6.006 Recitation

Build 2008.4

Outline

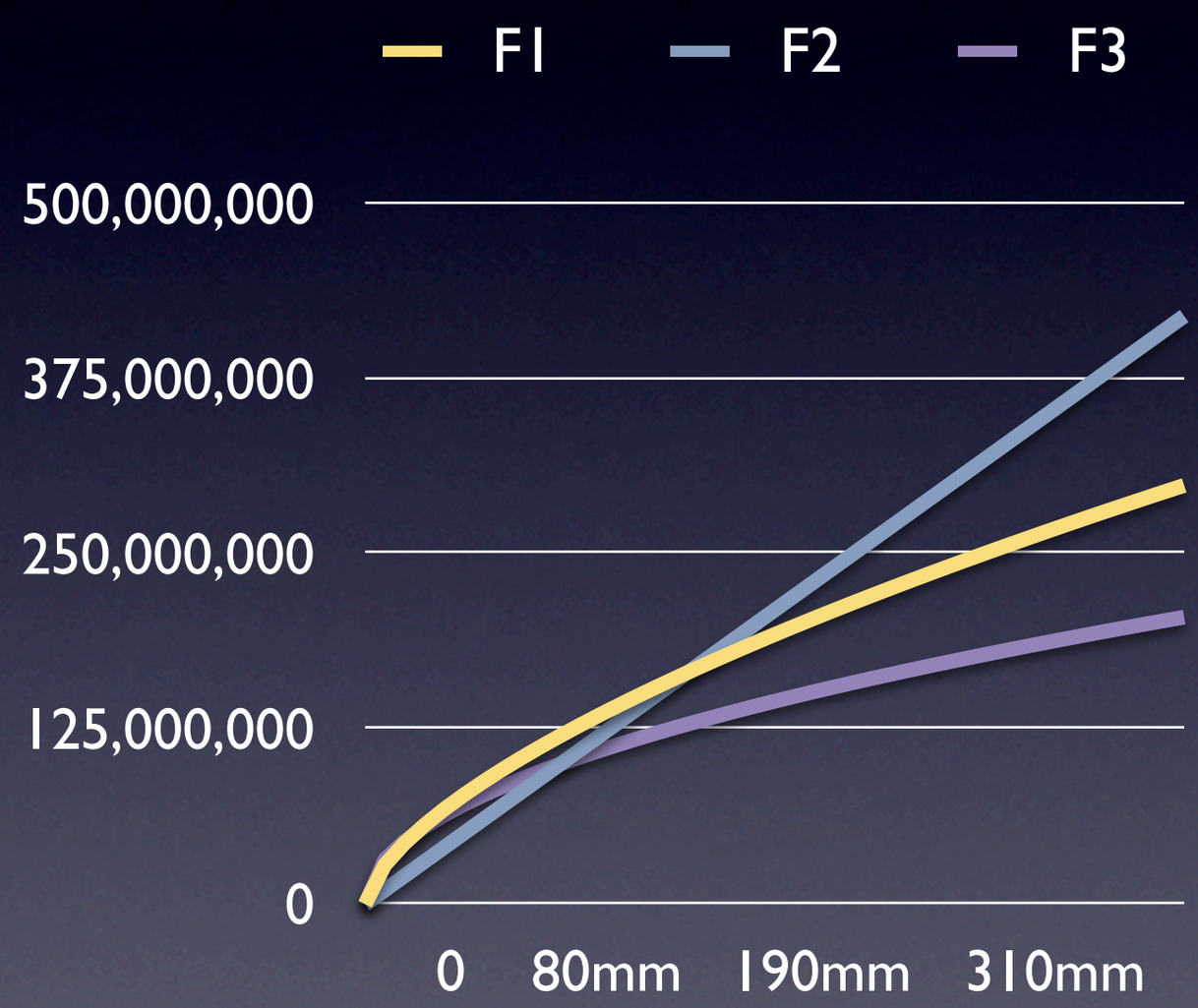
- Asymptotic Notation
- Merge-Sort
- Python Cost Model
- (maybe) docdist{5,6}.py

