Lecture 2

Benchmark

Teera Siriteerakul

1

Benchmark

- Good for measuring finished products.
- We see them a lot in hardware testing.

2

Key Properties

- Relevance: Focus on vital features.
- Representativeness: Accepted performance metrics.
- Equity: Should be fairly compared.
- Repeatability: Can be verified.
- Cost-effectiveness: Should be economical.
- Scalability: Should be able to test all range of system.
- Transparency: Should be easy to understand.

3

Benchmark Test Case

- $\pi(n)$ is a function that return number of prime
- For example: $\pi(100)$ = 25
- The following program compute $\pi(100)$

4

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);
    }
}
```


5

Another two algorithms

```
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;
}
```


6

Modified main() method

```
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);
      }
}
```

isPrime1(n), isPrime2(n)

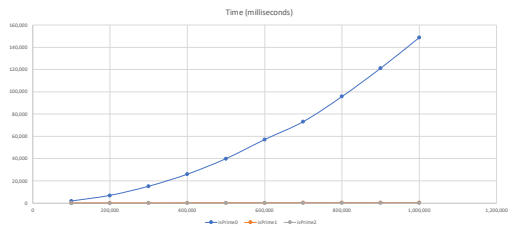
7

Recorded Results

N	$\pi(n)$	Times(milliseconds)		
		isPrime0	isPrime1	isPrime2
100,000	9,592	1,828	38	26
200,000	17,984	6,927	66	29
300,000	25,997	15,153	108	37
400,000	33,860	26,004	163	54
500,000	41,538	39,993	219	75
600,000	49,098	57,139	280	94
700,000	56,543	73,301	353	118
800,000	63,951	95,851	419	139
900,000	71,274	121,327	492	141
1,000,000	78,498	148,958	576	164

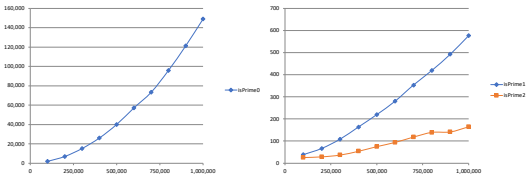
8

Performance Graph



9

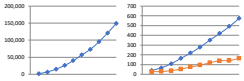
Performance Graphs



10

Observations

- As N grows, the computing time is longer.
- isPrime0 is noticeably slower, comparing to the other two
- isPrime1 and isPrime2 are comparatively similar
- We can safely say that isPrime0 is inferior to the other two
- However, if our program only need to compute $\pi(n)$ where n is relatively small, and only for a few times, any methods will do.



11

Limitation of Benchmark

For comparing algorithms, using benchmark has its limitations. We need to be careful in these issues:

- Must be done in the same environment, including hardware, operating system, selected computer language, etc.
- Implementation details should be the same
- May not reflect the real environment
- May not reflect size of data, especially in the future.

12

Benchmark: Summary

- Benchmark is good for evaluating and comparing finished products.
- There are some key properties and limitations to consider when performing benchmark.

13

14