

## Objectives

- Analyzing algorithms
- Asymptotic running times

Jan 15, 2010

Sprenkle - CSCI211

1

## Discussion: Quizzes vs Journals

- Assessment: some preference to journals
  - Check out Wiki on Sakai

Jan 15, 2010

Sprenkle - CSCI211

2

## Review: Our Process

- Understand/identify problem
  - Simplify as appropriate
- Design a solution
- Analyze
  - Correctness, efficiency
  - May need to go back to step 2 and try again
- Implement
  - Within bounds shown in analysis

Jan 15, 2010

Sprenkle - CSCI211

3

## Computational Tractability

As soon as an Analytic Engine exists, it will necessarily guide the future course of the science. Whenever any result is sought by its aid, the question will arise - *By what course of calculation can these results be arrived at by the machine in the shortest time?*

-- Charles Babbage



Charles Babbage  
(1864)



Analytic Engine  
(schematic)

Jan 15, 2010

Sprenkle - CSCI211

4

## Today's Goal: Define Algorithm Efficiency

Jan 15, 2010

Sprenkle - CSCI211

5

## Brute Force

- For many non-trivial problems, there is a natural brute force search algorithm that checks every possible solution
  - Typically takes  $2^N$  time or worse for inputs of size  $N$
  - Unacceptable in practice

How many possible solutions are there in the stable matching problem?

(In other words, how many possible perfect matchings are there? We're not worried about stability right now.)

Jan 15, 2010

Sprenkle - CSCI211

6

## Brute Force

- For many non-trivial problems, there is a natural brute force search algorithm that checks every possible solution
  - Typically takes  $2^N$  time or worse for inputs of size  $N$
  - Unacceptable in practice
- Example: Stable matching:  $n!$  with  $n$  men and  $n$  women
  - If  $n$  increases by 1, what happens to the running time?

Jan 15, 2010

Sprenkle - CSCI211

7

## Worst-Case Running Time

- Obtain bound on *largest possible* running time of algorithm on input of a given size  $N$ 
  - Generally captures efficiency in practice
  - Draconian view, but hard to find effective alternative

What are alternatives to worst-case analysis?

Jan 15, 2010

Sprenkle - CSCI211

8

## Average Case Running Time

- Obtain bound on running time of algorithm on *random* input as a function of input size  $N$ 
  - Hard (or impossible) to accurately model real instances by random distributions
  - Algorithm tuned for a certain distribution may perform poorly on other inputs

Jan 15, 2010

Sprenkle - CSCI211

9

## Polynomial-Time

There exists constants  $c > 0$  and  $d > 0$  such that on every input of size  $N$ , its running time is bounded by  $c N^d$  steps.

- Desirable scaling property:** When input size doubles, algorithm should only slow down by some constant factor  $C$  (choose  $C = 2^d$ )
- Def.** An algorithm is *polynomial time* (or *polytime*) if the above scaling property holds.

Jan 15, 2010

Sprenkle - CSCI211

10

## Algorithm Efficiency

- Def.** An algorithm is *efficient* if its running time is *polynomial*
- Justification:** It really works in practice!
  - In practice, poly-time algorithms that people develop almost always have low constants and low exponents
  - Breaking through the exponential barrier of brute force typically exposes some crucial structure of the problem
- Exceptions**
  - Some poly-time algorithms do have high constants and/or exponents ( $6.02 \times 10^{23} \times N^{20}$ ) and are useless in practice
  - Some exponential-time (or worse) algorithms are widely used because the worst-case instances seem to be rare

Jan 15, 2010

Sprenkle - CSCI211

11

## Running Times

Table 2.1 The running times (rounded up) of different algorithms on inputs of increasing size, for a processor performing a million high-level instructions per second. In cases where the running time exceeds  $10^{25}$  years, we simply record the algorithm as taking a very long time.

