Lab 2: Benchmarking

Data Structures and Algorithms

This lab is going to have you run benchmark on isPrime0, isPrime1, and isPrime2 mentioned in lectures.

For compute $\pi(100)$, we can use the following program:

CountPiN.java

```
public class CountPiN {
    static boolean isPrime0(int n) {
        if(n==1) return false;
        if(n<=3) return true;
        int m = n/2;
        for(int i=2; i<=m; i++) {
            if(n%i==0) return false;
        }
        return true;
    }
    public static void main(String[] args) {
        int count = 0;
        int N = 100;
        for(int n=1; n<N; n++) {
            if(isPrime0(n)) count++;
        }
        System.out.println("Pi("+N+")="+count);
    }
}
```

The output look like this:

```
>java CountPiN
Pi(100)=25
```

The method isPrime0(n) take any positive integer and return true if it is a prime, false otherwise. The method run through all integer from 2 to n/2 and check if n is divisible by any of them.

There are two more methods, isPrime1(n) and isPrime2(n). The method isPrime1(n) is similar to isPrime0(n) but only run from 2 to $\sqrt{n}$. The method isPrime2(n) improves upon

isPrime1(n) by take out anything divisible by 2 and 3 and not going to test divisibility of number that are multiple of 2 and 3.

Method isPrime1(n) and isPrime2(n)

```
static boolean isPrime1(int n) {
     if(n==1) return false;
     if(n<=3) return true;
     int m = (int)Math.sqrt(n);
     for(int i=2; i<=m; i++) {
          if(n%i==0) return false;
     }
     return true;
}
```

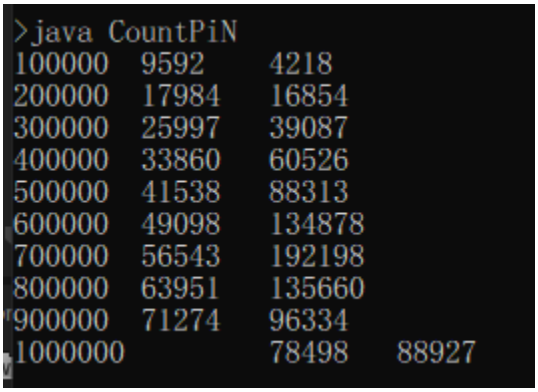
```
static boolean isPrime2(int n) {
     if(n==1) return false;
     if(n<=3) return true;

     if((n%2==0)||(n%3==0)) return false;
     int m = (int)Math.sqrt(n);
     for(int i=5; i<=m;i+=6) {
          if(n%i==0) return false;
          if(n%(i+2)==0) return false;
     }
     return true;
}
```

To measure efficiency of these methods, we must modify our main method like this

New main()

```
public static void main(String[] args) {
     for(int N=100000; N<=1000000; N+=100000) {
          long start = System.currentTimeMillis();
          int count = 0;
          for(int n=1; n<N; n++) {
               if(isPrime0(n)) count++;
          }
          long time = (System.currentTimeMillis()-
start);
          System.out.println(N+" \t"+count+" \t"+time);
     }
}
```

After the modification, the result of running with isPrime0(n) should be like this:



Your first task: run the program with isPrime0, isPrime1, and isPrime2 and record your result into the following table

Running-time table				
n	pi(n)	time (milliseconds)		
		isPrime0	isPrime1	isPrime2
100,000				
200,000				
300,000				
400,000				
500,000				
600,000				
700,000				
800,000				
900,000				
1,000,000				

Your second task: Plot two graphs, one is n vs. isPrime0's time and the other is n vs. isPrime1's time and isPrime2's time.

n vs. isPrime0	n vs. isPrime1 and isPrime2

Your final task: In your own words, describe trend of isPrime1, isPrime2, and isPrime3. Are your recorded times faster or slower than the recorded time shown in the lecture? Why?

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

Hand in your work in MS Team assignment by fill in the answer in this file. Change the name of this file to assignment1_xxxxxxx.pdf where xxxxxx is your student id.