Write down the definition of Big Oh.

Use it to prove that:

1. $T(n) = 6n^4 - 60n^2 + 7$ is in $O(n^4)$.

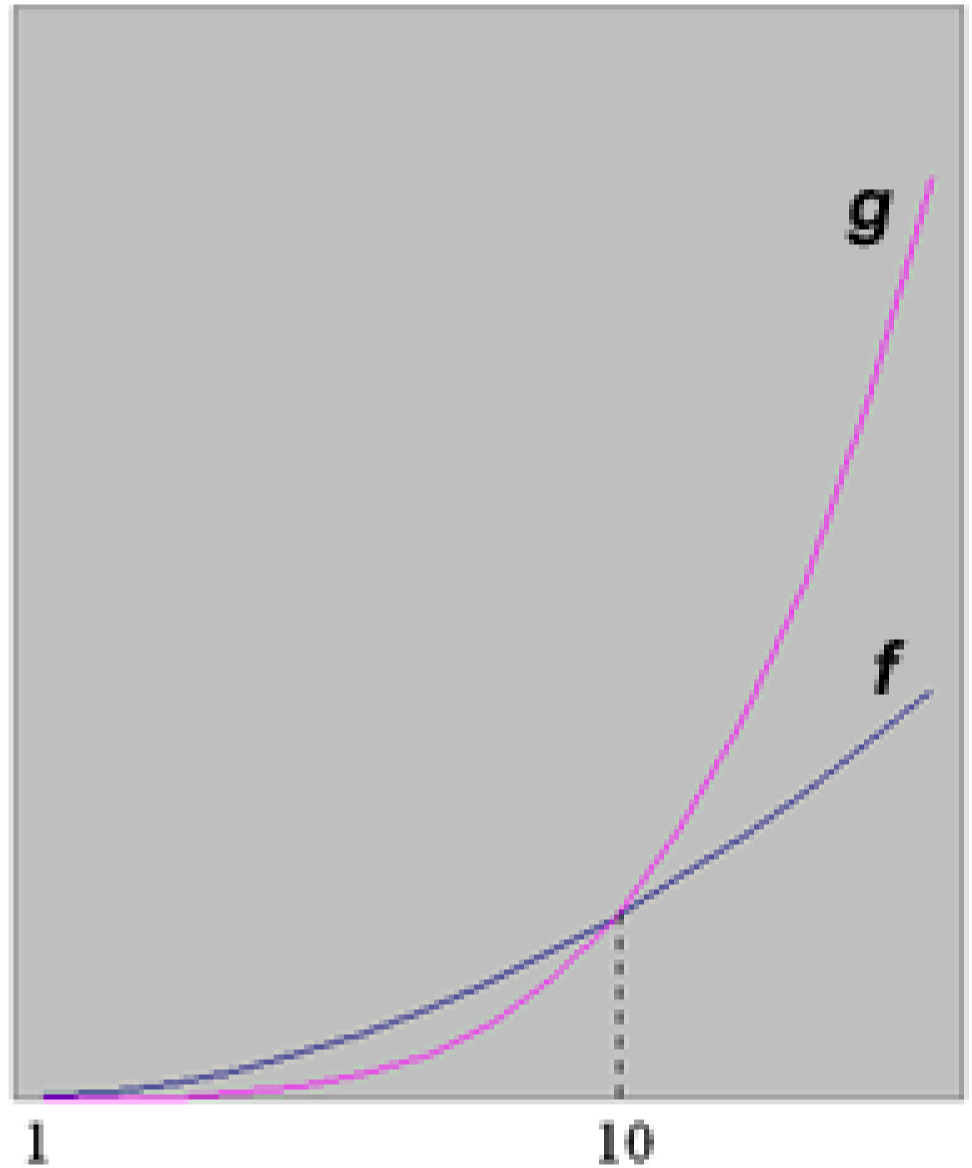
2. $T(n) = 2^0 + 2^1 + 2^2 + \ldots + 2^n$ is in $O(2^n)$.


SCRATCH
Teach kids in grades 6-9 to code!
Winners will be PAID to develop games that we'll host on csc.uvic.ca
PITCH YOUR SCRATCH PROJECT IDEA
FRIDAY SEPTEMBER 27 at 2:30 IN ECS 660

$f(n) = 100\, n^2$

$g(n) = n^4$

n	f(n)	g(n)
10	10,000	10,000
50	250,000	6,250,000
100	1,000,000	100,000,000
150	2,250,000	506,250,000



Example from:

http://www.cs.odu.edu/~toida/nerzic/content/function/growth.html

Assume that T, f are functions mapping the natural numbers {0, 1, 2, 3, ...} into the reals.

Definition: "Big Oh" A function $T(n)$ is in $O(f(n))$ if there exist constants $n_0 \geq 0$, and $c > 0$, such that for all $n \geq n_0$, $T(n) \leq c * f(n)$.

Important: here I differ from older usage in defining $O(f(n))$ to be a *set* of functions. This will prove useful later.

Big-Oh	**Informal name**
O(1)	constant
O(log n)	logarithmic
O(n)	linear
O(n log n)	n log n
O(n^2)	quadratic
O(n^3)	cubic
O(2^n)	exponential
O(n^c) for constant c,	polynomial time

Assume that T, f and g are functions mapping the natural numbers {0, 1, 2, 3, ...} into the reals.

Definition: "Omega" A function T(n) is in $\Omega(f(n))$ if there exist constants $n_0 \geq 0$, and $c > 0$, such that for all $n \geq n_0$, $T(n) \geq c * f(n)$.

Definition: "Theta" The set $\theta(g(n))$ of functions consists of $\Omega(g(n)) \cap O(g(n))$.

Regular Office hours:

Monday: no office hours.

TWF: 12:30-1:30

TWF: either 1:30-2:30 if I have no meeting scheduled, or 2:30-3:30 otherwise. The slots for the week will be announced in class or you can ask me by e-mail.

Thursdays: by appointment for students who have classes during the other office hours.

Announcements

Office hours this week:

T 12:30, 2:30

W 12:30, 1:30

F 12:30, 1:30

Please let me know if you plan to attend at one of these.

Assignment #2 parts A (due Fri. Oct. 4) and B (due Tues. Oct. 8) are posted. Read through them and let me know if you have any questions.

Relevant sections of text:

1.1: Java review.

1.2-1.3: Programming basics review.

1.4: Algorithm analysis.

We will cover 1.5 later when we do graph algorithms.

Now: Ch. 2: Sorting.

For recurrences/induction: Use a Math 122 text.

For the rest of this lecture, we covered the time complexity of mergesort.