Asymptotic Analysis

Big-O Notation: Definition

$f(x) = O(2^x)$
$f(x) = F(x-1) + F(x-2)$

running times of algs

f(x) = O(g(x)) means that

x = size of input

- There exists:
  - some positive constant M
  - some minimum x-value x_0
- Such that for all $x > x_0$:
  - $f(x) \le M * g(x)$

running time = f(x)
f(x) is the worst case # primitive ops for input of size x

$M = 1000$
$x_0 = 1000$

as $x \to \infty$

$f(x) = 7 + 4\cancel{x} + 4\cancel{x^2}$

$f(x) = O(x^2)$

if $x > 1000$, then $7 + 4x + 4x^2 \le 1000x^2$

Other Types of Asymptotic Relationships

$x^2 = o(x!)$

$f(x) = o(x!)$

- Little-o notation: $f(x) = o(g(x))$ iff:
  - For every positive ε,
  - There exists a constant x_ε
  - Such that $f(x) \le \varepsilon g(x)$ for all $x \ge x_\varepsilon$

$f(x) = x$

$g(x) = x^2$

$f(x) = o(g(x))$

Informally, this means that g(x) grows much faster than f(x): for EVERY ε, no matter how small, we can find a place where g(x) is 1/ε-times bigger than f(x). In other words, even if we shrink g(x) by a factor of 100, 1000, 1,000,000, …, it is still going to be bigger than f(x).

Other Types of Asymptotic Relationships

- Big-Omega notation:

$$f(x) = \Omega(g(x)) \text{ iff } g(x) = O(f(x))$$

- Big-Theta notation:

$$f(x) = \Theta(g(x)) \text{ iff } f(x) = O(g(x)) \text{ and } g(x) = O(f(x))$$

$g(x) = \Theta(f(x))$

Big-O Notation: In Practice

- Use these simplification rules:
  - Only pay attention to the dominant terms (x^3 more important than x^2)
  - Don't include constants in your big-O expression

x^c dominate x^d if $c > d$

$$5x^2 + 3x = O(2x^2)$$
$$= O(x^2)$$

Big-O Notation and Limits

Theorem: Let f(x) and g(x) be real-valued functions.

$x \geq 0$

$f(x), g(x) > 0$

Let $L = \lim_{x \to \infty} \frac{f(x)}{g(x)}$. Then:

1. If $L = 0$, $f(x) = o(g(x))$ and $f(x) = O(g(x))$
2. If $L = \infty$, $g(x) = o(f(x))$ and $f(x) = \Omega(g(x))$ $\Leftrightarrow$ $g(x) = O(f(x))$
3. If $0 < L < \infty$, $f(x) = \Theta(g(x))$ and $f(x) = O(g(x))$
4. ~~If $L < \infty$, $f(x) = O(g(x))$~~

$g(x) = O(f(x))$

$g(x) = \Theta(f(x))$

L'Hopital's Rule

What if f(x) and g(x) both = 0 or ∞ in the limit?

<u>L'Hopital's Rule:</u>

If $\lim_{x \to \infty} f(x) = \lim_{x \to \infty} g(x) = 0$ (or ∞), then $\lim_{x \to \infty} \frac{f(x)}{g(x)} = \lim_{x \to \infty} \frac{f'(x)}{g'(x)}$

(as long as the latter limit exists).

Examples

Example: What is the asymptotic relationship between $f(x) = x^2$ and $g(x) = 2^x$?

$$\lim \frac{x^2}{2^x} = \lim \frac{2x}{2^x \ln 2} = \lim \frac{2}{2^x (\ln 2)^2} = 0$$

$f(x) = o(g(x))$

$f(x) = O(g(x))$

Examples

Example: What is the asymptotic relationship between $g(x) = 10x^2 + 5$ and $h(x) = 15 \log x + 2x^2 + 5x + 2$?

Examples

Find a function g(x) such that $f(x) = 5x^4 + 3x^2 + 10000$ is $O(g(x))$, with proof.