Asymptotic Notation

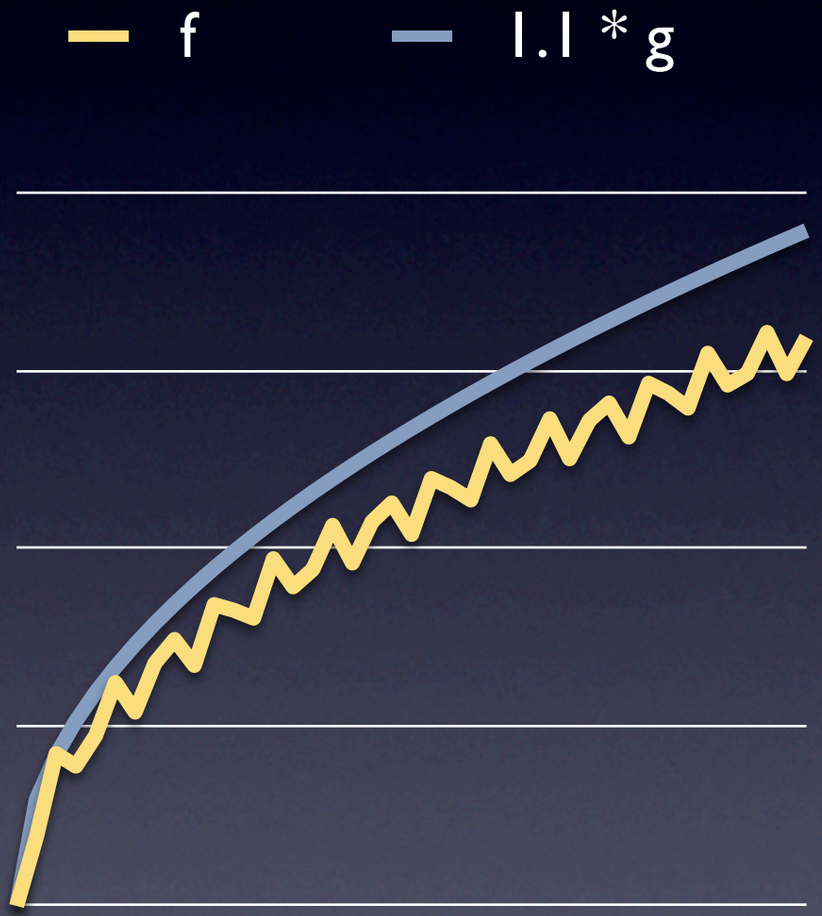
f1: 1,000 * $n^{0.635}$

f2: n

f3: 10,000 * $n^{0.5}$

