Chapter 7: Recursion

EECS 1030

moodle.yorku.ca

Powers of 3

```
/**
 * Returns 3 raised to the given power.
 *
 * @param n a number.
 * @pre. n >= 0
 */
public static BigInteger pow3(int n)
```

Powers of 3: Iteration

```
BigInteger power = BigInteger.ONE;
for (int i = 0; i < n; i++)
{
   power = power.multiply(THREE);
}
```

Powers of 3: Recursion – version 1

```
BigInteger power;
if (n == 0)
{
   power = BigInteger.ONE;
}
else
{
   power = pow3(n - 1).multiply(THREE);
}
```

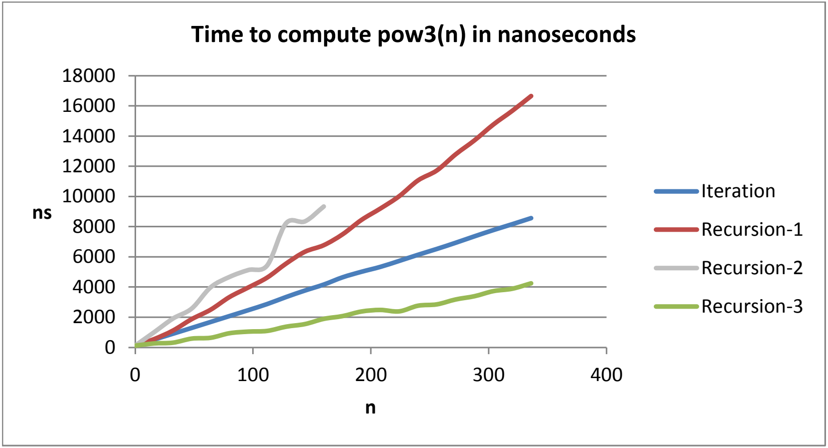
Powers of 3: Recursion – version 2

```
BigInteger power;
if (n == 0)
{
   power = BigInteger.ONE;
}
else if (n % 2 == 1)
{
   power = pow3(n - 1).multiply(THREE);
}
else
{
   power = pow3(n / 2).multiply(pow3(n / 2));
}
```

Powers of 3: Recursion – version 3

```
BigInteger power;
if (n == 0)
{
   power = BigInteger.ONE;
}
else if (n % 2 == 1)
{
   power = pow3(n - 1).multiply(THREE);
}
else
{
   BigInteger temp = power(n / 2);
   power = temp.multiply(temp);
}
```

Experimental comparison



Estimating the number of elementary actions

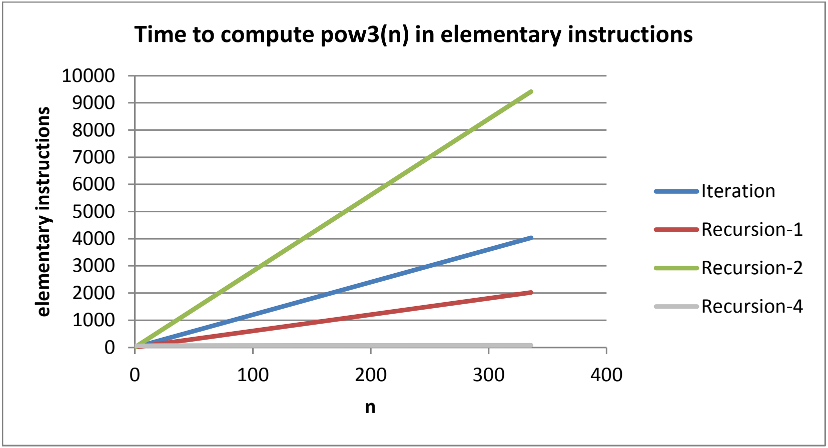
$$I(n) = 12n + 2$$

$$\left.\begin{array}{l} R_1(0) = 6 \\ R_1(n) = R_1(n-1) + 6 \end{array}\right\} R_1(n) = 6n + 6$$

