Chapter 5 Programming - In Practice

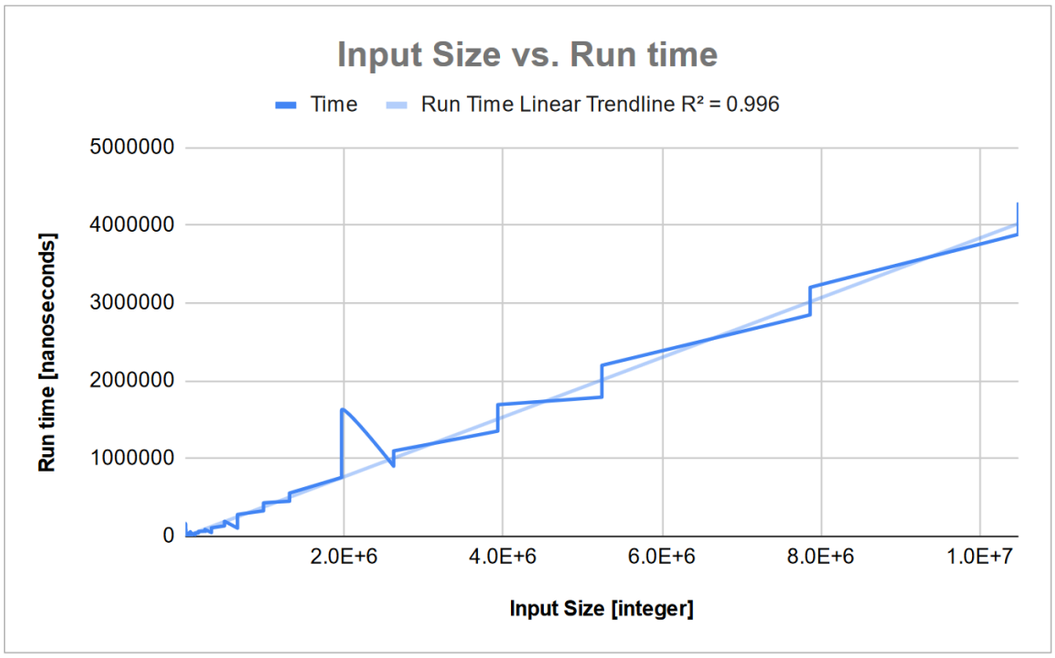
5.30 Method

```java
public static boolean doesTheArrayMeetCriteria_5_30(int [] a) {
    // Loop through the array and check each element one time

    for (int i = 0; i < a.length; i++)
    {
        // If they are equal then the array meets the criteria
        if (a[i] == i)
        {
            return true;
        }
    }

    return false;
}
```

5.30 Output

```
c:\Users\wes\github-repos\cs2420_summer2023\Chapter5 - VS Code Console
Starting 5.30 Test
Finished testing 5.30
Press any key to continue . . .
```

5.30 Analysis



- The run time of this algorithm increases at a linear rate, which means it has a Big-O time of O(x).

Chapter 5 Programming - In Practice

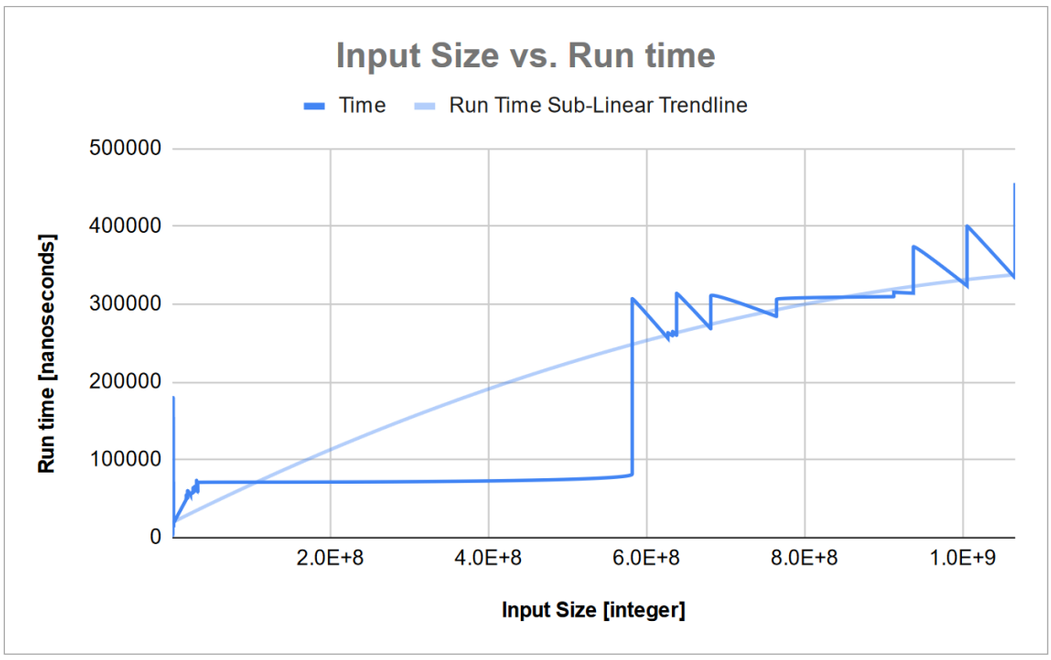
5.31 Method

```
19      public static boolean isAPrime(int num) {
20
21          // Search up to the sqrt of the number starting at 2
22          for (int i = 2; i <= Math.sqrt(num); i++)
23          {
24              // If the remainder is 0 then exit
25              if (num % i == 0)
26              {
27                  return false;
28              }
29          }
30
31          return true;
32      }
```

