Asymptotics

Tuesday, September 29, 2015 11:32 AM

Meeting
Software eng club
6-7 ACM room

kargers Alg

First, Stacks

RPN calculators

1 + 2
infix

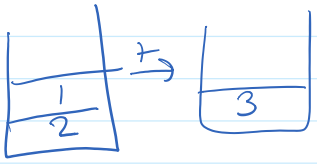
1 + 2
Polish

2 1 +
Reverse Polish (RPN)

Given a RPN formula:

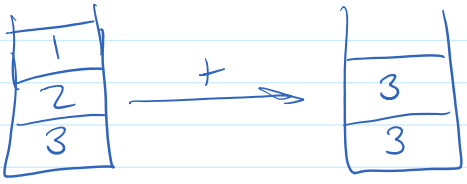
```
For each token in formula:
    If token is an operand:
        push it onto a stack
    If token is an operator with arity n:
        // arity is the number of operands
        pop n items off the stack, and apply operator
        push result back on stack
pop the answer off the stack // assumes "well-formed"
```

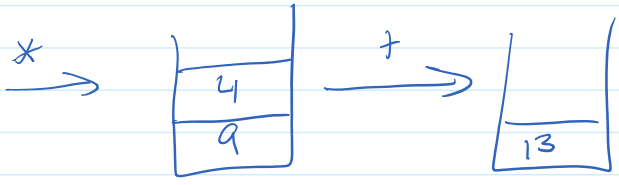
2 1 +



+, *

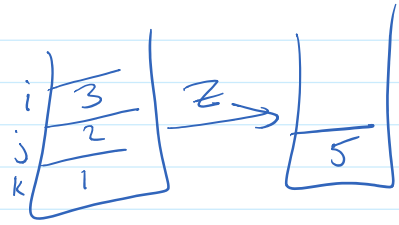
3 2 1 + x 4 +



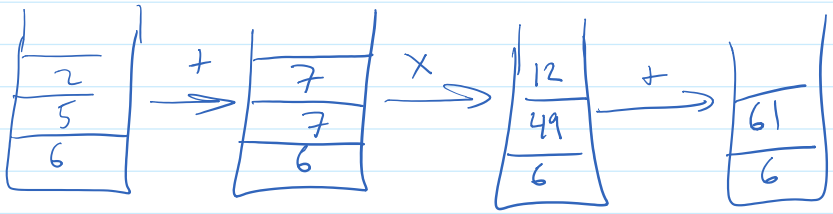


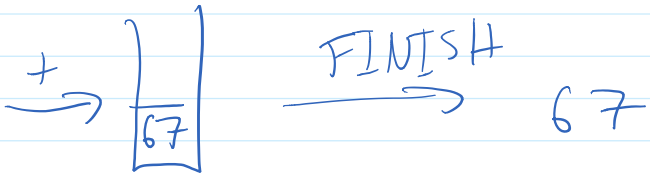
1 2 3 z

```
define z(i, j, k):
    (i+j)*k
```

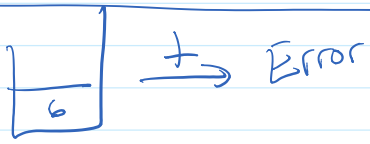


Fmla = 6 5 2 + 7 × 12 + +

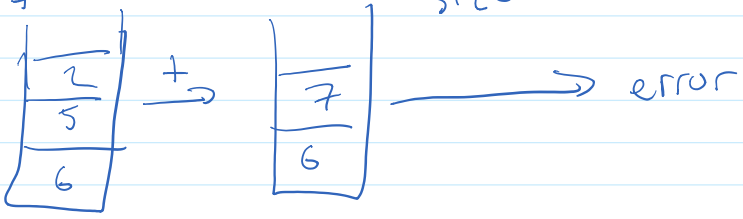




6 +



6 5 2 +



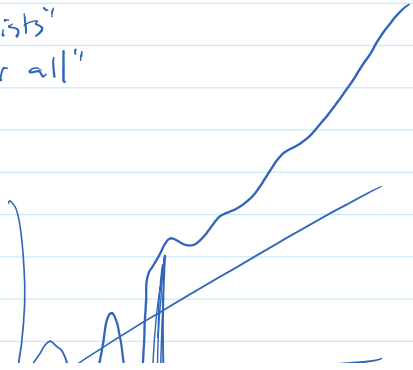
Asymptotics

- monotonic increasing functions "always goes up"
- positive semidefinite functions "never below 0"