Input Size	$n$	$n \log n$	$n^2$	$n^3$	$1.5^n$	$2^n$	$n!$
$n = 10$	< 1 sec	< 1 sec	< 1 sec	< 1 sec	< 1 sec	< 1 sec	4 sec
$n = 30$	< 1 sec	< 1 sec	< 1 sec	< 1 sec	< 1 sec	18 min	$10^{25}$ years
$n = 50$	< 1 sec	< 1 sec	< 1 sec	< 1 sec	11 min	36 years	very long
$n = 100$	< 1 sec	< 1 sec	< 1 sec	1 sec	12,892 years	$10^{17}$ years	very long
$n = 1,000$	< 1 sec	< 1 sec	1 sec	18 min	very long	very long	very long
$n = 10,000$	< 1 sec	< 1 sec	2 min	12 days	very long	very long	very long
$n = 100,000$	< 1 sec	2 sec	3 hours	32 years	very long	very long	very long
$n = 1,000,000$	1 sec	20 sec	12 days	31,710 years	very long	very long	very long

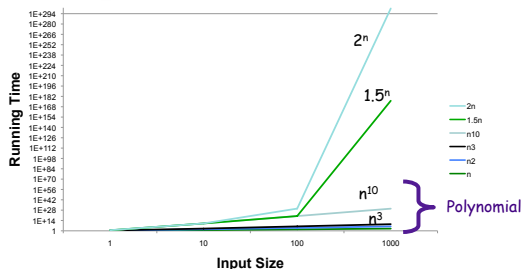
Polynomial

Jan 15, 2010

Sprenkle - CSCI211

12

## Running Times



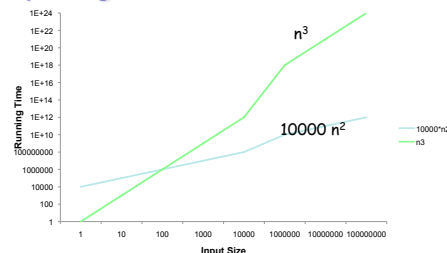
- Huge difference from polynomial to not polynomial
- Differences in runtime matter more as input size increases

Jan 15, 2010

Sprengle - CSCI211

13

## Comparing $10000 n^2$ and $n^3$



- As input size increases,  $n^3$  dominates large constant  $\cdot n^2$
- Care about running time as input size approaches infinity
- Only care about highest-order term

Jan 15, 2010

Sprengle - CSCI211

14

## Asymptotic Order of Growth: Upper Bounds

- $T(n)$  is the worst case running time of an algorithm
- We say that  $T(n)$  is  $O(f(n))$  if there exist constants  $c > 0$  and  $n_0 \geq 0$  such that for all  $n \geq n_0$ , we have  $T(n) \leq c \cdot f(n)$

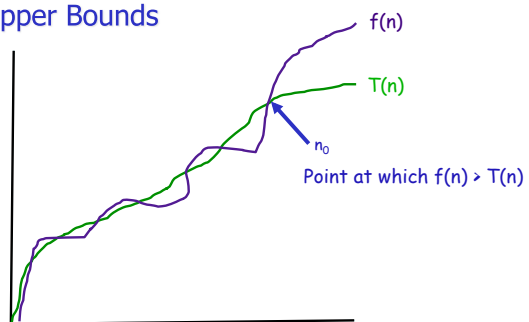
→  $T$  is **asymptotically upperbounded** by  $f$

Jan 15, 2010

Sprengle - CSCI211

15

## Asymptotic Order of Growth: Upper Bounds



Jan 15, 2010

Sprengle - CSCI211

16

## Upper Bounds Example

- $T(n) = pn^2 + qn + r$
- $p, q, r$  are positive constants

**Idea:** Let's inflate the terms in the equation so that all terms are  $n^2$

Jan 15, 2010

Sprengle - CSCI211

17

## Upper Bounds Example

- $T(n) = pn^2 + qn + r$
- $p, q, r$  are positive constants
- For all  $n \geq 1$ ,
 

$$\begin{aligned} T(n) &= pn^2 + qn + r \\ &\leq pn^2 + qn^2 + rn^2 \\ &= (p+q+r) n^2 \\ &= c n^2 \end{aligned}$$
- $T(n) \leq cn^2$ , where  $c = p+q+r$
- $T(n) = O(n^2)$
- Also correct to say that  $T(n) = O(n^3)$

