QUICKSORT EVALUATION

In this memo, **I will show that the length of the array and the varying duplicate item WILL NOT impact which selection method to choose**. However, the degree of sortedness WILL impact which selection method to choose.

- **Experiment Method:**

In this experience, I have 3 methods of choosing a pivot: First index, Random index, and Median of 3 indexes (For now, I will call this for short as Median method)

We will compare these 3 methods in 3 types of array: Random array, Duplicate array (with level), Sorted Array (with level).

I have these variables:

- The length of the array
- The level of duplicate (How often are the duplicated items)
- The level of the sorted array (There are 4 levels: Already Sorted, Almost Sorted, Almost Reversed Sorted, Reverse Sorted)

I will evaluate the running time of the 3 methods on different arrays to see which method is the most optimal. The program will run 50 times and take the average.

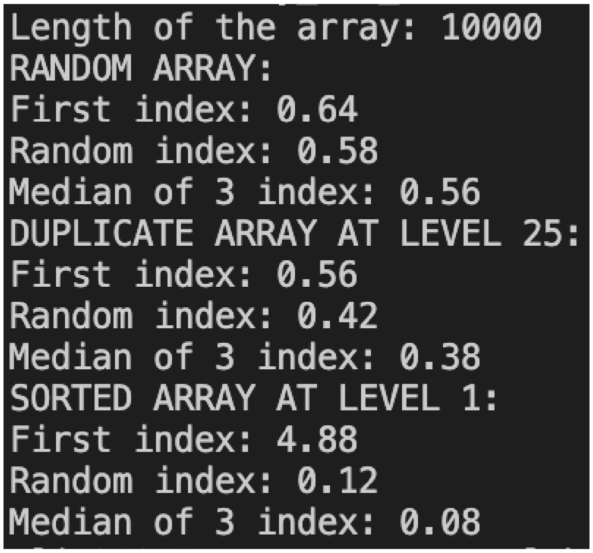
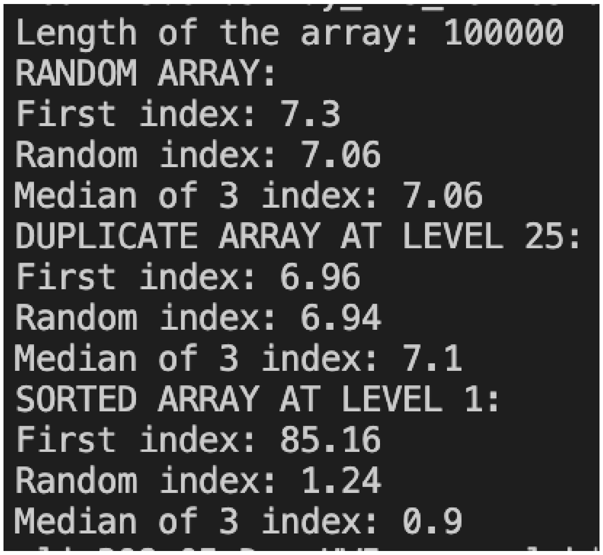
I. EVALUATE THE LENGTH OF THE ARRAY

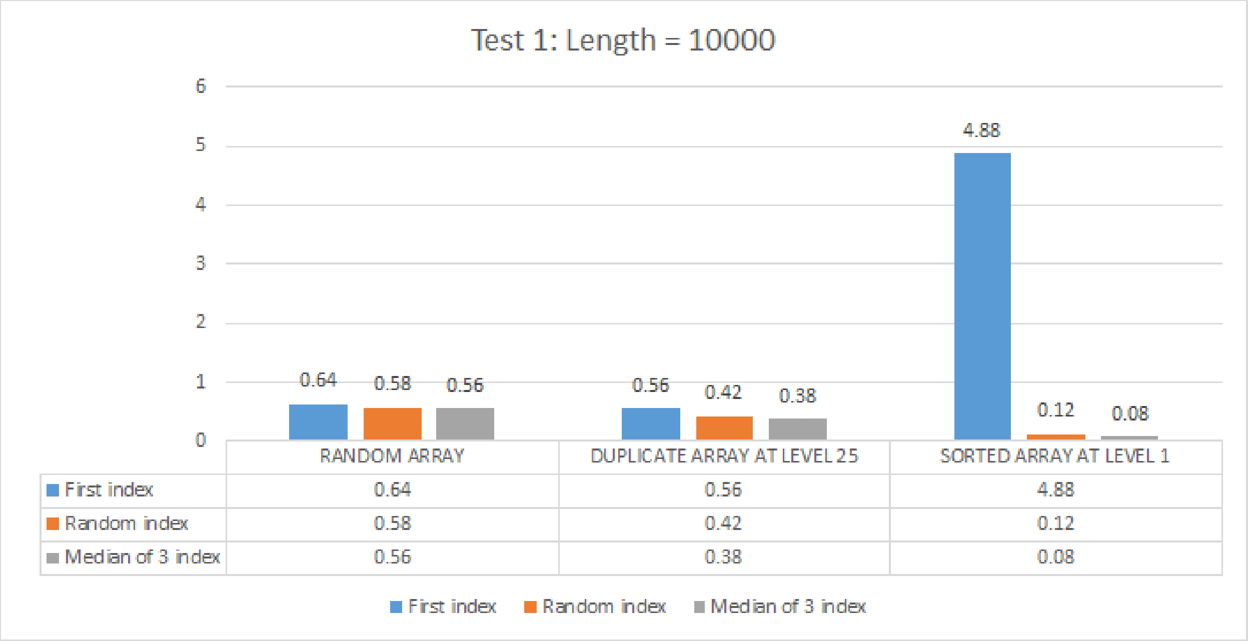
1. Experiment Method:

We will change the length of the array and print out the time it took to sort 3 mentioned different arrays in 3 different methods of choosing a pivot.

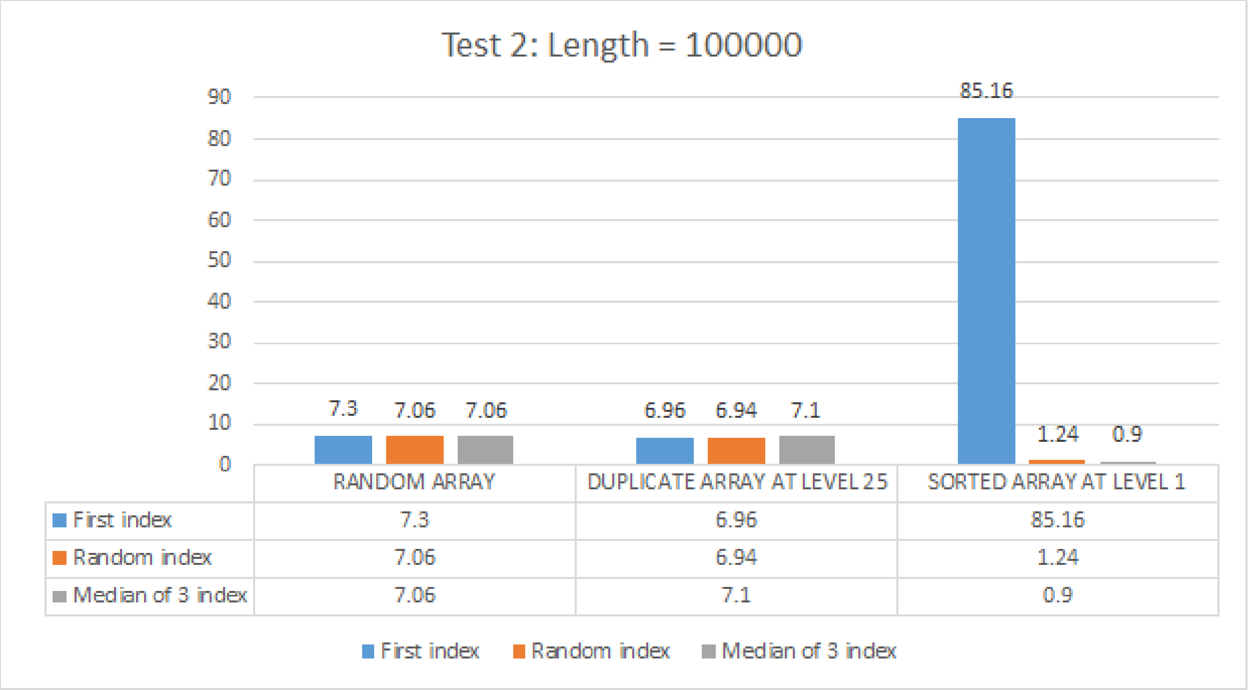
To evaluate the effect of the length, I will remain the level of duplicate and level of the sorted array (controlled variable). In this test: level of duplicate = 25 and level of sorted = 1.

