CS61A Lecture 6

2011-06-28

Colleen Lewis

How long does a function take to run?

- It depends on what computer it is run on!

Assumptions

- We want something independent of the speed of the computer
- We typically use N which is some reference to the "size" of the data.
  - It might be the variable N (in factorial)
  - It might be the size of your sentence
- We don't care about constant factors
  - This is pretty silly in some cases but makes the math more beautiful
- We typically think about the worst case!
- It only matters for BIG input!

Colleen's Morning Routine

- Eat breakfast (10 minutes)
- Brush teeth (5 minutes)

Takes a consistent or **"constant"** amount of time

- Go swimming

Depends **"linearly"** upon the number of laps `L`

- Read Piazza (0 minutes – ???)

Depends upon the number of posts `P` and the words per post `W`

Constant Runtime

O(1)

Constant Runtime

- The runtime doesn't depend upon the "size" of the input

```
(define (square x)
    (* x x))
(define (average a b)
    (/ (+ a b) 2))
(define (max val1 val2)
    (if (> val1 val2) val1 val2))
```

Write a constant runtime procedure? (Brushing your teeth)

Linear Runtime

O(N)

Linear Runtime

- The runtime DOES depend upon the "size" of the input – but just with a linear relationship

```
(define (sum-up-to x)
    (if (< x 0)
        0
        (+ x (sum-up-to (- x 1)))))
(define (count sent)
    (if (empty? sent)
        sent
        (+ 1 (count (bf sent)))))
```

Write a linear runtime procedure? (Swimming laps)

What is the runtime of this?

```
(define (factorial n)
  (if (= n 0)
      1
      (* n (factorial (- n 1)))))
```

A) Constant runtime
B) Linear runtime
C) Linear*Linear (quadratic)
D) Something else
E) No clue!

Problem Solving Tip!

- THINK: Does this depend upon the "size" of the input?
  - Constant vs. Not Constant
- Think about this for ANY function that is called!

Linear * Linear (Quadratic) Runtime

O(N^2)

Linear * Linear Runtime (quadratic) (Reading Piazza)

- The runtime DOES depend upon the "size" of the first input
- For each thing in the first input, we call another linear thing

```
(define (factorial-sent sent)
   (if (empty? sent)
      sent
      (se (factorial (first sent))
        (factorial-sent (bf sent)))))
```

Linear * Linear Runtime (quadratic) (Reading Piazza)

```
(define (factorial-sent sent)
   (if (empty? sent)
      sent
      (se (factorial (first sent))
        (factorial-sent (bf sent)))))
```

Runtime:

O(sent-length * value-of-elements)

```
(define (factorial-sent sent)
   (if (empty? sent)
      sent
      (se (factorial (first sent))
        (factorial-sent (bf sent)))))

(define (square-sent sent)
  (if (empty? sent)
      sent
      (se (square (first sent))
           (square-sent (bf sent)))))
```

Write a linear*linear (quadratic) runtime procedure? (Reading Piazza)

Bad Variable Names

(Slight break from runtimes)

What is the runtime of this?

```
(define (mystery a b)
  (cond
    ((empty? b) (se a b))
    ((< a (first b)) (se a b))
    (else (se (first b)
          (mystery a (bf b))))))
```

A) Constant runtime
B) Linear runtime
C) Linear*Linear (quadratic)
D) Something else
E) No clue

- Try to guess what type of thing the variables `a` & `b` will be
- Try to come up with (small) input that triggers each case

Bad variable names!

- Make your code harder to read!
  - Don't do this!
- Even though we do
  - We will use bad variable names on exams so that you have to read/understand the code

Better variable names!

```
(define (insert num sent)
  (cond
    ((empty? sent) (se num sent))
    ((< num (first sent)) (se num sent))
    (else (se (first sent)
              (insert num (bf sent))))))

STk>(insert 7 '())
()

STk>(insert 1 '(2 5 8))
(1 2 5 8)
```

Insert

```
STk>(insert 8 '(2 5 6 9))
(2 5 6 8 9)
```

```
(insert 8 '(2 5 6 9))
(se 2
    (insert 8 '(5 6 9))
    (se 5
        (insert 8 '(6 9))
        (se 6
            (insert 8 '(9))
            (se 8 '(9))
(se 2 (se 5 (se 6 (se 8 '(9)))))
```

Best vs. Worst Case

```
(define (insert num sent)
  (cond
    ((empty? sent) (se num sent))
    ((< num (first sent)) (se num sent))
    (else (se (first sent)
              (insert num (bf sent))))))

STk>(insert 1 '(2 3 4 5 6 7 8))
(1 2 3 4 5 6 7 8)
```

BEST CASE

```
STk>(insert 9 '(2 3 4 5 6 7 8))
(2 3 4 5 6 7 8 9)
```

WORST CASE

What is the runtime of this?

```
(define (sort sent)
  (if (empty? sent)
      sent
      (insert (first sent)
              (sort (bf sent)))))
```

A) Constant runtime
B) Linear runtime
C) Linear*Linear (quadratic) Correct Answer
D) Something else
E) No clue!