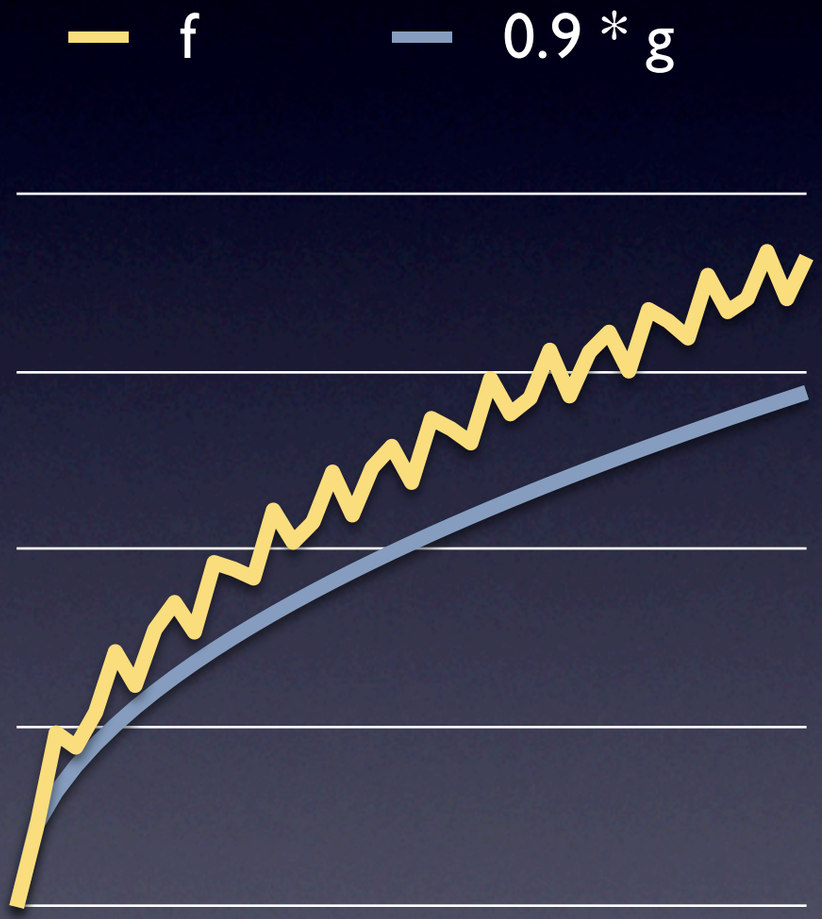


Asymptotic Notation



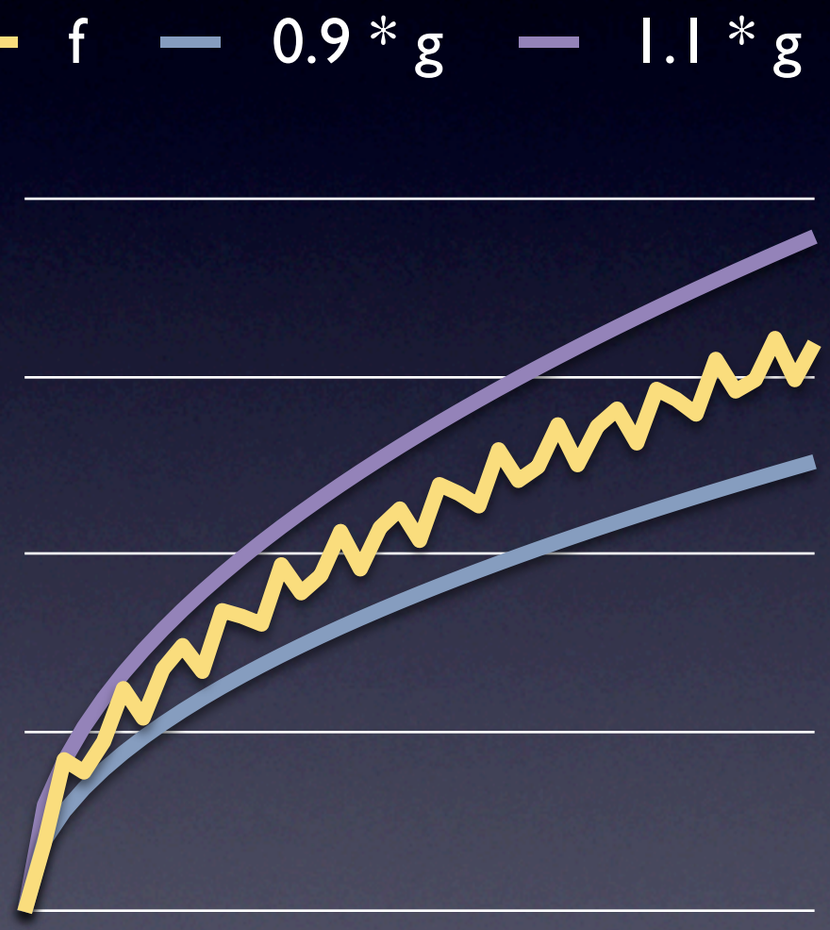
- f = O(g)