2. Evidence:

Test 1: Length = 10000	Test 2: Length = 100000
 Length of the array: 10000 RANDOM ARRAY: First index: 0.64 Random index: 0.58 Median of 3 index: 0.56 DUPLICATE ARRAY AT LEVEL 25: First index: 0.56 Random index: 0.42 Median of 3 index: 0.38 SORTED ARRAY AT LEVEL 1: First index: 4.88 Random index: 0.12 Median of 3 index: 0.08	 Length of the array: 100000 RANDOM ARRAY: First index: 7.3 Random index: 7.06 Median of 3 index: 7.06 DUPLICATE ARRAY AT LEVEL 25: First index: 6.96 Random index: 6.94 Median of 3 index: 7.1 SORTED ARRAY AT LEVEL 1: First index: 85.16 Random index: 1.24 Median of 3 index: 0.9



	RANDOM ARRAY	DUPLICATE ARRAY AT LEVEL 25	SORTED ARRAY AT LEVEL 1
First index	0.64	0.56	4.88
Random index	0.58	0.42	0.12
Median of 3 index	0.56	0.38	0.08



	RANDOM ARRAY	DUPLICATE ARRAY AT LEVEL 25	SORTED ARRAY AT LEVEL 1
First index	7.3	6.96	85.16
Random index	7.06	6.94	1.24
Median of 3 index	7.06	7.1	0.9

3. Description & Conclusion

- In the Random array, at the length of 10000 (Test 1), 3 methods have an approximately similar value around 0.6. And at the length of 100000 (Test 2), 3 methods have an approximately similar value around 7. The increase in time is due to the increase in length, but all 3 methods experience the same increase. The same situation in Duplicate. In the Sorted array, in both tests, the first index has the longest time, while random and median have similar and much smaller times. Overall, They stay the same relative to each other
- **Therefore, the length WILL NOT impact which selection method to choose**.

4. Recommendation

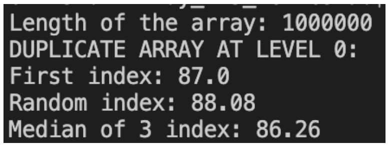
We can see that the first index always has the longest running time. So we should consider not taking the first index method. And we should consider using Median methods because most of the time it runs quicker than others. (For general, not because of the length)

II. EVALUATE THE DUPLICATE ARRAY

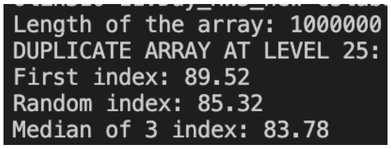
1. Experiment Method

We will change the duplicate of the array level and print out the time it took to sort in 3 different methods of choosing a pivot. (The duplicate level is the percent of the number that has duplicated items. For example, level 100 means 100% of items have the duplicate)
To evaluate the effect of the duplicate array, I will remain the length. In this test: length = 1000000.

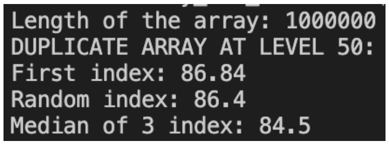
2. Evidence



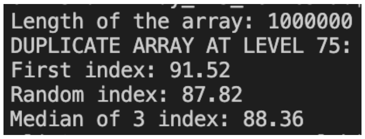
```
Length of the array: 1000000
DUPLICATE ARRAY AT LEVEL 0:
First index: 87.0
Random index: 88.08
Median of 3 index: 86.26
```



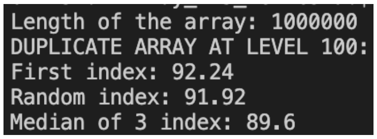
```
Length of the array: 1000000
DUPLICATE ARRAY AT LEVEL 25:
First index: 89.52
Random index: 85.32
Median of 3 index: 83.78
```



