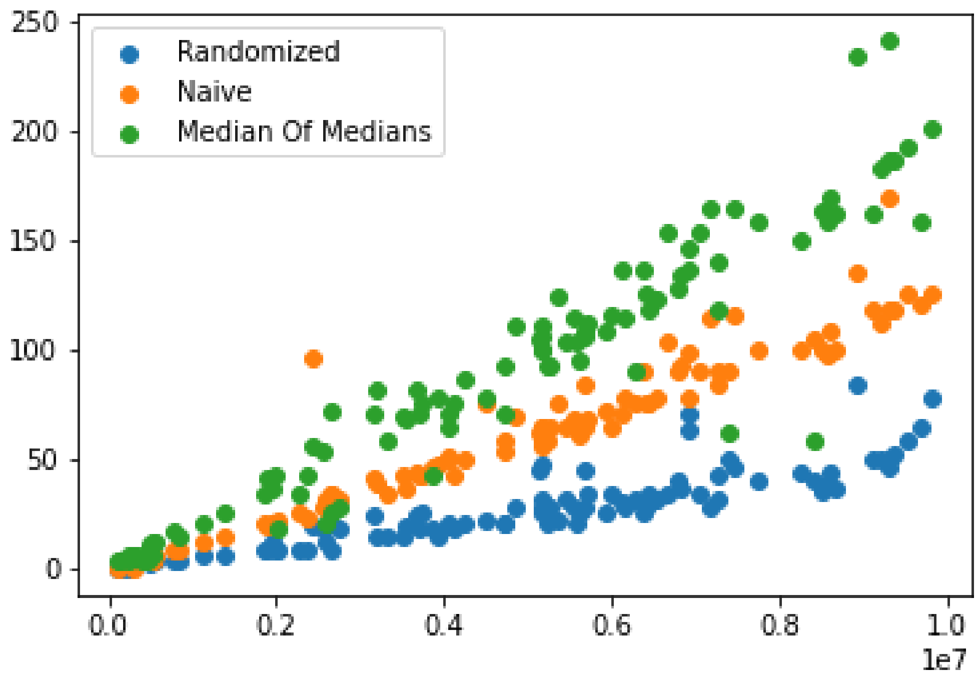
ASSIGNMENT 1

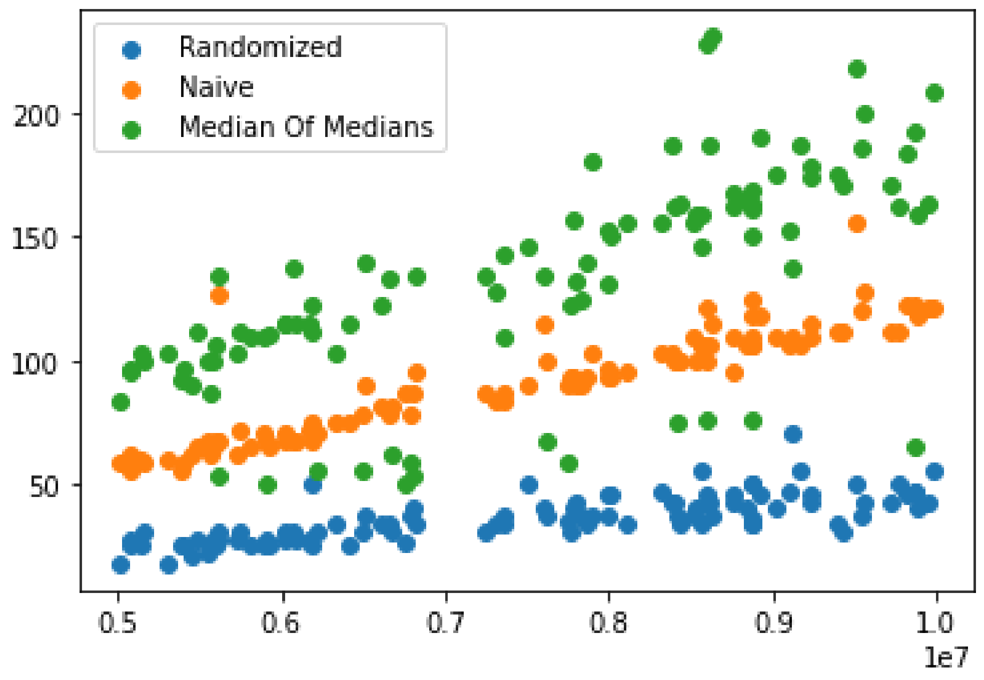
Name: Gamal Abdul Hameed Nasef Newysar

ID: 19015550

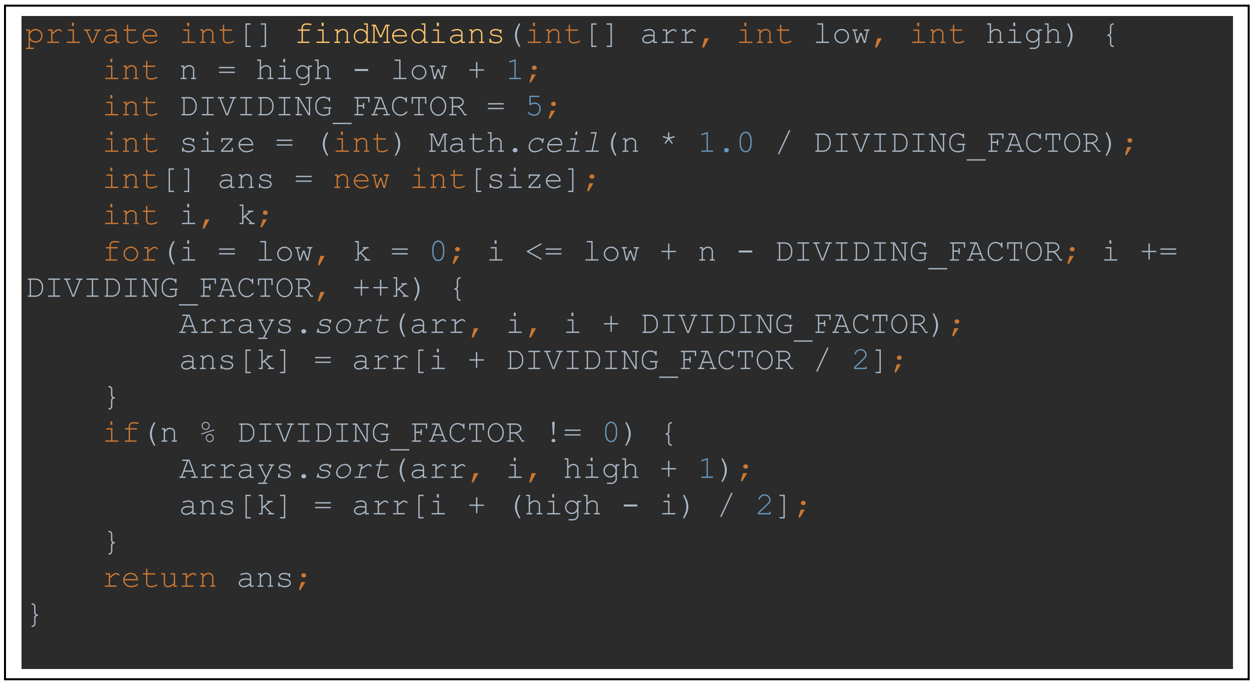
Analysis and performance comparison:

1. The Randomized Divide and Conquer Algorithm:
   It is the fastest method between the three in practice. Linear expected time on average Θ (n) but the worst case is $O(n^2)$.
2. The Deterministic Linear-time Selection Algorithm using median-of-medians:
   Linear time Θ(n) in all cases. In practice, it is the slowest method between them.
3. The Naive method using sorting and returning the k-th smallest number:
   Θ(nlgn) because of sorting. In practice, it is faster than the deterministic linear-time selection algorithm but slower than the Randomized Divide and conquer algorithm.





- Code snippets:



```
private int[] findMedians(int[] arr, int low, int high) {
    int n = high - low + 1;
    int DIVIDING_FACTOR = 5;
    int size = (int) Math.ceil(n * 1.0 / DIVIDING_FACTOR);
    int[] ans = new int[size];
    int i, k;
    for(i = low, k = 0; i <= low + n - DIVIDING_FACTOR; i +=
DIVIDING_FACTOR, ++k) {
        Arrays.sort(arr, i, i + DIVIDING_FACTOR);
        ans[k] = arr[i + DIVIDING_FACTOR / 2];
    }
    if(n % DIVIDING_FACTOR != 0) {
        Arrays.sort(arr, i, high + 1);
        ans[k] = arr[i + (high - i) / 2];
    }
    return ans;
}
```

```java
private int select(int[] arr, int target, int low, int high) {
    if(low == high)
        return arr[low];

    // 1. Divide the n elements into groups of 5. Find the
median of each 5-element group.
    int[] medians = findMedians(arr, low, high);
    // 2. Recursively SELECT the median x of the ⌊n / 5⌋ group
medians to be the pivot.
    int mediansSize = medians.length;
    int medianOfMedians;
    if(mediansSize % 2 == 0)
         medianOfMedians = select(medians, mediansSize / 2 -