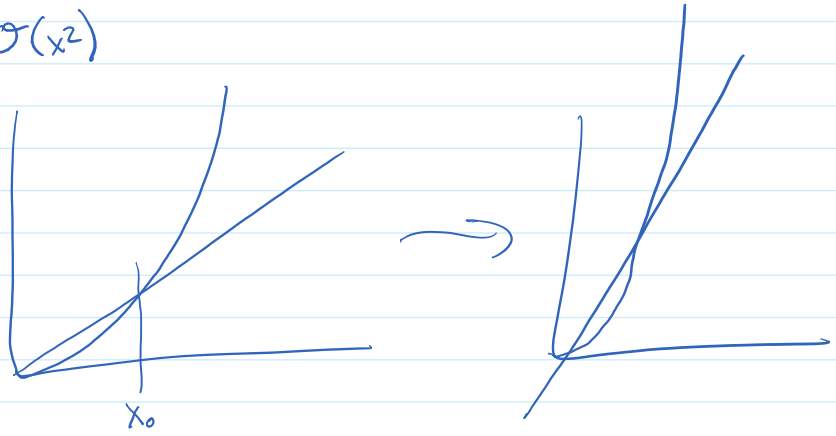
$f(x) = O(g(x))$

$\exists\ x_0, c$ s.t. $\forall\ x > x_0$ $f(x) \leq cg(x)$

$\exists$ "exists"
$\forall$ "for all"



$x = O(x^2)$

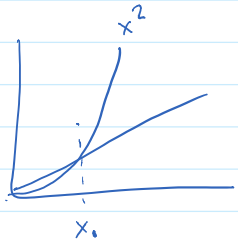


Big Omega

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

$\exists\ x_0, c$ s.t. $\forall\ x > x_0$ $f(x) \geq c\,g(x)$

$x^2 = \Omega(x)$



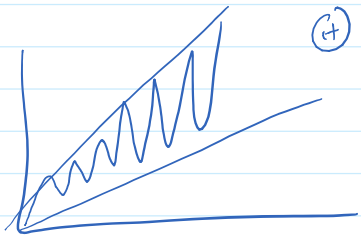
Big Theta

$f(x) = \Theta(g(x))$ if $f(x) = O(g(x))$ and $f(x) = \Omega(g(x))$

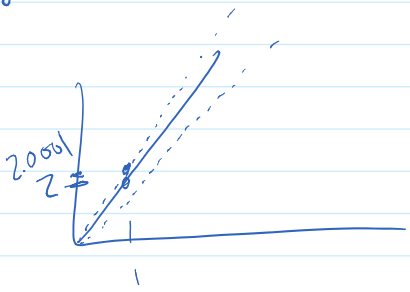
Example: $2x = \Theta(x)$

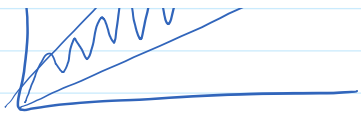
$2x = O(x)$ by choosing $c > 2$

$2x = \Omega(x)$ by choosing $c < 2$



$\Theta(x)$





$2x = O(x)$

$\exists x_0, c$ s.t. $2x < cx \quad \forall x > x_0$

choose $x_0 = 0, c = 3$

$2x < 3x$

$2 = O(1)$

Recall an ArrayList's size may be (at most) $2n$
n = # of elements in the list.

size of an ArrayList = $O(n)$
$\Omega(n)$ bound is clear
size = $\Theta(n)$

Example

```
for (int i=0; i<n; i++) {
    j=i
    Constant time operations
    k=i+1
}
```

$O(n)$

```
for (int i=0; i<n; i++) {
    //O(n)
}
```

$O(n^2)$

```
for (int i=0; i<n; i++) {
    for (int j=0; j<n; j++) {
        //constant
    }
}
```