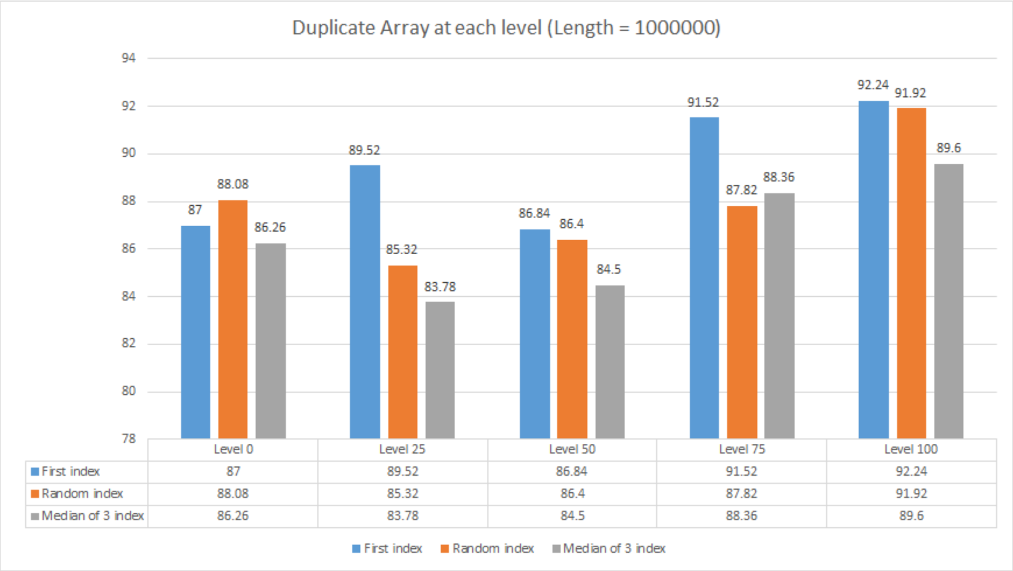
```
Length of the array: 1000000
DUPLICATE ARRAY AT LEVEL 50:
First index: 86.84
Random index: 86.4
Median of 3 index: 84.5
```



```
Length of the array: 1000000
DUPLICATE ARRAY AT LEVEL 75:
First index: 91.52
Random index: 87.82
Median of 3 index: 88.36
```



```
Length of the array: 1000000
DUPLICATE ARRAY AT LEVEL 100:
First index: 92.24
Random index: 91.92
Median of 3 index: 89.6
```



	Level 0	Level 25	Level 50	Level 75	Level 100
■ First index	87	89.52	86.84	91.52	92.24
■ Random index	88.08	85.32	86.4	87.82	91.92
■ Median of 3 index	86.26	83.78	84.5	88.36	89.6

3. Description & Conclusion

- We can see the running time of the 3 methods in different levels remains the same relative to each other. (The chart starts at 78)
- **Therefore, varying numbers of duplicate items WILL NOT impact which method to choose**.

4. Recommendation

We can see that the First index method most of the time runs slower than other methods, and the Median method most of the time runs quicker than others. Therefore we should use the Median method (For general, not because of the duplication)

III. EVALUATE THE INITIAL ORDERING (SORTEDNESS LEVEL)

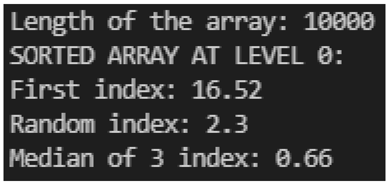
1. Experiment Method

We will change the sort level of the array and print out the time it took to sort 3 mentioned different arrays in 3 different methods of choosing a pivot.

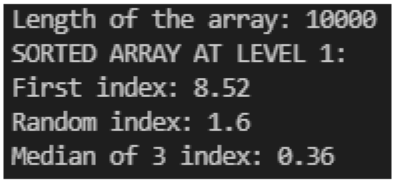
- There are 4 levels: 0 - Already Sorted, 1- Almost Sorted, 2- Almost Reversed Sorted, 3- Reverse Sorted
- In Level 1: I will sort the array in ascending order. And swap randomly a number of items. (Number of items swapped = array length/501). Same for Level 2, but in descending order.

To evaluate the effect of the sorted array, I will remain the length. In this test: length = 10000.

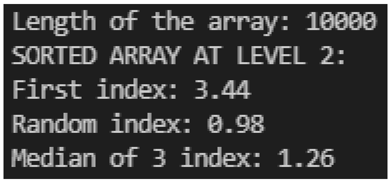
2. Evidence



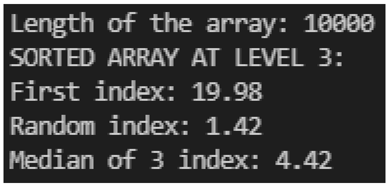
```
Length of the array: 10000
SORTED ARRAY AT LEVEL 0:
First index: 16.52
Random index: 2.3
Median of 3 index: 0.66
```



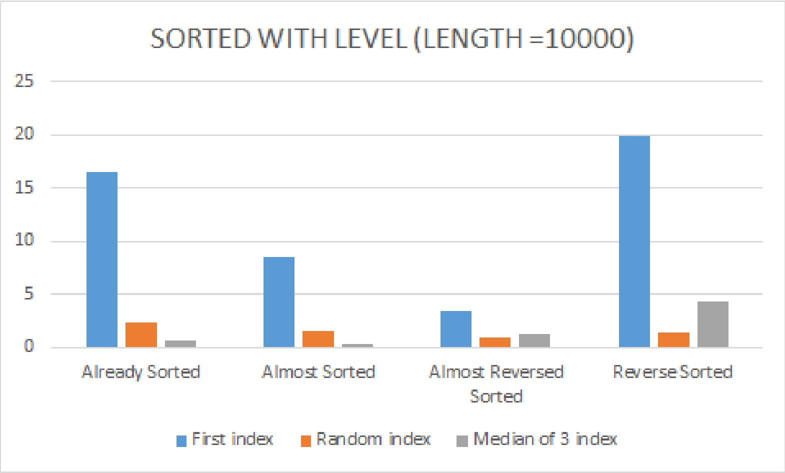
```
Length of the array: 10000
SORTED ARRAY AT LEVEL 1:
First index: 8.52
Random index: 1.6
Median of 3 index: 0.36
```



```
Length of the array: 10000
SORTED ARRAY AT LEVEL 2:
First index: 3.44
Random index: 0.98
Median of 3 index: 1.26
```