Examples

$f(x) = f(x-1) + f(x-2)$

$f(1) = 1$, $F(2) = 1$

Claim: Let f(x) be the x^{th} Fibonacci number. Prove that f(x) is $O(2^x)$.

x	f(x)	2^x
1	1	2
2	1	4
3	2	8
4	3	16
5	5	32
6	8	64
7	13	128

$M = 1$

$x_0 = 0$

Proof: $f(x) \le M(2^x)$ if $x > x_0$

$\rightarrow f(x) \le 2^x$ for all x

Base step: $x = 1$

I.H.: Assume $f(x) \le 2^x$ for $x = k$

I.S.: Show that $f(x) \le 2^x$ for $x = k+1$

Examples

Claim: $f(x) = O(2^x)$

Proof: We will work from the def'n of big-O.
Set $M=1$, $x_0=0$. Then we show that $f(x) \leq 1 \cdot 2^x$
for all $x > 0$, by induction.

Base case: $x=1$. $f(1)=1$, $2^1=2$, so it clearly holds.

I.H.: Suppose for some $k \geq 1$, $f(k) \leq 2^k$ (and it also for all smaller k)

I.S.: We have to show that $f(k+1) \leq 2^{k+1}$.

$$f(k+1) = f(k)+f(k-1) \leq 2^k + 2^{k-1} \leq 2^k + 2^k = 2^{k+1}$$

Thus, $f(k+1) \leq 2^{k+1}$, so the claim holds.

Some Running Time Functions that Computer Scientists Like

- Polynomial time: $O(n^4)$, $O(n^2)$
- Logarithmic: $O(\log n)$, $O(\log \log n)$
- Quasi-linear: $O(n \log n)$ fairly common
- Sublinear: $O(n^{1/2})$ this is the best!
- Exponential: $O(2^x)$ very bad! Often indicates something that is basically brute force

Recurrence Relations

Recurrence Relations

- A recurrence relation is an equation that recursively defines a function's values in terms of earlier values

Fibonacci

- Very useful for analyzing an algorithm's running time!

Recurrence Relations in Math

- For instance: Fibonacci numbers

$F(1) = 1$

$F(2) = 2$

$F(n) = F(n - 1) + F(n - 2)$

Recurrence Relations in Code

```
def naivePower(x, n):
    if n == 0:
        return 1
    else:
        return x * naivePower(x, n - 1)
```

x^n n int ≥ 0

x^2 return $x \cdot nP(x,1)$

$x = x \cdot nP(x,0)$

1

$x \cdot x^{n-1} = x^n$

How can we write the running time?

$$\begin{cases} T(n) = c_2 + T(n-1) \\ T(0) = c_1 \end{cases}$$

look-up of x, branching, mult., return, subtract.

Recurrence Relations in Code

$T(0) = c_1$

$T(n) = c_2 + T(n-1)$

If only we had an expression for $T(n-1)$...

$$T(n) = c_2 + T(n-1)$$
$$= c_2 + (c_2 + T(n-2))$$
$$= c_2 + c_2 + (c_2 + T(n-3))$$
$$\vdots$$
$$= c_2 + \cdots + c_2 + T(0) = nc_2 + c_1$$

$O(n)$

Recurrence Relations in Code

$T(0) = c_1$

$T(n) = c_2 + T(n - 1)$

Expanded:

Recurrence Relations in Code

$T(n) =$ the number of p.o.s required for input n

```
def betterPower(x, n):
    if n == 0:
        return 1
    else if n == 1:
        return x
    else ~~if n % 2 == 0~~:
        return ~~Power~~(x * x, n/2)
```

bP

$c_3 > c_2 > c_1$

$(x^2)^{n/2} = x^n$

(assume for simplicity that n is a power of 2)

How can we write the running time?

mult, div, return, etc.

$T(0) = c_1 \quad T(1) = c_2 \quad T(n) = T(n/2) + c_3$

Recurrence Relations in Code

$T(0) = c_1$

$T(1) = c_2$

$T(n) = c_3 + T(n/2)$

(for simplification we'll assume n is a power of 2)

If only we had an expression for $T(n/2)$…

Recurrence Relations in Code

$T(0) = c_1,\ T(1) = c_2,\ \ldots,\ T(n) = c_3 + T(n/2)$

Expanded:

$\log_2 n = m$

$2^m = n$

$$T(n) = c_3 + T(n/2) \qquad 16$$

$$= c_3 + (c_3 + T(n/4)) \qquad 8$$

$$= c_3 + c_3 + c_3 + T(n/8) \qquad 4$$

$$\vdots \qquad \vdots \quad 2$$

$$= c_3 + \cdots + c_3 + T(1) = c_3 \cdot \log_2 n + c_2 = O(\log n)$$

Recurrence Relations in Code

$T(0) = c_{1,}\ \ T(1) = c_2,\ \ \ldots,\ T(n) = c_3 + T(n/2)$

$$
\begin{aligned}
T(n) &= c_3 + T(n/2) \\
&= c_3 + c_3 + T(n/4) \\
&= c_3 + c_3 + c_3 + T(n/8)
\end{aligned}
$$

.....

$$= kc_3 + T(n/(2^k))$$

What should k be in order for us to get down to T(1)?