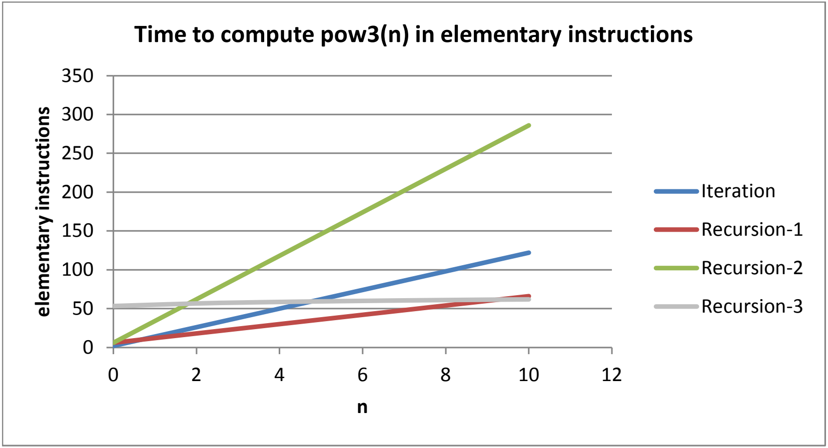
$$\left.\begin{array}{l} R_2(0) = 6 \\ R_2(n) = \begin{cases} R_2(n-1) + 11 & \text{if } n \text{ is odd} \\ 2R_2(n/2) + 11 & \text{if } n \text{ is even} \end{cases} \end{array}\right\} \begin{array}{l} R_2(n) \leq 28n + 6 \\ R_2(n) \geq 11n + 6 \end{array}$$

$$\left.\begin{array}{l} R_3(0) = 6 \\ R_3(n) = \begin{cases} R_3(n-1) + 11 & \text{if } n \text{ is odd} \\ R_3(n/2) + 11 & \text{if } n \text{ is even} \end{cases} \end{array}\right\} \begin{array}{l} R_3(n) \leq 11 \log_2(n+2) + 50 \\ R_3(n) \geq 6 \log_2(n+2) \end{array}$$

Theoretical comparison



Theoretical comparison



But those are just estimates ...

Question

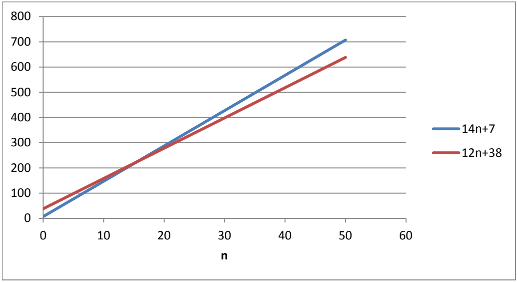
I estimate

$$R(n) = 14n + 7$$

and you estimate

$$R(n) = 12n + 38$$

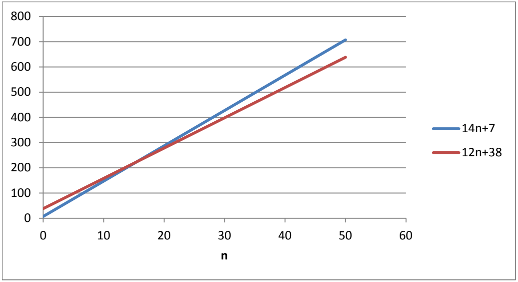
How can these estimates be of any use?



But those are just estimates ...

Answer

To compare different algorithms we are only interested in the shape of the graphs, not the actual numbers.



Running time comparison: first attempt

Definition

Let $f : \mathbb{N} \to \mathbb{N}$ and $g : \mathbb{N} \to \mathbb{N}$ estimate the number of elementary instructions executed by two algorithms. We say that f is[a] at least as good as g if

there exists a $F > 0$, such that for all $n \geq 0$, $f(n) \leq F \times g(n)$

[a] Instead of "is", "scales" might be even a better word here.

Running time comparison: first attempt

Definition

Let $f : \mathbb{N} \rightarrow \mathbb{N}$ and $g : \mathbb{N} \rightarrow \mathbb{N}$ estimate the number of elementary instructions executed by two algorithms. We say that f is[a] at least as good as g if

there exists a $F > 0,$ such that for all $n \geq 0, f(n) \leq F \times g(n)$

[a] Instead of "is", "scales" might be even a better word here.

Notation

Instead of

there exists a $F > 0,$ such that for all $n \geq 0, f(n) \leq F \times g(n)$

we write

$$\exists F > 0 : \forall n \geq 0 : f(n) \leq F \times g(n)$$

Running time comparison: first attempt

Claim

Recall that $I(n) = 12n + 2$ and $R_1(n) = 6n + 6$. I is at least as good as R_1.

