Recap: Flatten Running Time

(define (flatten-commands ll)
  (if (null? ll) ll
      (if (is-lsystem-command? (car ll))
          (cons (car ll) (flatten-commands (cdr ll)))
          (flat-append (car ll) (flatten-commands (cdr ll))))))

First: determine running times of all the procedures applied in flatten-commands.

Each recursive call involves constant work.

For input list that is all lsystem commands of length \( N \):
flatten-commands has running time in \( \Theta(N) \) where \( N \) is the number of lcommand in the input list.

Second: determine running time for each application except for recursive call.

Paths to Flattening

Each recursive call reduces the number of elements in \( ll \) by one.

For input list that is all lists of length \( P \):
flatten-commands has running time in \( \Theta(P) \) where \( P \) is the number of elements in \( ll \) and all its sub-lists, not counting elements in offshoot command lists.

Combining the Paths

(define (flatten-commands ll)
  (if (null? ll) ll
      (if (is-lsystem-command? (car ll))
          (cons (car ll) (flatten-commands (cdr ll)))
          (flat-append (car ll) (flatten-commands (cdr ll))))))

For input list that is all lsystem commands:
flatten-commands has running time in \( \Theta(N) \) where \( N \) is the number of elements in the input list.

For input list that is all lists of length \( P \):
flatten-commands has running time in \( \Theta(QP) \) where \( Q \) is the number of sub-lists of length \( P \) in the input list.

For any input:
flatten-commands has running time in \( \Theta(M) \) where \( M \) is the size of the input list (the total number of lcommands in \( ll \) and all its sub-lists, not counting elements in offshoot command lists).
Power

Define and analyze the asymptotic running time of a procedure `power` that takes two numbers, `a` and `n`, and input, and outputs \(a^n\).

Hint:
\[ a^0 = 1 \]
\[ a^n = a \cdot a^{n-1} \text{ for } n > 0 \]

Simple Power

```scheme
(define (power a n)
  (if (= n 0) 1
      (* a (power a (- n 1))))))
```

What is the asymptotic running time of `power`?

Running Time Analysis

```scheme
(define (power a n)
  (if (= n 0) 1
      (* a (power a (- n 1))))))
```

1. What are the running times of procedures applied by `power`?
2. What is the running time of evaluating the body except for the recursive call?
3. How many recursive calls are there?

What about `*`?

```scheme
(* a (power a (- n 1)))
```

- Cannot be constant time
- Multiplication must at least look at all the digits in both numbers
  \( * \) is in \( \Omega(W) \) where \( W \) is the total length (number of bits) of the inputs.
- Grade-school multiplication algorithm has running time in \( \Theta(W^2) \)
  \( * \) is in \( O(W^2) \) where \( W \) is the total length (number of bits) of the inputs.

Note: \( \Theta \) instead of \( O \) since there may be faster `*` algorithms, and we don't know what Scheme interpreter actually does.

What about `*` as it is used here? (what is \( W \)?)

* has running time in \( O(a + b \cdot n^2) \) where \( a \) is number of bits in a and \( n \) is value of n.
Running Time Analysis

(define (power a n)
  (if (= n 0) 1
      (* a (power a (- n 1))))))

Each body evaluation:
- and – with one input constant: constant running time
- * has running time in \( O(a_b n^2) \) and \( \Omega(a_b n^2) \)

Number of recursive calls:
- \( n_v \), the value of the second input \( n \)

Running time for power is in \( O(a_b n_v^2) = O(a_b n_v^2) \) and \( \Omega(a_b n_v^2) \)

Bits and Values

Running time for power is in \( O(a_b n_v^2) \) and \( \Omega(a_b n_v^2) \)

What is the running time in terms of the size of the input?

\[ n_v = 2^{n_b} \]

Running time for power is in \( O(a_b^2(2^{n_b})^3) \) and \( \Omega(a_b^2(2^{n_b})^2) \).

Testing Power Analysis

Running time for power is in \( O(a_b^2(2^{n_b})^3) \) and \( \Omega(a_b^2(2^{n_b})^2) \).

\[ n_b = 14 \]

Running time for power is in \( O(a_b^2(2^{n_b})^3) \) and \( \Omega(a_b^2(2^{n_b})^2) \).

Faster Powering?

\[ a^{2n} = a^n \times a^n \]

Gold Star Challenge Problem: define and analyze (correctly!) an asymptotically faster power procedure.

Charge

- ACs’ Exam Review Session: Tonight at 6:30, Olsson 001
- Extra Office Hours, Thursday 1:30-3pm
- Exam 1 out Friday