Jan 15, 2010

Sprengle - CSCI211

18

## Notation

- Slight abuse of notation.  $T(n) = O(f(n))$ 
  - Asymmetric:
    - $f(n) = 5n^3$ ;  $g(n) = 3n^2$
    - $f(n) = O(n^3) = g(n)$
    - But  $f(n) \neq g(n)$ .
  - Better notation:  $T(n) \in O(f(n))$
- Meaningless statement. Any comparison-based sorting algorithm requires *at least*  $O(n \log n)$  comparisons
  - Use  $\Omega$  for lower bounds

Jan 15, 2010

Sprenkle - CSCI211

19

## Asymptotic Order of Growth: Lower Bounds

- Complementary to upper bound

- $T(n)$  is  $\Omega(f(n))$  if there exist constants  $\epsilon > 0$  and  $n_0 \geq 0$  such that for all  $n \geq n_0$ , we have

$$T(n) \geq \epsilon \cdot f(n)$$

$\epsilon$  cannot depend on  $n$   
sufficiently large  $n$   
 $T(n)$  is bounded below by a constant multiple of  $f(n)$

→  $T$  is asymptotically lowerbounded by  $f$

Jan 15, 2010

Sprenkle - CSCI211

20

## Example: Lower Bound

- $T(n) = pn^2 + qn + r$ 
  - $p, q, r$  are positive constants
- Idea: Deflate terms rather than inflate
- For all  $n \geq 0$ ,
 

$$T(n) = pn^2 + qn + r \geq pn^2$$

$$\rightarrow T(n) \geq \epsilon n^2, \text{ where } \epsilon = p > 0$$

$$\rightarrow T(n) = \Omega(n^2)$$
- Also correct to say that  $T(n) = \Omega(n)$

Jan 15, 2010

Sprenkle - CSCI211

21

## Tight bounds

$T(n)$  is  $\Theta(f(n))$  if  $T(n)$  is both  $O(f(n))$  and  $\Omega(f(n))$

- The "right" bound

Jan 15, 2010

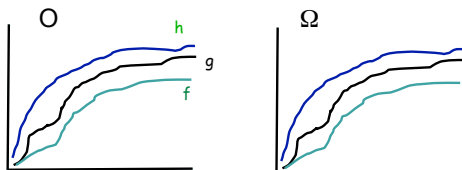
Sprenkle - CSCI211

22

## Property: Transitivity

- If  $f = O(g)$  and  $g = O(h)$  then  $f = O(h)$
- If  $f = \Omega(g)$  and  $g = \Omega(h)$  then  $f = \Omega(h)$
- If  $f = \Theta(g)$  and  $g = \Theta(h)$  then  $f = \Theta(h)$

Proofs in book



Jan 15, 2010

Sprenkle - CSCI211

23

## Property: Additivity

- If  $f = O(h)$  and  $g = O(h)$  then  $f + g = O(h)$
- If  $f = \Omega(h)$  and  $g = \Omega(h)$  then  $f + g = \Omega(h)$
- If  $f = \Theta(h)$  and  $g = \Theta(h)$  then  $f + g = \Theta(h)$

Proofs in book

### Sketch proof for $O$ :

By defn,  $f \leq c \cdot h$   
By defn,  $g \leq d \cdot h$   
 $f + g \leq c \cdot h + d \cdot h = (c + d)h = c' \cdot h$   
→  $f + g$  is  $O(h)$

Jan 15, 2010

Sprenkle - CSCI211

24