Sort

```
STk>(sort '(8 3 9))
(3 8 9)
```

```
(sort '(8 3 9))
(insert 8
    (sort '(3 9))
    (insert 3
        (sort '(9))
        (insert 9
            (sort '())
            '()
(insert 8 (insert 3 (insert 9 '())))
```

```
(insert 3                              Add to sent size 7
    (insert 6                          Add to sent size 6
        (insert 4                      Add to sent size 5
            (insert 7                  Add to sent size 4
Add to sent size 3  (insert 1
Add to sent size 2      (insert 8
Add to sent size 1          (insert 3
Add to sent size 0              (insert 9
                                    '()
                                ))))))))
(sort sent) if sent length is N
1 + 2 + 3 + 4 + … + (N-1) + N
```

Some Algebra to calculate: 1+2+…+n

$$\sum_{i=0}^{n} i = 1 + 2 + 3 + \cdots + (n-2) + (n-1) + n$$

// Written backwards

$$\sum_{i=0}^{n} i = n + (n-1) + (n-2) + \ldots + 3 + 2 + 1$$

[redacted] [redacted] [redacted] [redacted] [redacted] [redacted]

Some Algebra to calculate: 1+2+…+n

$$\sum_{i=0}^{n} i = 1 + 2 + 3 + \cdots + (n-2) + (n-1) + n$$

$$+ \quad \sum_{i=0}^{n} i = n + (n-1) + (n-2) + \ldots + 3 + 2 + 1$$

$$2\sum_{i=0}^{n} i = n + (n-1) + (n-2) + \ldots + 3 + 2 + 1 \; +$$

Some Algebra to calculate: 1+2+…+n

$$2\sum_{i=0}^{n} i = n + (n-1) + (n-2) + \ldots + 3 + 2 + 1 \; +$$

$$2\sum_{i=0}^{n} i = (n+1) + (n+1) + (n+1) + \ldots + (n+1) + (n+1) + (n+1)$$

$$2\sum_{i=0}^{n} i = n(n+1)$$

OR [redacted]

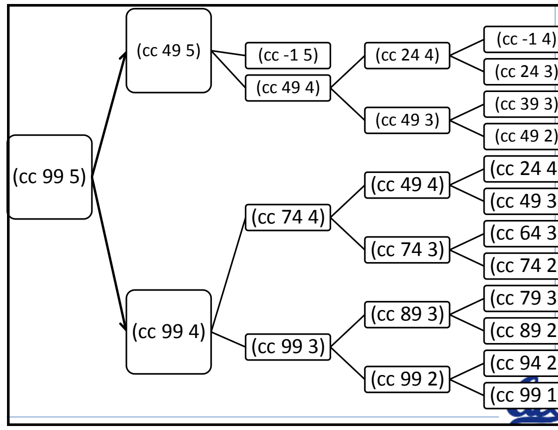
How long does it take you to read Piazza posts and check your email and shower?

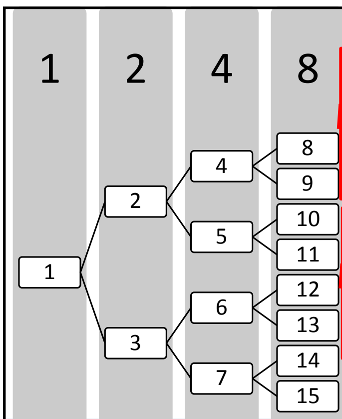
- `P` is the number of posts
- W_p is the number of words per post `P`
- `E` is the number of emails
- W_e is the number of words per email `E`
- `S` is the number of minutes you shower