Recurrence Relations in Code

$T(0) = c_{1,}\ \ T(1) = c_2,\ \ \ldots, T(n) = c_3 + T(n/2)$

$$
\begin{aligned}
T(n) &= c_3 + T(n/2) \\
&= c_3 + c_3 + T(n/4) \\
&= c_3 + c_3 + c_3 + T(n/8)
\end{aligned}
$$

.....

$$= kc_3 + T(n/(2^k))$$

What should k be in order for us to get down to T(1)?

$2^k = n$, so $k = \log n$

Recurrence Relations in Code

$T(0) = c_{1,}\ \ T(1) = c_2,\ \ldots, T(n) = c_3 + T(n/2)$

$$
\begin{aligned}
T(n) &= c_3 + T(n/2) \\
&= c_3 + c_3 + T(n/4) \\
&= c_3 + c_3 + c_3 + T(n/8)
\end{aligned}
$$

.....

$$
\begin{aligned}
&= kc_3 + T(n/(2^k)) \\
&= c_3 * \log n
\end{aligned}
$$

What should k be in order for us to get down to T(1)?

$2^k = n$, so $k = \log n$

Writing Functions as Recurrences

- Many "normal" mathematical functions can be written as recurrence relations:
- $f(x) = x$

$f(1) = 1$

$f(x) = 1 + f(x-1)$

- $f(x) = 2^x$

$f(1) = 2$

$f(x) = 2 \cdot f(x-1)$

- $f(x) = x!$

$f(1) = 1$

$f(x) = x \cdot f(x-1)$

Solving Recurrences

- Solving recurrence relations is like integrating an expression- there are tricks, but no techniques are guaranteed to work

Solving Recurrences

- Simplest method: Guess the solution, prove with induction
- Sometimes it helps to do a few expansions to get some intuition

Example

Claim: ~~The recurrence relation given by:~~ Define T:

$T(0) = c_1$

$T(n) = c_2 + T(n-1)$. Define $T'(n) = nc_2 + c_1$.

~~has the solution $T(n) = nc_2 + c_1$~~ Claim: $T(n) = T'(n)$.

Proof: By induction.

Base case: $n = 0$. $T(0) = c_1$. $T'(0) = c_1$. ✓

I.H.: Assume that up to some k, $T(k) = T'(k)$.

I.S.: We need to show that $T(k+1) = T'(k+1)$.

$T(k+1) = c_2 + T(k) = c_2 + T'(k) = c_2 + kc_2 + c_1 =$

✓ $T'(k+1) = (k+1)c_2 + C_1$.

Example

In-Class Exercise

Solve (with ~~proof!~~) the following recurrences:

1. $T(0) = 1$, $T(1) = 0$, $T(n) = 2 * T(n - 2)$
2. $T(0) = 0$, $T(n) = T(n - 1) + n$
3. $T(0) = 0$, $T(n) = T(n - 1) * n$
4. $T(1) = 1$, $T(n) = 2 * T(n/2)$ (assume n is a power of 2)
5. $T(0) = -1$, $T(n) = T(n - 1)^2$

Solutions

$T(0)=1$, $T(1)=0$, $T(n)=2\cdot T(n-2)$

n	$T(n)$
0	1
1	0
2	$2=2^1$
3	0
4	$4=2^2$
5	0
6	$8=2^3$
7	0
8	$16=2^4$

if n is odd, $T(n)=0$

if n is even, $T(n)=2^{(n/2)}$

B.C.: $T(0)=2^{0/2}$ ✓

I.H.: Assume for some even k,
$T(k)=2^{k/2}$

I.S.: Show it works for $k+2$.

$T(k+2)=2\cdot T(k)=2\cdot 2^{k/2}=2^{k/2+1}$
$=2^{(k+2)/2}$. ✓