5.31 Output

```
c:\Users\wes\github-repos\cs2420_summer2023\Chapter5 - VS Code Console
Starting Prime Test
Finished testing primes
Press any key to continue . . .
```

5.31 Analysis



- The run time of this algorithm is increasing at a rate of root x or x^(1/2). I believe that the Big-O time of this algorithm is O(x^½) which would be better than linear time.

Chapter 5 Programming - In Practice

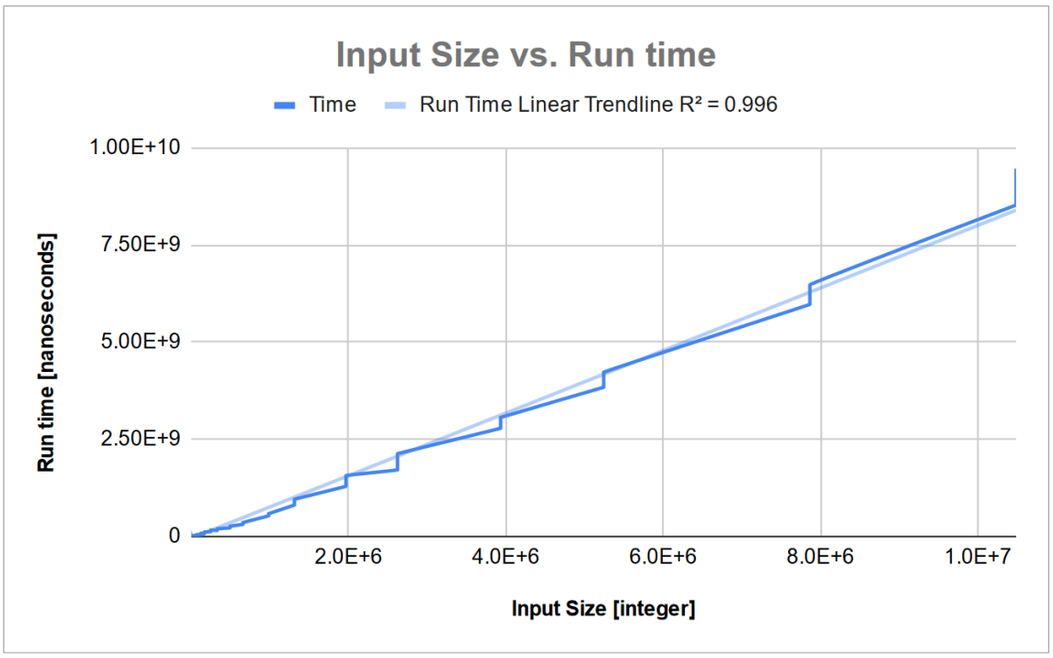
5.33 Method

```java
public static boolean majorityElement(int [] _a) {

    // Create a map to hold the count for each number
    Map<Integer, Integer> frequency = new HashMap<Integer, Integer>();

    // Loop through the whole array once
    for (int i = 0; i < _a.length; i++)
    {
        // Get the Number and initilize the count
        Integer value = _a[i];
        Integer amount = 0;

        // Update the amount if we've already found this number
        if (frequency.containsKey(value))
        {
            amount = frequency.get(value);
        }
        // Add one to the count
        amount += 1;

        // Exit if the count is larger than _a.length / 2
        if (amount > _a.length / 2)
        {
            return true;
        }
        // Update the amount
        frequency.put(value, amount);
    }

    return false;
}

}
```

5.33 Output

```
c:\Users\wes\github-repos\cs2420_summer2023\Chapter5 - VS Code Console
Starting majority element test
Finished testing majority element
Press any key to continue . . .
```

5.33 Analysis



- The run time of this algorithm increases at a linear rate, which means it has a Big-O time of O(x).