CS 255, Spring 2014, SJSU

Growth of functions. Asymptotic notation.

Fernando Lobo

Motivation

- Have a way of describing the scalability of algorithms.
- Example: What happens to execution time when we double the input size?
- We're interested in the order of growth for an algorithm's execution time.
- Use asymptotic notation to describe the order of growth.
  - O → big oh
  - Ω → big omega
  - Θ → big theta

O notation

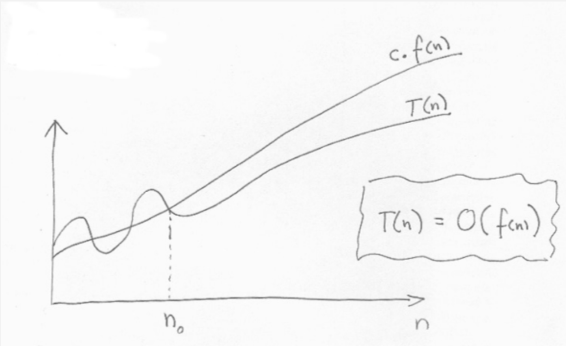
- Let $T(n)$ be a function (ex: time taken by INSERTION-SORT to sort n elements.)
- We say $T(n) = \mathrm{O}(f(n))$ if for sufficiently large values of n,

  $T(n) \leq c \cdot f(n)$, for some positive constant c
- '=' sign doesn't mean 'equal', means $\in$
- We say $T(n)$ is of order $f(n)$
- The reverse is not true.

O notation

- Formally $\mathrm{O}(f(n))$ is a set of functions.

  $\mathrm{O}(f(n)) = \{T(n) : \exists_{c>0}\ \exists_{n_0>0}\ \forall_{n>n_0}\ 0 \leq T(n) \leq c \cdot f(n)\}$



- $f(n)$ is an asymptotic upper bound for $T(n)$, up to a constant factor.

Example

- $3n^2 = O(n^2)$
- Why? Because we can find a constant $c > 0$ and $n_0 > 0$ such that: $3n^2 \leq c \cdot n^2$, $\forall_{n>n_0}$
- For example $c = 3$, $n_0 = 1$
(could also choose $c = 5$, $n_0 = 83$)
- The important thing is to find some $\underline{c}$ and some $\underline{n_0}$.

Another example

- $3n^2 = O(n^3)$
- We can find a c and n_0 such that: $3n^2 \leq c \cdot n^3$, $\forall_{n>n_0}$
- For example: $c = 3$, $n_0 = 1$

Another example

- $1000n^2 + 1000n = O(n^2)$
- We must show that $1000n^2 + 1000n \leq c \cdot n^2$, for some $c > 0$, $\forall_{n>n_0}$
- Ex: $c = 1200$

$$1000n^2 + 1000n \leq 1200n^2$$
$$1000n \leq 200n^2$$
$$n \geq 5$$

- In summary: For $c = 1200$ and $n_0 > 5$,

$$1000n^2 + 1000n \leq c \cdot n^2$$

Another example

- Show that $T_{\text{INSERTION-SORT}}(n) = an^2 + bn + c$ is $O(n^2)$, with a, b, c constants.
- $\exists_{k>0}\ \exists_{n_0>0}\ :\ an^2 + bn + c \leq kn^2$, $\forall_{n>n_0}$
- We only need to observe that for $n \geq 1$: ($\rightsquigarrow$ 1 is our n_0)
  - $bn \leq bn^2$
  - $c \leq cn^2$
- Thus:

$$\begin{aligned} an^2 + bn + c &\leq an^2 + bn^2 + cn^2 \\ &= (a + b + c)n^2 \\ &= kn^2 \end{aligned}$$

$n_0 = 1$, $k = (a + b + c)$ $\square$

O notation gives us an upper bound

- $T_{\textsc{Insertion-Sort}}(n) = O(n^2)$
- and is also $O(n^3)$
- and also $O(n^7)$
- and also $O(2^n)$
- etc.

