60-254, Lecture 2

Algorithm Analysis

Asish Mukhopadhyay, School of Computer Science, University of Windsor

Big-Oh notation

- Let *f(n)* and *g(n)* be non-negative functions of the non-negative integer *n.* Then,

  $$f(n) = O(g(n))$$

  is shorthand for

  $$f(n) \leq c*g(n)$$

  for $n \geq n_0$ and $c > 0$

Graphical Illustration

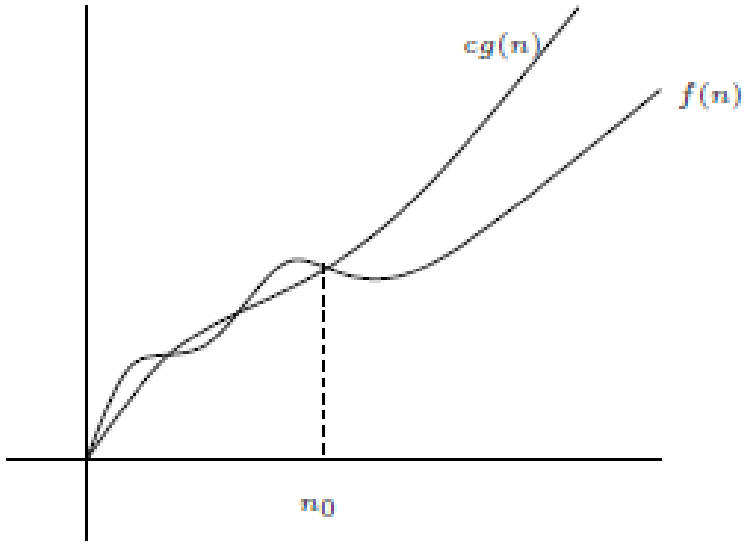

Figure 2.1: Graphical illustration of $f(n) = O(g(n))$

Learning tip

- Big-Oh means *ceiling* or *upper bound*

Notion of a dominant term

- Let n be a positive integer. Consider the function

$$f(n) = 2n^3 + n^2 + n + 1$$

- $2n^3$ determines how $f(n)$ increases as n grows large

Growth Table

n	1	n	n^2	$2n^3$	$f(n)$
10^0	1	1	1	2	5
10^1	1	10	100	2000	2111
10^2	1	100	10000	2000000	2010101
10^3	1	1000	1000000	2000000000	2001 001 001

Formally

- A term *a(n)* of a function *f(n)* **dominates** another term *b(n)*
  - if *b(n)/a(n)* goes to 0 as *n* goes to ∞

Examples

- If $f(n) = \sqrt{n} + \log n + 1$, the term $\sqrt{n}$ dominates both $\log n$ and 1

- If $f(n) = 2^n + n^2 + n \log n + 1$, the term 2^n dominates the remaining 3 terms

Reinforce

- Let $f(n) = n^4 + n^3 + n^2 + 1$. Prove formally that $f(n) = O(n^4)$.
- Let $T(n)$ be the running time of the program fragment. Give a big-Oh estimate of $T(n)$.

```
for(int i = 0; i < n; i++)
    for(int j = 0; j < n; j++)
        for(int k = 0; k < j; k++)
        sum++;
```

Running time and Input size

- Given a table like

Input Size	Time
I_1	T_1
I_2	T_2
.	.
.	.
.	.

- Can we find $T = f(I)$?

Too ambitious!

- Instead, try to find a ceiling or upper bound for *T*

- Mathematically, find *f(I)* such that
  - *T (I) = O(f(I)),* where *I* is the input size

Some examples

- *Maximum* of a list of *n* elements
  - $T(n) = O(n)$
- Find a *closest pair* of a set of *n* points
  - $T(n) = O(n^2)$
- Given *n* points in a plane determine if any 3 are *collinear*
  - $T(n) = O(n^3)$