cond

(cond (predicate1 exp1)
  (predicate2 exp2)
  ...
  (else expn))

else is not mandatory
no begin is needed (because parenthesis are here
If multiple predicates are true, only the first expression
is evaluated
How would you survive without else?
How would you survive without cond?

Formal definition

\[ R(n) = \Theta(f(n)) \]
iff there exists \( N, k_1, k_2 \)
so that
For each \( n > N \),
\[ k_1 f(n) \leq R(n) \leq k_2 f(n) \]

Order of growth

<table>
<thead>
<tr>
<th>Name</th>
<th>Notation</th>
<th>( n=2 )</th>
<th>( n=10 )</th>
<th>( n=100 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constant</td>
<td>( \Theta(1) )</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Logarithmic</td>
<td>( \Theta(\log n) )</td>
<td>1</td>
<td>3.33</td>
<td>6.66</td>
</tr>
<tr>
<td>Linear</td>
<td>( \Theta(n) )</td>
<td>2</td>
<td>10</td>
<td>100</td>
</tr>
<tr>
<td>Quadratic</td>
<td>( \Theta(n^2) )</td>
<td>4</td>
<td>100</td>
<td>10,000</td>
</tr>
<tr>
<td>Exponential</td>
<td>( \Theta(2^n) )</td>
<td>4</td>
<td>1024</td>
<td>1.26 \times 10^{30}</td>
</tr>
</tbody>
</table>

Order of growth space/time

Order of growth space/time

(except (fib1 n)
  (cond ((= n 0) 0)
    ((= n 1) 1)
    (else (+ (fib1 (- n 1))
              (fib1 (- n 2))))))

Order of growth space/time

(except (fib2 n)
  (define (fib-iter a b count)
    (if (= count 0) b
      (fib-iter (+ a b) a (- count 1))))
  (fib-iter 1 0 n))
Print all factorials

Write a procedure that prints all factorials from 1 to n
Be dumb, then be smart.
Analyze the orders of growth

Print all factorials

Write a procedure that prints all factorials from 1 to n
Be dumb, then be smart.
Analyze the orders of growth
Now use display-bar. What does it change?

(define display-bar (lambda (n)  
    (if (= n 0) (display \n")  
        (begin (display ".")  
            (display-bar (- n 1))))))

Multiplication

Write a procedure (fastmul m n) that is faster than linear

Binary numbers

(define binary (lambda (n)  
    (if (> n 0)  
        (begin  
            (binary (floor (/ n 2)))  
            (display (if (even? n) "0" "1"))))))

Patterns

Recursive call:  
- n calls n-1 → linear  
- n calls n/2 → log  
- n calls n-1 & n-2 → exponential

2 issues  
- How many steps  
- Cost of each step
Exponential processes: often, the recursive procedure calls itself twice (or more) at each iteration
Quadratic processes often correspond to two linear processes where the first one calls the second one

Order?

(define (dummy x)  
    (if (<= 1 x) x  
        (+ (dummy (floor (/ x 2)))  
            (dummy (- x (floor (x-x/2)))))))