Running time comparison: first attempt

Claim

Recall that $I(n) = 12n + 2$ and $R_1(n) = 6n + 6$. I is at least as good as R_1.

Proof

We have to show that

$$\exists F > 0 : \forall n \geq 0 : I(n) \leq F \times R_1(n)$$

We pick $F = 2$. Let $n \geq 0$. Then

$$\begin{array}{rcl} I(n) & = & 12n + 2 \\ & \leq & 12n + 12 \\ & = & 2 \times (6n + 6) \\ & = & F \times R_1(n). \end{array}$$

Running time comparison: first attempt

Claim

Recall that $I(n) = 12n + 2$ and $R_1(n) = 6n + 6$. R_1 is at least as good as I.

Running time comparison: first attempt

Claim

Recall that $I(n) = 12n + 2$ and $R_1(n) = 6n + 6$. R_1 is at least as good as I.

Proof

We have to show that

$$\exists F > 0 : \forall n \geq 0 : R_1(n) \leq F \times I(n)$$

We pick $F = 3$. Let $n \geq 0$. Then

$$\begin{array}{rcl} R_1(n) & = & 6n + 6 \\ & \leq & 36n + 6 \\ & = & 3 \times (12n + 2) \\ & = & F \times I(n). \end{array}$$

Running time comparison: first attempt

Claim

Recall that $R_1(n) = 6n + 6$ and $11n + 6 \leq R_2(n) \leq 28n + 6$. R_1 is at least as good as R_2.

Running time comparison: first attempt

Claim

Recall that $R_1(n) = 6n + 6$ and $11n + 6 \leq R_2(n) \leq 28n + 6$. R_1 is at least as good as R_2.

Proof

We have to show that

$$\exists F > 0 : \forall n \geq 0 : R_1(n) \leq F \times R_2(n)$$

We pick $F = 1$. Let $n \geq 0$. Then

$$\begin{array}{rcl} R_1(n) & = & 6n + 6 \\ & \leq & 11n + 6 \\ & = & 1 \times (11n + 6) \\ & \leq & F \times R_2(n). \end{array}$$

Running time comparison: first attempt

Claim

Recall that $R_1(n) = 6n + 6$ and $11n + 6 \leq R_2(n) \leq 28n + 6$. R_2 is at least as good as R_1.

Running time comparison: first attempt

Claim

Recall that $R_1(n) = 6n + 6$ and $11n + 6 \leq R_2(n) \leq 28n + 6$. R_2 is at least as good as R_1.

Proof

We have to show that

$$\exists F > 0 : \forall n \geq 0 : R_2(n) \leq F \times R_1(n)$$

We pick $F = 5$. Let $n \geq 0$. Then

$$\begin{array}{rcl} R_2(n) & = & 28n + 6 \\ & \leq & 30n + 30 \\ & = & 5 \times (6n + 6) \\ & \leq & F \times R_1(n). \end{array}$$

Running time comparison: first attempt

Claim

Recall that $R_1(n) = 6n + 6$ and $6 \log_2(n+2) \leq R_3(n) \leq 11 \log_2(n+2) + 50$. R_3 is at least as good as R_1.

Running time comparison: first attempt

Claim

Recall that $R_1(n) = 6n + 6$ and
$6\log_2(n+2) \leq R_3(n) \leq 11\log_2(n+2) + 50$. R_3 is at least as good as R_1.

Proof

We have to show that

$$\exists F > 0 : \forall n \geq 0 : R_3(n) \leq F \times R_1(n)$$

We pick $F = 11$. Let $n \geq 0$. Then

$$\begin{array}{rcl} R_3(n) & \leq & 11\log_2(n+2) + 50 \\ & \leq & 66n + 66 \\ & = & 11 \times (6n + 6) \\ & \leq & F \times R_1(n). \end{array}$$

Running time comparison: first attempt

Claim

Recall that $R_1(n) = 6n + 6$ and $6 \log_2(n+2) \leq R_3(n) \leq 11 \log_2(n+2) + 50$. R_1 is not at least as good as R_3.