a) $P*W_p*E*W_e*S$
b) $P+W_p+E+W_e+S$
c) $P*W_p+E*W_e+S$
d) P+E+S
e) P*E*S

Exponential Runtime

2^n





1 branch = 2 calls
2 branches = 3
3 branches = 7
4 branches = 15

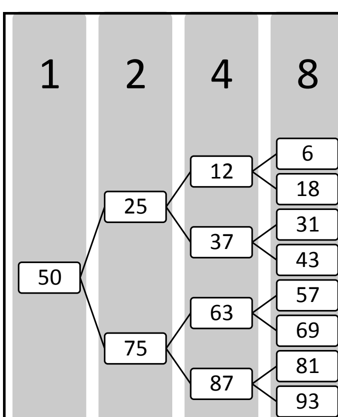
N branches = N^2 - 1 calls

Logarithmic Runtime

$\mathrm{Log}_2(N)$

Number Guessing Game

- I'm thinking of a number between 1 and 100
- How many possible guesses could it take you? (WORST CASE)
- Between 1 and 10000000?
- How many possible guesses could it take you? (WORST CASE)



[redacted]

$$n = 2^{h+1} - 1$$

// Take the log of both sides

$$log_2(n) = \boxed{log_2(2^{h+1}}- 1)$$

// Remember:

$$\boxed{\log_b m^n} = n * \log_b m$$

$$\log_2(n) \approx (h+1)(\log_2 2)$$

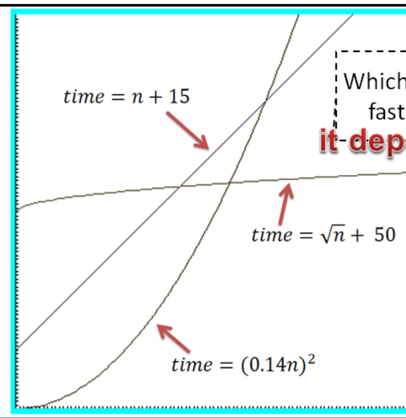
$$\boxed{log_2 n \approx h}$$

$\text{Log}_2(\text{N})$

- When we're able to keep dividing the problem in half (or thirds etc.)
- Looking through a phone book

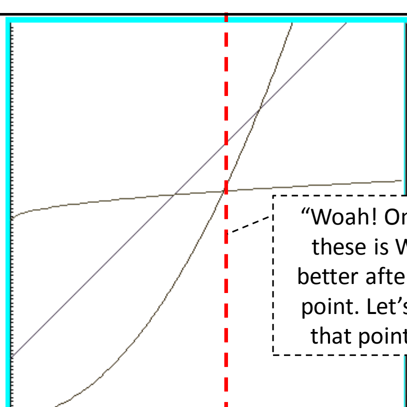
Asymptotic Cost

- We want to express the speed of an algorithm *independently* of a *specific implementation* on a *specific machine.*
- We examine the cost of the algorithms for large input sets i.e. *the asymptotic cost*.
- In later classes (CS70/CS170) you'll do this in more detail



Which one is fastest?

it depends



"Woah! One of these is WAY better after this point. Let's call that point N"

Formal definition

$$T(n) \in O(f(n))$$

if and only if

$$T(n) \leq c * f(n)$$

for all n > N

Which is fastest after some big value N?

$$time = \sqrt{n} + 50$$
$$time = (0.14n)^2$$
$$time = n + 15$$

Important Big-Oh Sets

Subset of

Function	Common Name
$O(1)$	Constant
$O(\log n)$	Logarithmic
$O(\log^2 n)$	Log-squared
$O(\sqrt{n})$	Root-n
$O(n)$	Linear
$O(n \log n)$	n log n
$O(n^2)$	Quadratic
$O(n^3)$	Cubic
$O(n^4)$	Quartic
$O(2^n)$	Exponential
$O(e^n)$	Bigger exponential

Which is fastest after some value N?

$$time = \sqrt{n} - 1000$$
$$time = 0.75\sqrt{n} + 50$$
$$time = \sqrt{n}$$

WAIT – who cares? These are all proportional!

Sometimes we do care, but for simplicity we ignore constants

Formal definition

$$T(n) \in O(f(n))$$

if and only if

$$T(n) \le c * f(n)$$

for all n > N

Simplifying stuff is important

$$f(n) \in O(5n^3 + 10n^2 + 1000n)$$

$$T(n) \in O(n^3)$$

Write `sum-up-to` to generate an iterative process!

```
(define (sum-up-to x)
  (define (sum-up-to-iter x answer)
     (if (= x 0)
        answer
        (sum-up-to-iter (- x 1)
                        (+ answer x))))
  (sum-up-to-iter x 0))
```