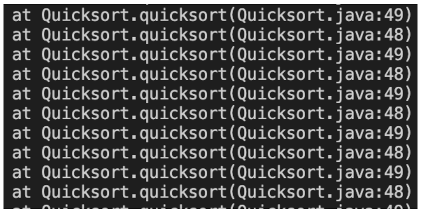


```
Length of the array: 10000
SORTED ARRAY AT LEVEL 3:
First index: 19.98
Random index: 1.42
Median of 3 index: 4.42
```

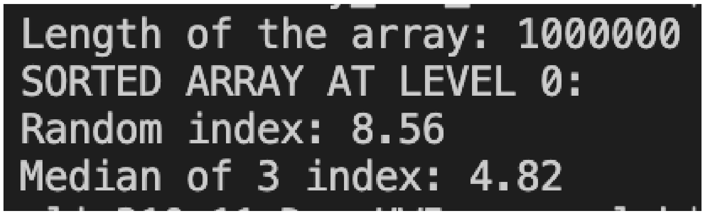


*** Stack Overflow Error:**

If I increase the length of the array to 1000000, I will meet the Stack Overflow error.
If I don't run the First Index method in the array length of 1000000, there will be no error.



```
at Quicksort.quicksort(Quicksort.java:49)
at Quicksort.quicksort(Quicksort.java:48)
at Quicksort.quicksort(Quicksort.java:49)
at Quicksort.quicksort(Quicksort.java:48)
at Quicksort.quicksort(Quicksort.java:49)
at Quicksort.quicksort(Quicksort.java:48)
at Quicksort.quicksort(Quicksort.java:49)
at Quicksort.quicksort(Quicksort.java:48)
at Quicksort.quicksort(Quicksort.java:49)
at Quicksort.quicksort(Quicksort.java:48)
```



```
Length of the array: 1000000
SORTED ARRAY AT LEVEL 0:
Random index: 8.56
Median of 3 index: 4.82
```

Therefore, in a large array, the First index will have too much run time and cause a runtime error.

3. **Description & Conclusion**

- We can see that the First index method (8-20) runs significantly slower than the other one (0-5). Even in the larger array (length = 1000000), the First index method will cause Stack Overflow.
- Also in Level 0, 1 (Sorted, Almost Sorted), the Median method, for most of the time, is the fastest one. In Level 2,3 (Almost Reverse Sorted, Reversed Sorted), the Median, for most of the time, runs slower than the Random method, but still significantly faster than the First Index
- Therefore, varying amounts of sortedness (in both directions) WILL impact which method to choose.

4. **Recommendation**

- In (Almost) Sorted or (Almost) Reverse Sorted, we should not use the First index method. Because, for most of the time, it will pick the smallest/ largest pivot, which is not efficient in Quicksort.
- Instead, we should use the Median Method (if the array is (Almost) Sorted) or the Random Method (if the array is (Almost) Reverse Sorted).

PROMPT:

- **Why do you think the timing data looks the way it does base on what you know about how the algorithm works?**
  + The First Index is usually the slowest because, for most of the time, it will pick the smallest/ largest pivot, which is not efficient in Quicksort.
  + The Median is usually the fastest because, for most of the time, it will pick the median or not the largest/smallest pivot, which is more efficient in Quicksort.
- **How could you compensate for the fact that timing data is "noisy," with the same code sometimes taking a little longer or shorter to run?**
  + The data is "noisy" because of the randomness in making the array and choosing pivot (Random method)
  + Sometimes, The First Index is faster than others, because it picks the more efficient pivot (closer to median) for most of the time. Or sometimes, the Random is slower than others, because it picks the less efficient pivot (closer to smallest/biggest) for most of the time. Or, it will be the fastest if reverse. Sometimes, The Median is slowest because it picks the less efficient pivot (closer to smallest/biggest) for most of the time.