CSE 326: Data Structures
Lecture #3
Analysis of Algorithms II

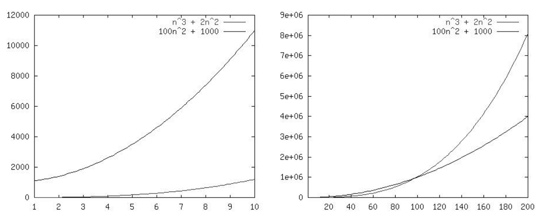
Henry Kautz
Winter 2002

Silicon Downs

Post #1	Post #2
$n^3 + 2n^2$	$100n^2 + 1000$
$n^{0.1}$	$\log n$
$n + 100n^{0.1}$	$2n + 10 \log n$
$5n^5$	$n!$
$n^{-15}2^n/100$	$1000n^{15}$
$8^{2\log n}$	$3n^7 + 7n$

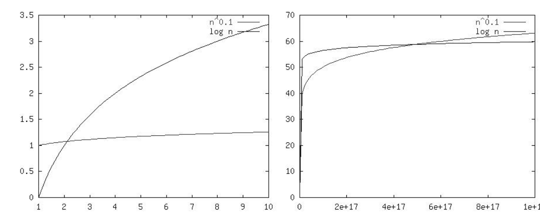
Race I

$n^3 + 2n^2$ vs. $100n^2 + 1000$



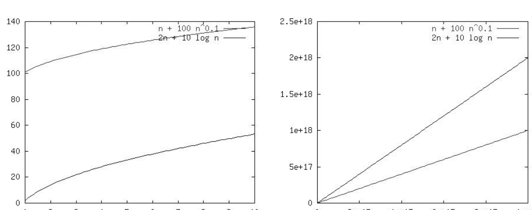
Race II

$n^{0.1}$ vs. $\log n$



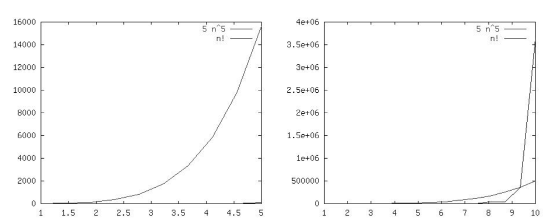
Race III

$n + 100n^{0.1}$ vs. $2n + 10 \log n$



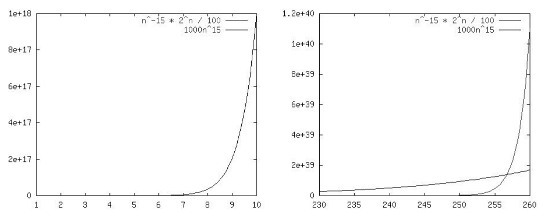
Race IV

$5n^5$ vs. $n!$



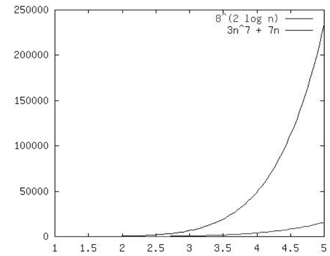
Race V

$n^{-15}2^n/100$ vs. $1000n^{15}$



Race VI

$8^{2\log(n)}$ vs. $3n^7 + 7n$



The Losers Win

Post #1	Post #2	Better algorithm!
$n^3 + 2n^2$	$100n^2 + 1000$	$O(n^2)$
$n^{0.1}$	$\log n$	$O(\log n)$
$n + 100n^{0.1}$	$2n + 10 \log n$	**TIE** $O(n)$
$5n^5$	$n!$	$O(n^5)$
$n^{-15}2^n/100$	$1000n^{15}$	$O(n^{15})$
$8^{2\log n}$	$3n^7 + 7n$	$O(n^6)$

Common Names

constant:	$O(1)$	
logarithmic:	$O(\log n)$	
linear:	$O(n)$	
log-linear:	$O(n \log n)$	
superlinear:	$O(n^{1+c})$	(c is a constant > 0)
quadratic:	$O(n^2)$	
polynomial:	$O(n^k)$	(k is a constant)
exponential:	$O(c^n)$	(c is a constant > 1)

Analyzing Code

- C++ operations - constant time
- consecutive stmts - sum of times
- conditionals - sum of branches, condition
- loops - sum of iterations
- function calls - cost of function body
- recursive functions - solve recursive equation

Above all, use your head!