1, 0, mediansSize - 1);
    else
        medianOfMedians = select(medians, mediansSize / 2, 0,
mediansSize - 1);
    // 3. Partition around the pivot x. Let k = rank(x).
    int pivot = partitionAround(arr, medianOfMedians, low,
high);
    if(pivot - low == target)
        return medianOfMedians;
    if(target < pivot - low)
        return select(arr, target, low, pivot - 1);
    return select(arr, low + target - pivot - 1, pivot + 1,
high);
}
```

```java
private int partitionAround(int[] arr, int x, int low, int
high) {
    int pivot = low;
    for(int i = low; i <= high; ++i) {
        if(arr[i] == x) { // if(Math.abs(arr[i] - x) <=
Math.pow(10, -7))
            pivot = i;
            break;
        }
    }
    helper.swap(arr, pivot, high);
    return helper.partition(arr, low, high);
}
```

```java
private int RandSelect(int[] arr, int low, int high, int
target) {
    if(low == high)
        return arr[low];
    int r = helper.RandomizedPartition(arr, low, high);
    int k = r - low;
    if(k == target)
        return arr[r];
    if(target < k)
        return RandSelect(arr, low, r - 1, target);
    return RandSelect(arr, r + 1, high, target - k - 1);
}
```

```java
public int partition(int[] arr, int low, int high) {
    int x = arr[high];
    int i = low - 1;
    for(int j = low; j < high; ++j) {
        if(arr[j] < x) {
            ++i;
            swap(arr, i, j);
        }
    }
     swap(arr, i + 1, high);
    return i + 1;
}
```

```java
public int RandomizedPartition(int[] arr, int low, int high) {
    int i = getRandomNumber(low, high);
    swap(arr, i, high);
    return partition(arr, low, high);
}
```

- ## Acknowledgment:

  “I acknowledge that I am aware of the academic integrity guidelines of this course, and that I worked on this assignment independently without any unauthorized help”.