CSE 326
Lecture 4

Analysis of Algorithms

Winter Quarter 1999

Analysis of Algorithms

- Analysis of an algorithm provides insight into how long the program will run and how much storage it will need

  - Time Complexity

  - Space Complexity.
- It is not uncommon that an analysis provides insight into alternative algorithms.
- Input size is usually indicated by a number n. Sometimes there are multiple inputs.
- Running time is a function of n such as:

  n, n^2, $n \log n$, 2^n
- Computer Scientist and Engineers typically use order notation so as not to bother with constant factors and low order terms.

  $20n^2 + 500n + 10000$

  low order terms

Rates of Growth

- Suppose a computer executes 10^{12} operations per second.

Input Size

Time Complexty

	10	100	1,000	10,000
n	10^{-11} s	10^{-10} s	10^{-9} s	10^{-8} s
n log n	10^{-11} s	10^{-9} s	10^{-8} s	10^{-7} s
n^2	10^{-10} s	10^{-8} s	10^{-6} s	10^{-4} s
n^3	10^{-9} s	10^{-6} s	10^{-3} s	1 s
n^4	10^{-8} s	10^{-4} s	1 s	10^{4} s
2^n	10^{-9} s	10^{18} s	10^{289} s	

10^4 s = 2.8 hr

10^{18} s = 30 billion years

Order Notation

- $T(n) = O(f(n))$ if there are constants c and n_0 such that $T(n) \le cf(n)$ for all $n \ge n_0$.
- $T(n) = \Omega(f(n))$ if there are constants c and n_0 such that $T(n) \ge cf(n)$ for all $n \ge n_0$.
- $T(n) = \Theta(f(n))$ if $T(n) = O(f(n))$ and $T(n) = \Omega(f(n))$.
- Examples:

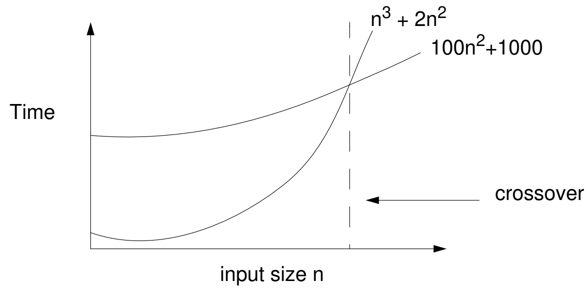
  $10{,}000\, n^2 + 25\, n = \Theta(n^2)$

  $10^{-10}\, n^2 = \Theta(n^2)$

  $n \log n = O(n^2)$

  $n \log n = \Omega(n)$

Understanding Orders of Magnitude



Some Basic Relationships

- if for all $c > 0$ there is an n_0 such that $f(n) \le cg(n)$ for all $n \ge n_0$ then f(n) will crossover g(n).

  Example: $10{,}000\ n^2$ and n^3
- Polynomial time is time $O(n^k)$ for some k.
- Exponential time is time $O(c^n)$ for some $c > 1$.
- Logarithmic time is time O(log n)

  Note that in order notation the base of the logarithm is ignored. Why? Because $\log_k n = \log_k 2 \log_2 n$ for any k.
- $c^n = \Omega(n^k)$ for any $c > 1$ and any k.
- $\log^k n = O(n)$ for any k.

Kinds of Analysis

- Most common are time and storage "asymptotic" analysis

  - ignores constant factors and low order terms
- Upper bounds or lower bounds

  - $T(n) = O(f(n))$ implies f(n) is an upper bound

  - $T(n) = \Omega(f(n))$ implies f(n) is a lower bound
- Worst case - worst over all possible inputs of length n
- Average case - assuming some distribution of inputs.
- Amortized analysis.

  - Complexity averaged over many operations vs. complexity of each operation.
- "Common case analysis"
- Many other kinds of analysis are possible

  - Example: cache miss analysis

Example 1

- For loops - assume k = 0 initially.

```
for i = 1 to n do
  for j = 1 to n do
     k := k + 1
```

$$\sum_{i=1}^{n}\left(\sum_{j=1}^{n} 1\right) = \sum_{i=1}^{n} n = n^2$$

Example 2

- for i = 1 to n do
    for j = i to n do
        k := k + 1

$$\sum_{i=1}^{n}\left(\sum_{j=i}^{n} 1\right) = \sum_{i=1}^{n} (n-i+1) = \sum_{i=1}^{n} (n+1) - \sum_{i=1}^{n} i = n(n+1) - \frac{n(n+1)}{2}$$

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

Example 3

- Conditional
  if C then S1 else S2
  time $\leq$ time of C + max(time of S1, time of S2)
- While loop
  while C do S
  time = sum of times of each iteration of the loop

Example 4

- Recursion
- Analysis almost always yields a recurrence of some form.
- Recursive max.
  $T(0) \leq b$
  $T(n) \leq c + T(n-1)$ if $n > 0$
- Analysis
  $T(n) \leq c + c + T(n-2)$ by substitution
  $\leq c + c + c + T(n-3)$ by substitution again
  $\leq kc + T(n - k)$ for all k from 1 to n.
  $\leq nc + T(0)$
  $\leq cn + b$
- $T(n) = O(n)$

Example 5

- Binary Search -

1	2	3	4	5	6	7	8	9	10
7	12	30	35	75	83	87	90	97	99

- $T(1) \leq b$
  $T(n) \leq T(n/2) + c$ if $n > 1$
- Analysis
  $T(n) \leq T(n/2) + c$
  $\leq T(n/4) + c + c$
  $\leq T(n/8) + c + c + c$
  $\leq T(n/2^k) + kc$
  $\leq T(1) + kc$ where $n = 2^k$
  $\leq b + c \log_2 n$
  $= O(\log n)$

Example 6

- Mergesort

  Basic idea: mergesort the first half, mergesort the second half then merge the two halves together.

- $T(1) \le b$

  $T(n) \le 2T(n/2) + cn \quad \text{if } n > 1$

- Analysis - for simplicity assume that n is a power of 2

  $T(n) \le 2T(n/2) + cn$
  $\le 2(2T(n/4) + cn/2) + cn$
  $= 4T(n/4) + cn + cn$
  $\le 4(2T(n/8) + c(n/4)) + cn + cn$
  $= 8T(n/8) + cn + cn + cn$
  $\le 2^k T(n/2^k) + kcn$
  $\le 2^k T(1) + kcn$ where $n = 2^k$
  $\le bn + cn \log_2 n$
  $= O(n \log n)$