Lecture 2

Operations Counting

Teera Siriteerakul

1

Estimating Time by Counting Operations

- We will start by assuming that one command, one programming statement, takes a fixed amount of time.
  - If we have m statement, it will take km milliseconds to follow them.
  - Linearly proportional to each other.
- For simplicity, we will assume that any statement takes the same amount time to compute.
  - For example, `x=3;` or `Math.abs(-178)` are consider to take the same amount of time to compute.
  - This might not be true, but good enough for our purpose.
  - Note that `int m=n/2;` is count as 3 operations: declaration, assignment, and division.
- Let start counting

2

isPrime0()

	code	operation count
1	`static boolean isPrime0(int n) {`	
2	`if(n==1) return false;`	1
3	`if(n<=3) return true;`	1
4	`int m = n/2;`	3
5	`for(int i=2; i<=m; i++) {`	2+m+(m-1) = 2m+1
6	`if(n%i==0) return false;`	2(m-1)
7	`}`	
8	`return true;`	1
9	`}`	
	total	$= 4m + 5$
		$= 2n + 5$

3

Let Count isPrime1()

<table>
<tr><th></th><th>code</th><th>operation count</th></tr>
<tr><td>1</td><td><code>static boolean isPrime1(int n) {</code></td><td></td></tr>
<tr><td>2</td><td><code>if(n==1) return false;</code></td><td>1</td></tr>
<tr><td>3</td><td><code>if(n<=3) return true;</code></td><td>1</td></tr>
<tr><td>4</td><td><code>int m = (int)Math.sqrt(n);</code></td><td>4</td></tr>
<tr><td>5</td><td><code>for(int i=2; i<=m; i++) {</code></td><td>2+m+(m-1) = 2m+1</td></tr>
<tr><td>6</td><td><code>if(n%i==0) return false;</code></td><td>2(m-1)</td></tr>
<tr><td>7</td><td><code>}</code></td><td></td></tr>
<tr><td>8</td><td><code>return true;</code></td><td>1</td></tr>
<tr><td>9</td><td><code>}</code></td><td></td></tr>
<tr><td></td><td>total</td><td>$= 4m + 6$</td></tr>
<tr><td></td><td></td><td>$= 4\sqrt{n} + 6$</td></tr>
</table>

4

isPrime2()

<table>
<tr><th></th><th>code</th><th>operation count</th></tr>
<tr><td>1</td><td><code>static boolean isPrime2(int n) {</code></td><td></td></tr>
<tr><td>2</td><td><code>if(n==1) return false;</code></td><td>1</td></tr>
<tr><td>3</td><td><code>if(n<=3) return true;</code></td><td>1</td></tr>
<tr><td>4</td><td><code>if((n%2==0)||(n%3==0)) return false;</code></td><td>5</td></tr>
<tr><td>5</td><td><code>int m = (int)Math.sqrt(n);</code></td><td>4</td></tr>
<tr><td>6</td><td><code>for(int i=5; i<=m; i+=6) {</code></td><td>$2+(\frac{m}{6}+1)+\frac{m}{6} = 2\frac{m}{6}+3$</td></tr>
<tr><td>7</td><td><code>if(n%i==0) return false;</code></td><td>$2\frac{m}{6}$</td></tr>
<tr><td>8</td><td><code>if(n%(i+2)==0) return false;</code></td><td>$3\frac{m}{6}$</td></tr>
<tr><td>9</td><td><code>}</code></td><td></td></tr>
<tr><td>10</td><td><code>return true;</code></td><td>1</td></tr>
<tr><td>11</td><td><code>}</code></td><td></td></tr>
<tr><td></td><td>total</td><td>$= 7\frac{m}{6} + 15$</td></tr>
<tr><td></td><td></td><td>$= 7\frac{\sqrt{n}}{6} + 15$</td></tr>
</table>

5

Operation Function

- These are functions of number operations where n is the size of the input

$$f_0(n) = 2n + 5$$

$$f_1(n) = 4\sqrt{n} + 6$$

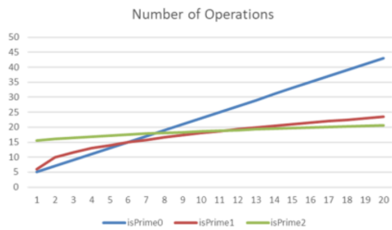
$$f_2(n) = 7\frac{\sqrt{n}}{6} + 15$$

Let call them Operation Function

- Let plot some graph

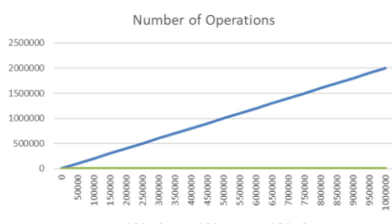
6

Small input (n≤20)



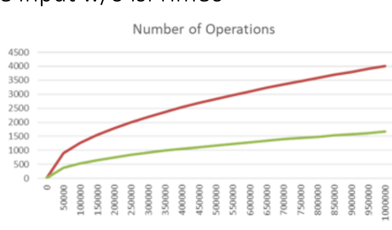
7

Large Input



8

Large Input w/o isPrime0



9

Some Discussion

$f_0(n) = 2n + 5 \qquad f_1(n) = 4\sqrt{n} + 6 \qquad f_2(n) = 7\frac{\sqrt{n}}{6} + 15$

- The result of statement counting is similar to our benchmark results.
  - isPrime0 grows at the same rate as the input.
  - isPrime1 and isPrime2 grows proportion to the square root of the input.
  - isPrime1 and isPrime2 are comparable, while isPrime0 is by far the slowest.
- It is typically possible to count number of operations of algorithms.
- It is also applicable to counting space.
- While the actual time depends on machines, number of operations does not.
  - Make it good for directly compare algorithms
- Rather than remember the functions of all known algorithms, we will compare them to well-known functions using **Asymptotic Analysis.**

10

Summary

- Rather than measuring the time, we can count number of operations in algorithms
- Number of operations are just an estimate number, which is good enough for our purpose.
- If we know the function of operations of algorithms, we can compare them easily
- However, we will not have to memorize the exact function of operations of each algorithm, we will use **Asymptotic Analysis** to compare them to well-known function.

11