Asymptotic Notation



- $f = \Omega(g)$

Asymptotic Notation



- $f = \Theta(g)$

Asymptotic Drowning

- $\lg(n^{100}) = 100 * \lg(n) = \Theta(\lg(n))$
- $\lg(n^{0.1}) = 0.1 * \lg(n) = \Theta(\lg(n))$
- $\lg_5(n) = \lg(n) / \lg(5) = \Theta(\lg(n))$
- $n^{\lg(5)} = n^{2.3219...}$ so $n^2 < n^{\lg(5)} < n^3$

Asymptotic Headaches

- 10^{80} (atoms in the universe)
- $lg_{log5}(lg^{lg100}\ n)$
- $(20n)^7$
- $5^{lg3}n^3 + 10^{80}n^2 + (lg3)n^{3.1} + 6006$
- $lg(\binom{N}{N/2})$

Stirling Banishes the Evil

- N! ~= sqrt(2πN) * ((N/e) ^ N)
- Substitute in $\lg\binom{N}{N/2}$
- Reduce terms, obtain O(N)

Binary Search for 23

1	3	4	9	11	**15**	20	24	29	34	38

1	3	4	9	11	15	20	24	**29**	34	38

1	3	4	9	11	15	**20**	24	29	34	38

1	3	4	9	11	15	20	**24**	29	34	38

1	3	4	9	11	15	20	24	29	34	38

Divide and Conquer

1. Divide

   Break into smaller subproblems

2. Conquer

   Really small subproblems are easy

3. Profit

   Combine answers to sub-problems

Merge-Sort

1. Divide

    Break into 2 equal-sized sublists

2. Conquer

    1-element lists are sorted

3. Profit

    Merge sorted sublists

Just Merge

L1 = [2, 3, 5, 7]

L = []

L2 = [2, 4, 8, 16]

Just Merge

L1 = [3, 5, 7]

L = [2]

L2 = [2, 4, 8, 16]

Just Merge

L1 = [3, 5, 7]

L = [2, 2]

L2 = [4, 8, 16]

Just Merge

L1 = [5, 7]

L = [2, 2, 3]

L2 = [4, 8, 16]

Just Merge

L1 = [5, 7]

L = [2, 2, 3, 4]

L2 = [8, 16]

Just Merge

L1 = [7]

L = [2, 2, 3, 4, 5]

L2 = [8, 16]

Just Merge

```
LI = []
```

L = [2, 2, 3, 4, 5, 7]

L2 = [8, 16]

Just Merge

L1 = []

L = [2, 2, 3, 4, 5, 7, 8, 16]

L2 = []

Running Time Analysis

	Binary Search	Merge Sort
Subproblems	1	2
Subproblem size	N / 2	N / 2
Time to Divide	Θ(1)	Θ(1)
Time to Profit	Θ(1)	Θ(N)
T(N)	T(N/2) + Θ(1)	2T(N/2) + Θ(N)

Recursion Tree Analysis

	Binary Search	Merge Sort
T(N)	T(N/2) + Θ(1)	2T(N/2) + Θ(N)
Tree depth	lg(N)	lg(N)
Cost per level	Θ(1)	Θ(N)
Total cost	Θ(lg(N))	Θ(N*lg(N))

Python Cost Model

- Motivation
  - change + to **`extend`** for 1000X speed
- Approach
  - stand on the shoulders of giants (Ron)
  - focus on the asymptotic cost