Running time comparison: first attempt

Claim

Recall that $R_1(n) = 6n + 6$ and $6\log_2(n+2) \leq R_3(n) \leq 11\log_2(n+2) + 50$. R_1 is not at least as good as R_3.

Proof

We have to show that

$$\forall F > 0 : \exists n \geq 0 : R_1(n) > F \times R_3(n)$$

Let $F > 0$. Pick $n = 2^{F+10}$. Then

$$\begin{array}{rcl} R_1(n) & = & 6n + 6 \\ & = & 6 \times 2^{F+10} + 6 \\ & > & F \times (11(\log_2(2^{F+10} + 2) + 50) \\ & = & F \times (11\log_2(n+2) + 50) \\ & \geq & F \times R_3(n). \end{array}$$

Running time comparison: first attempt

Question

I, R_1 and R_2 are all comparable. What do these functions have in common?

Running time comparison: first attempt

Question

I, R_1 and R_2 are all comparable. What do these functions have in common?

Answer

They are all linear functions. A function $f : \mathbb{N} \to \mathbb{N}$ is linear if

$$\exists a > 0 : \exists b \geq 0 : f(n) = an + b.$$

Running time comparison: first attempt

Question

I, R_1 and R_2 are all comparable. What do these functions have in common?

Answer

They are all linear functions. A function $f : \mathbb{N} \to \mathbb{N}$ is linear if

$$\exists a > 0 : \exists b \geq 0 : f(n) = an + b.$$

Question

What is the "simplest" linear function?

Running time comparison: first attempt

Question

I, R_1 and R_2 are all comparable. What do these functions have in common?

Answer

They are all linear functions. A function $f : \mathbb{N} \to \mathbb{N}$ is linear if

$$\exists a > 0 : \exists b \geq 0 : f(n) = an + b.$$

Question

What is the "simplest" linear function?

Answer

The function $id : \mathbb{N} \to \mathbb{N}$ defined by $id(n) = n$.

Question

Let $id(n) = n$ and $R_1(n) = 6n + 6$. Is R_1 at least as good as id?

Running time comparison: first attempt

Question

Let $id(n) = n$ and $R_1(n) = 6n + 6$. Is R_1 at least as good as id?

Answer

No.

Running time comparison: first attempt

Question

Let $id(n) = n$ and $R_1(n) = 6n + 6$. Is R_1 at least as good as id?

Answer

No.

Proof

Towards a contradiction, assume that R_1 is at least as good as id. Then

$$\exists F > 0 : \forall n \geq 0 : R_1(n) \leq F \times id(n)$$

Then $6 = R_1(0) \leq F \times id(0) = F \times 0 = 0$. Since it is not the case that $6 \leq 0$, we have reached a contradiction.

Running time comparison: second attempt

Definition

Let $f : \mathbb{N} \rightarrow \mathbb{N}$ and $g : \mathbb{N} \rightarrow \mathbb{N}$ estimate the number of elementary instructions executed by two algorithms. We say that f is at least as good as g, denoted $f \in O(g)$, if

$$\begin{array}{l} \text{there exists a } M \geq 0, \text{there exists a } F > 0, \\ \text{such that for all } n \geq M, f(n) \leq F \times g(n) \end{array}$$

Running time comparison: second attempt

Definition

Let $f : \mathbb{N} \to \mathbb{N}$ and $g : \mathbb{N} \to \mathbb{N}$ estimate the number of elementary instructions executed by two algorithms. We say that f is at least as good as g, denoted $f \in O(g)$, if

$$\begin{array}{l}\text{there exists a } M \geq 0, \text{there exists a } F > 0,\\ \text{such that for all } n \geq M, f(n) \leq F \times g(n)\end{array}$$

Notation

Instead of

$$\begin{array}{l}\text{there exists a } M \geq 0, \text{there exists a } F > 0,\\ \text{such that for all } n \geq M, f(n) \leq F \times g(n)\end{array}$$

we write

$$\exists M \geq 0 : \exists F > 0 : \forall n \geq M : f(n) \leq F \times g(n)$$

Running time comparison: second attempt

Claim

Recall that $I = 12n + 2$ and $R_1(n) = 6n + 6$. $I \in O(R_1)$.

Running time comparison: second attempt

Claim

Recall that $I = 12n + 2$ and $R_1(n) = 6n + 6$. $I \in O(R_1)$.