Nested Loops

```
for i = 1 to n do
  for j = 1 to n do
    sum = sum + 1
```

Nested Loops

```
for i = 1 to n do
  for j = 1 to n do
    sum = sum + 1
```

$$\sum_{i=1}^{n} \sum\nolimits_{j=1}^{n} 1 = \sum_{i=1}^{n} n = n^2$$

Nested Dependent Loops

```
for i = 1 to n do
  for j = i to n do
    sum = sum + 1
```

Nested Dependent Loops

```
for i = 1 to n do
  for j = i to n do
    sum = sum + 1
```

$$\sum_{i=1}^{n} \sum_{j=1}^{n} 1 = \sum_{i=}^{n} (n-i-1) = \sum_{i=1}^{n} (n+1) - \sum_{i=1}^{n} i =$$

?

Arithmetic Series

$$S(N) = 1 + 2 + \ldots + N = \sum_{i=1}^{N} i = ?$$

- The sum is: S(1) = 1, S(2) = 3, S(3) = 6, S(4) = 10, …
- Is S(N) = N(N+1)/2 ?

  Prove by induction

  – Base case: for N = 1,   S(N) = 1(2)/2 = 1 ➨
  – Assume true for N = k
  – Suppose N = k+1.
  – S(k+1) = 1 + 2 + …+ k + (k+1) = S(k) + (k+1)

  = k(k+1)/2 + (k+1) = (k+1)(k/2 + 1) = (k+1)(k+2)/2. ➨

Other Important Series

(know them well!)

- Sum of squares: $\sum_{i=1}^{N} i^2 = \frac{N(N+1)(2N+1)}{6} \approx \frac{N^3}{3}$ for large N
- Sum of exponents: $\sum_{i=1}^{N} i^k \approx \frac{N^{k+1}}{|k+1|}$ for large N and k ≠ -1
- Harmonic series (k = -1): $\sum_{i=1}^{N} \frac{1}{i} \approx \log_e N$ for large N
  – $\log_e N$ (or ln N) is the natural log of N
- Geometric series: $\sum_{i=0}^{N} A^i = \frac{A^{N+1}-1}{A-1}$

Nested Dependent Loops

```
for i = 1 to n do
  for j = i to n do
    sum = sum + 1
```

$$\sum_{i=1}^{n} \sum_{j=1}^{n} 1 = \sum_{i=}^{n} (n-i-1) = \sum_{i=1}^{n} (n+1) - \sum_{i=1}^{n} i =$$

$$n(n+1) - \frac{n(n+1)}{2} = \frac{n(n+1)}{2} = O(n^2)$$

Conditionals

- Conditional
  `if C then S1 else S2`

- Suppose you are doing a O( ) analysis?

- Suppose you are doing a Ω( ) analysis?

Value-Dependent Operations

```
for (i = 1; i<k && A[i]==A[i+i]; i++);
```

What does this do?

- Suppose you are doing a O( ) analysis?

- Suppose you are doing a Ω( ) analysis?

Some Educational Statistics

- Most teachers speak at a rate of 100-200 words per minute. If students really concentrate, they can understand about 50-100 words per minute.
- During a lecture, about 40% of the students are thinking about something else.
- Students of lecture-based courses show that students remember about 8% of the material after the course is over.

Active Learning

- Get up and stretch!

- For Monday:
  - Read Sections 2.4.3 and 2.4.4. Then:
  - Read Section 7.6 (Mergesort) carefully
  - On a single side of a sheet of paper, summarize in your own words the major steps in analyzing a recursive procedure. Bring **two** copies to class.