --- |
| n | Experimental | Theoretical |
| 100 | 0 | 0.0001 |
| 500 | 0.001 | 0.0005 |
| 1000 | 0.001 | 0.001 |
| 5000 | 0.002 | 0.005 |
| 10000 | 0.004 | 0.01 |
| 20000 | 0.009 | 0.02 |
| 50000 | 0.021 | 0.05 |

**3.4 Graph**

A graph with a red line

AI-generated content may be incorrect.

**3.5 Graph Observations**

Experimental values are smaller that the theoretical estimates. This is because the theoretical values are worst-case bounds. Random arrays result in good pivot splits, reducing recursion.

The graph shows that the execution time increases approximately linearly with the input size.

## 4 Conclusions

The theoretical values are worst-case bounds, not predictions for specific randomly generated arrays. The graph grows linearly, this aligns with the O(n) theoretical time complexity of the deterministic Median of Medians Quick Select algorithm.

Note :- Most of the work has been referenced from Amrinder Arora Analysis and Design of Algorithms revised third edition.