Proof

We have to show that

$$\exists M \geq 0 : \exists F > 0 : \forall n \geq M : I(n) \leq F \times R_1(n)$$

We pick $M = 0$ and $F = 2$. Let $n \geq M$. Then

$$\begin{array}{rcl} I(n) & = & 12n + 2 \\ & \leq & 12n + 12 \\ & = & 2 \times (6n + 6) \\ & = & F \times R_1(n). \end{array}$$

Running time comparison: second attempt

Claim

Recall that $R_1(n) = 6n + 6$ and $id(n) = n$. $R_1 \in O(id)$.

Running time comparison: second attempt

Claim

Recall that $R_1(n) = 6n + 6$ and $id(n) = n$. $R_1 \in O(id)$.

Proof

We have to show that

$$\exists M \geq 0 : \exists F > 0 : \forall n \geq M : R_1(n) \leq F \times id(n)$$

We pick $M = 6$ and $F = 7$. Let $n \geq M$. Then

$$\begin{aligned} R_1(n) &= 6n + 6 \\ &\leq 6n + n \\ &= 7n \\ &= 7 \times id(n). \end{aligned}$$

Big-O notation

Let $f : \mathbb{N} \to \mathbb{N}$ be defined by $f(n) = n$. Instead of $O(f)$ we write $O(n)$.

Let $g : \mathbb{N} \to \mathbb{N}$ be defined by $g(n) = \log_2(n)$. Instead of $O(g)$ we write $O(\log_2(n))$.

Let $h : \mathbb{N} \to \mathbb{N}$ be defined by $h(n) = n^2$. Instead of $O(h)$ we write $O(n^2)$.

Running time comparison: second attempt

Claim

Recall $6\log_2(n+2) \leq R_3(n) \leq 11\log_2(n+2) + 50$.
$R_3 \in O(\log_2(n))$.

Running time comparison: second attempt

Claim

Recall $6\log_2(n+2) \leq R_3(n) \leq 11\log_2(n+2) + 50$.
$R_3 \in O(\log_2(n))$.

Proof

We have to show that

$$\exists M \geq 0 : \exists F > 0 : \forall n \geq M : R_3(n) \leq F \times \log_2(n)$$

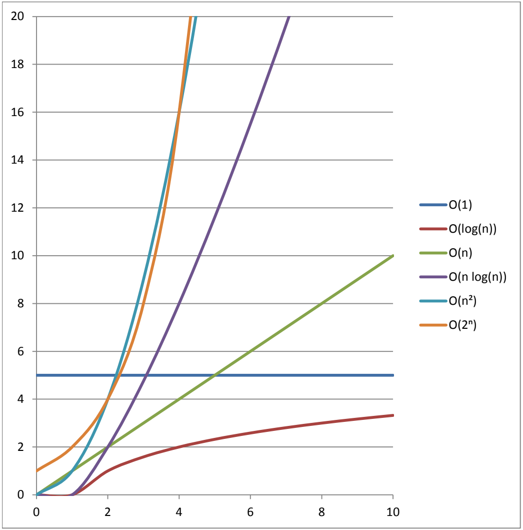
Pick $M = 2$ and $F = 100$. Let $n \geq M$. Then

$$\begin{array}{rcl} R_3(n) & \leq & 11\log_2(n+2) + 50 \\ & \leq & 100 \times \log_2(n) \\ & = & F \times \log_2(n). \end{array}$$

Big-O notation: terminology

$O(1)$	constant
$O(\log(n))$	logarithmic
$O(n)$	linear
$O(n \log(n))$	linearithmic
$O(n^2)$	quadratic
$O(2^n)$	exponential

Big-O notation



Powers of 3

- $I,\ R_1,\ R_2 \in O(n)$
- $R_3 \in O(\log(n))$

Powers of 3

Claim

Recall $6\log_2(n+2) \leq R_3(n) \leq 11\log_2(n+2) + 50$. $R_3 \in O(n)$.

Claim

Recall $6\log_2(n+2) \leq R_3(n) \leq 11\log_2(n+2) + 50$. $R_3 \in O(n)$.

