Menu

- Finishing Analyzing Flattening
- Power

Reminders:
Review Session, Wednesday 6:30 in Olsson 001
Extra Office Hours, Thursday 1:30-3 in Olsson 236A

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.

null?, **car**, **cons**, **cdr**, and **is-lsystem-command?** are constant time
flat-append has running time in $\theta(N_1)$ where N_1 is the number of elements in the first input.

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

Need to consider both paths:

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

Paths to Flattening

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

Each recursive call involves constant work .
Each recursive call reduces the number of elements in **ll** by one.

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 lcommands in the input list.

Paths to Flattening

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

Each recursive call involves $\theta(P)$ work where P is the number of elements in (car ll).
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 $\theta(QP)$ where Q is the number of sub-lists (of length P) in the input list.

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 $\mathbf{a^n}$.

Hint:

$a^0 = 1$

$a^n = a * a^{n-1}$ for $n > 0$

Simple Power

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

What is the asymptotic running time of power?

Running Time Analysis

```
(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?**

n, the **value** of the second input *n*

Running Time Analysis

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

1. **What are the running times of procedures applied by power?** = and – with one input constant: constant running time
2. What is the running time of evaluating the body except for the recursive call?
3. How many recursive calls are there?

n, the **value** of the second input *n*

What about *?

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

What about *?

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

* is in $\Omega(W)$ where W is the total length (number of bits) of the inputs.

* is in $O(W^2)$ where W is the total length (number of bits) of the inputs.

What can we say about * as it is used here? (what is *W*?)

Worst case: W = bits in a + bits in a^{n-1}

* has running time in $O((a_b n_v)^2)$ where a_b is number of bits in a and n_v 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 $O((a_b n_v)^2)$ and $\Omega(a_b n_v)$

Number of recursive calls:

n_v, the **value** of the second input *n*

Running time for power is in $O(n_v(a_b n_v)^2) = O(a_b^2 n_v^3)$ and $\Omega(a_b n_v^2)$

Bits and Values

Running time for power is in $O(a_b^2 n_v^3)$ and $\Omega(a_b n_v^2)$
where a_b is number of bits in **a** and n_v is value of **n**.

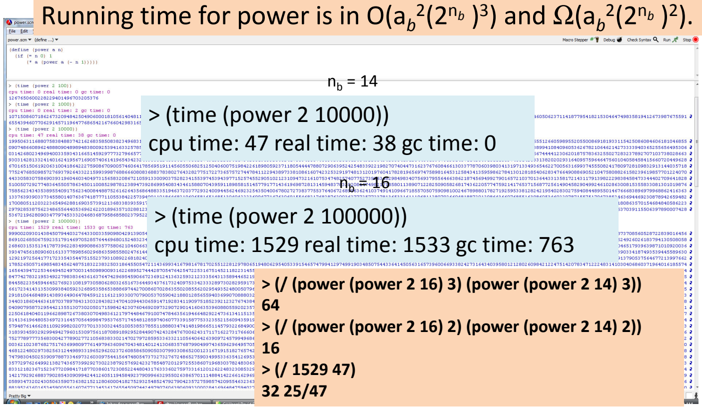
What is the running time in terms of the ***size*** of the input?

$$n_v = 2^{n_b}$$

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 $\Omega(a_b^2 (2^{n_b})^2)$.



$n_b = 14$

```
> (time (power 2 10000))
cpu time: 47 real time: 38 gc time: 0
```

$n_b = 16$

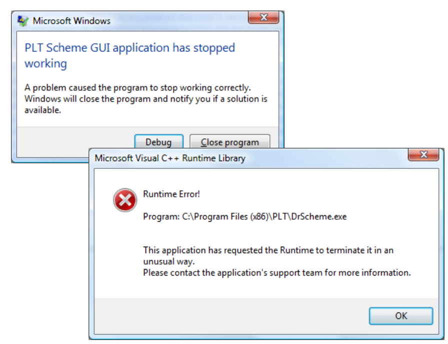
```
> (time (power 2 100000))
cpu time: 1529 real time: 1533 gc time: 763
```

```
> (/ (power (power 2 16) 3) (power (power 2 14) 3))
64
> (/ (power (power 2 16) 2) (power (power 2 14) 2))
16
> (/ 1529 47)
32 25/47
```

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

$n_d = 7$

> (time (power 2 1000000))



Faster Powering?

$$a^{2n} = a^n * a^n$$

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