Ω notation

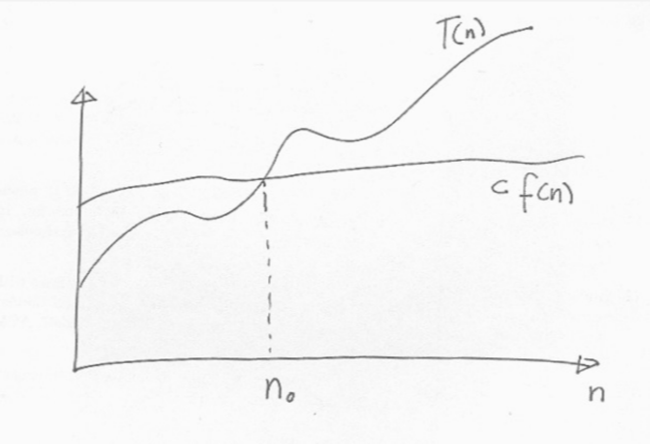
- asymptotic lower bound
- $T(n) = \Omega(f(n))$ if for sufficiently large values of n,

$$T(n) \geq c \cdot f(n), \text{ for some positive constant } c$$

- Formally $\Omega(f(n))$ is also a set of functions:

$$\Omega(f(n)) = \{T(n) : \exists_{c>0}\ \exists_{n_0>0}\ \forall_{n>n_0}\ 0 \leq c \cdot f(n) \leq T(n)\}$$

Graphically



Examples

Examples of functions in $\Omega(n^2)$

- n^2
- $n^2 + n$
- $1000n^2 + 1000n$
- n^3
- 2^n

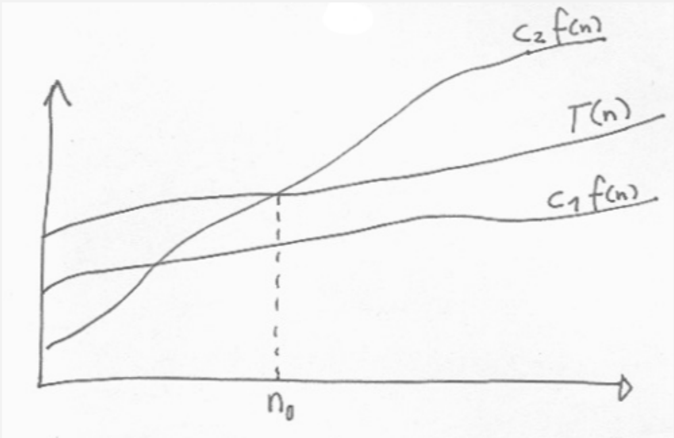
Θ notation

- asymptotic tight bound

$$\Theta(f(n)) = \{T(n) : \exists_{c_1>0}\ \exists_{c_2>0}\ \exists_{n_0>0} \forall_{n>n_0}\ :\ 0 \leq c_1 \cdot f(n) \leq T(n) \leq c_2 \cdot f(n)\}$$

- $T(n)$ is between $c_1 \cdot f(n)$ and $c_2 \cdot f(n)$, for $n > n_0$.

Graphically



Theorem

$f(n) = \Theta(g(n))$

iff:

$f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$

Typical running times

Complexity	Examples
$\Theta(1)$	Hashing
$\Theta(\lg n)$	Binary search, searching in a balanced tree
$\Theta(n)$	Linear search
$\Theta(n \lg n)$	MergeSort, QuickSort, HeapSort
$\Theta(n^2)$	InsertionSort
$\Theta(n^3)$	Traditional algorithm for matrix multiplication
$\Theta(a^n)$, $a > 1$	Set partitioning
$\Theta(n!)$	Travelling salesman problem

- Algorithms with polynomial complexity are usually referred as tractable; the others as intractable.

Math review

- Exponentials and logarithms occur often in analysis of algorithms.
- Exponentials:
  - $a^m \cdot a^n = a^{m+n}$
  - $(a^m)^n = a^{m \cdot n}$
  - $a^{-1} = 1/a$
- Logarithms:
  - $a = b^{\log_b a}$
  - $\log_c(a \cdot b) = \log_c a + \log_c b$
  - $\log_b a^n = n \cdot \log_b a$
  - $\log_b a = \frac{\log_c a}{\log_c b}$
  - $\log_b a = \frac{1}{\log_a b}$

Math review

- Logarithms grow slower than polynomials.
- Polynomials grow slower than exponentials.

$$\lim_{n\to\infty} \frac{n^b}{a^n} = 0 \qquad \forall_{a,b\,\in\,\mathbb{R}\ ,\ a>1}$$

- Therefore:

$$n^b = O(a^n) \qquad \forall_{a,b\,\in\,\mathbb{R}\ ,\ a>1}$$

- Examples:

$$\begin{aligned} 1000n^3 &= O(2^n) \\ 1000n^{500} &= O(1.000001^n) \end{aligned}$$