Proof

We have to show that

$$\exists M \geq 0 : \exists F > 0 : \forall n \geq M : R_3(n) \leq F \times n$$

Pick $M = 2$ and $F = 100$. Let $n \geq M$. Then

$$\begin{aligned} R_3(n) &\leq 11\log_2(n+2) + 50 \\ &\leq 100 \times n \\ &= F \times n. \end{aligned}$$

Powers of 3

Claim

Recall $6\log_2(n+2) \leq R_3(n) \leq 11\log_2(n+2) + 50$. $R_3 \in O(n^2)$.

Claim

Recall $6\log_2(n+2) \leq R_3(n) \leq 11\log_2(n+2)+50$. $R_3 \in O(n^2)$.

Proof

We have to show that

$$\exists M \geq 0 : \exists F > 0 : \forall n \geq M : R_3(n) \leq F \times n^2$$

Pick $M = 2$ and $F = 100$. Let $n \geq M$. Then

$$\begin{array}{rcl} R_3(n) & \leq & 11\log_2(n+2)+50 \\ & \leq & 100 \times n^2 \\ & = & F \times n. \end{array}$$

Big-Theta notation

Definition

Let $f : \mathbb{N} \to \mathbb{N}$ and $g : \mathbb{N} \to \mathbb{N}$ estimate the number of elementary instructions executed by two algorithms. We say that f is as good as g, denoted $f \in \Theta(g)$, if

$$f \in O(g) \text{ and } g \in O(f).$$

Running time comparison

Claim

Recall $6 \log_2(n+2) \leq R_3(n) \leq 11 \log_2(n+2) + 50$.
$\log_2(n) \in O(R_3)$.

Running time comparison

Claim

Recall $6\log_2(n+2) \leq R_3(n) \leq 11\log_2(n+2)+50$.
$\log_2(n) \in O(R_3)$.

Proof

We have to show that

$$\exists M \geq 0 : \exists F > 0 : \forall n \geq M : \log_2(n) \leq F \times R_3(n)$$

Pick $M = 0$ and $F = 1$. Let $n \geq M$. Then

$$\begin{aligned}\log_2(n) &\leq 11\log_2(n+2)+50 \\ &= F \times R_3(n).\end{aligned}$$

Big-Theta notation

Claim

$R_3 \in \Theta(\log_2(n))$.

Big-Theta notation

Claim

$R_3 \in \Theta(\log_2(n))$.

Proof

Since we have already shown that $R_3 \in O(\log_2(n))$ and $\log_2(n) \in O(R_3)$, we can conclude that $R_3 \in \Theta(\log_2(n))$.

Big-Theta notation

Claim

$R_3 \in \Theta(\log_2(n))$.

Proof

Since we have already shown that $R_3 \in O(\log_2(n))$ and $\log_2(n) \in O(R_3)$, we can conclude that $R_3 \in \Theta(\log_2(n))$.

Claim

$I,\ R_1,\ R_2 \in \Theta(n)$.

Powers of 3

Question

If you have to calculate some power(s) of 3, which algorithm would you use?

Question

If you have to calculate some power(s) of 3, which algorithm would you use?

Answer

This depends on the value(s) of n for which you have to compute pow3(n).

Powers of 3

Question

If you have to calculate pow3(4), which algorithm would you use?

Powers of 3

Question

If you have to calculate pow3(4), which algorithm would you use?

Answer

None. Just use 81.

Powers of 3

Question

If you have to calculate many pow3(n) in the range $0 \leq n \leq 100$, which algorithm would you use?

Question

If you have to calculate many pow3(n) in the range $0 \leq n \leq 100$, which algorithm would you use?

Answer

Use a simple algorithm such as I or R_1 to compute pow3(n) for all $0 \leq n \leq 100$ and store them in a map.

Powers of 3

```
private final static Map<Integer, BigInteger> pow3 =
   new HashMap<Integer, BigInteger>();

static
{
   BigInteger power = BigInteger.ONE;
   for (int i = 0; i <= 100; i++)
   {
      pow3.put(i, power);
      power = power.multiply(THREE);
   }
}

public static BigInteger pow3(int n)
{
   return pow3.get(n);
}
```

Powers of 3

Question

If you have to calculate pow3(n) for large values of n, which algorithm would you use?

Powers of 3

Question

If you have to calculate pow3(n) for large values of n, which algorithm would you use?

Answer

Use R_3.