$O(n^2)$

```
for (int i=0; i<n; i++) {
```

```
for (int i=0; i<n; i++) {
    O(n)    : //constant
}
for (int i=0; i<n; i++) {
    j=i
    k=i+j
O(n)  :
}
```

$O(n)$

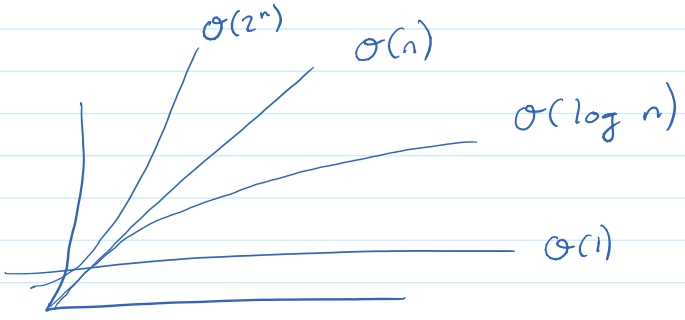
linear (n)

quadratic (n^2)

$1 < n < n^2 < n^3$

logarithmic $(\log n)$

exponential (2^n)



$1 < \log n < n < 2^n$

$\forall k, \ell \quad n^k < \ell^n \quad (\ell > 1)$

polynomial ↑ ↑ exponential

$(\log n < n^{1+\epsilon})$

$n \log n < n^{2+\epsilon} \qquad 0 < \epsilon$

$n(\log n < n^{1+\epsilon})$

$n \log n < n(n^{1+\epsilon}) = n^{1+\epsilon+1} = n^{2+\epsilon}$

$\log_2 8 = \frac{\log 8}{\log 2} \qquad \log_2 n = \frac{\log n}{\log 2}$ ←

$\log_2 n = \Theta(\log n)$

choose constants a, b

$$\log_a n = \frac{\log_b n}{\log_b a}$$ constant

$$\log_a n = \Theta(\log_b n) = \Omega(\log_b n)$$
$$= \Theta(\log_b n)$$

$\lg n := \log_2 n$

$$\log_a n = \frac{\log_b n}{\log_b a}$$

$\log_a n = c \log_b n$

$\log_a n \le c \log_b n \Rightarrow \log_a n = O(\log_b n)$

$c = \frac{1}{\log_b a}$

$\log_a n \ge c \log_b n \Rightarrow \log_a n = \Omega(\log_b n)$

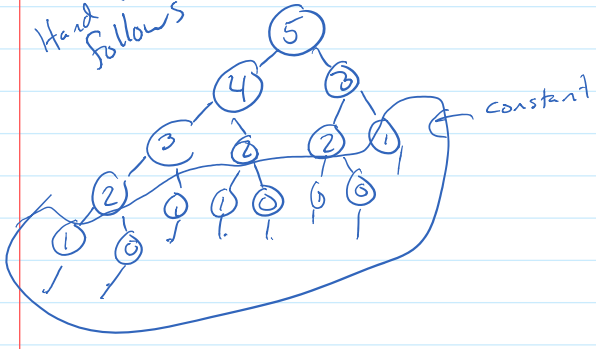
$\log_a n = \Theta(\log_b n)$

```
def fib(x):
    if (x < 2):
        return x
    return fib(x-1) + fib(x-2)
```
$\Theta(fib(x))$

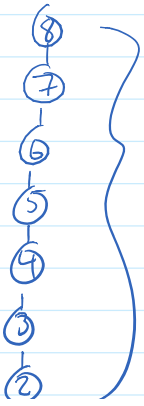
0, 1, 1, 2, 3, 5, 8,

n	3	4	5
	3,	5,	8, ...

Hand waving follows




```
def fact(x)
    return x * fact(x-1)
```



$\Theta(n)$

```
def fact(x):
    prod = 1
    for (i=x; i>0; i--) {
```

$O(n)$

```
prod = 1
for (i=x; i>0; i--) {
    prod *= i;
}
```

$O(n \lg n)$

```
for (int i=n; i>0; i--)
    // O(lg n)
}
```

$x^3 = O(x^2)$ False

1, 2, x, x^2, 5x, 3, x^3

1 x x^2 x^3
2 5x
3