Timing.py
a.k.a. Ron's Shoulders

```
   1  print "Test List-11: Sort"
 2  spec_string = "1000<=n<=100000"
 3  growth_factor = 2
 4  print "Spec_string: ",spec_string, "by factors of", growth_factor
 5  var_list, param_list = make_param_list(spec_string,growth_factor)
 6  # f_list = ("n","1")
 7  f_list = ("n*lg(n)",)
 8  run_times = []
 9  trials = 200
10  for D in param_list:
11      t = timeit.Timer("L.sort()",
          "import random;L=[random.random() for i in range(%(n)s)]"%D)
12      run_times.append(t.timeit(trials)*1e6/float(trials))
13  fit(var_list,param_list,run_times,f_list)
```

Pset Hint

"Good artists borrow, great artists steal"

Steve Jobs, CEO Apple Inc.

quoting Pablo Picasso

Python Lists

L, M have n **items**

Creation	list()	Θ(1)
Access	L[i]	Θ(1)
Append	L.append(0)	Θ(1)
Concatenate	L + M	Θ(n)
Pop	L.pop()	Θ(1)
Delete first	del L[0]	Θ(n)

Python Lists II

L, M have n **items**
P has n/2 **items**

Slice extraction	L[0:n/2]	Θ(n)
Slice assignment	L[0:n/2] = P	Θ(n)
Copy	L[:]	Θ(n)
Reverse	L.reverse()	Θ(n)
Sort	L.sort()	Θ(n * lg(n))

Python Strings

s, t have n **characters**

Creation	list()	Θ(1)
Extract a char	s[i]	Θ(1)
Concatenate	s + t	Θ(n)
Extract substring of n/2 characters	s[0:n/2]	Θ(n)

Python Dictionaries

D has n **items**

Creation	dict()	Θ(1)
Access	D[i]	Θ(1)
Copy	D.copy()	Θ(n)
List items	D.items()	Θ(n)

Python Cost Exercise

```
1 def merge(L,R):
2    i = 0
3    j = 0
4    answer = []
5    while i<len(L) and j<len(R):
6        if L[i]<R[j]:
7            answer.append(L[i])
8            i += 1
9        else:
10           answer.append(R[j])
11           j += 1
12   if i<len(L):
13       answer.extend(L[i:])
14   if j<len(R):
15       answer.extend(R[j:])
16   return answer
```

Θ(1)	1
Θ(1)	1
Θ(1)	1
Θ(1)	Θ(N)
Θ(1)	Θ(N)
Θ(1)	Θ(N)
Θ(1)	Θ(N)
Θ(1)	Θ(N)
Θ(1)	Θ(N)
Θ(1)	Θ(N)
Θ(1)	1
Θ(N)	Θ(1)
Θ(1)	1
Θ(N)	Θ(1)
Θ(1)	1

Python Cost Exercise II

```
1 def merge_sort(A):
2     n = len(A)
3     if n==1:
4         return A
5     mid = n//2
6     L = merge_sort(A[:mid])
7     R = merge_sort(A[mid:])
8     return merge(L,R)
```

Θ(1)	1
Θ(1)	1
Θ(1)	Θ(1)
Θ(1)	1
Θ(N)	T(N/2)
Θ(N)	T(N/2)
Θ(N)	1

Python Arithmetic

x, y, z have n **bits**

Addition	x + y	$\Theta(n)$
Subtraction	x - y	$\Theta(n)$
Multiplication	x * y	$\Theta(n^{1.585...})$
Division	x / y	$\Theta(n^2)$
Modular Exponentiation	powmod(x, y, z)	$\Theta(n^3)$
Power of 2	2 ** n	$\Theta(1)$

Python Arithmetic

x, y, z have n **digits**

Addition	x + y	$\Theta(n)$
Subtraction	x - y	$\Theta(n)$
Multiplication	x * y	$\Theta(n^{1.585...})$
Division	x / y	$\Theta(n^2)$
Modular Exponentiation	powmod(x, y, z)	$\Theta(n^3)$
Power of 2	2 ** n	$\Theta(1)$

And we're done!

- costan@mit.edu
- (617) 230-9694, no voicemail
- AIM: victorcostan
- Google Talk: costan@gmail.com
- 32G-8th Floor