You’ll need access to a laptop.
From last time

Operation Counting  Constants  Big-O (loosely)

Today

Big-O (Precisely!)

So: you want to analyze an algorithm...
Let $T(n)$ denote the running time of your algorithm on an input of size $n$. 
Let $T(n)$ denote the running time of your algorithm on an input of size $n$. 
Which function is “faster”?

“Dominated”, in mathematical parlance

Red!

No, green!
1. Analyze large N.
I. Analyze large $N$. 

![Graph showing comparison between Old laptop and New laptop.](chart.png)
1. Analyze large N.
2. Ignore constants!
So, is $f(n)$ smaller than $g(n)$?

(Dominated by)

For “big enough” $N$  

Give or take a constant
So, is $f(n)$ smaller than $g(n)$?

(Dominated by)

For “big enough” $N$

Give or take a constant

Is $f(n) < g(n) \forall n$ what we want?
So, is $f(n)$ smaller than $g(n)$?

(Dominated by)

For “big enough” $N$

$\forall n > n_0$

Is $f(n) < g(n)$ what we want?

Give or take a constant

Can be as large as you want!
So, is \( f(n) \) smaller than \( g(n) \)?

(Dominated by)

For “big enough” \( N \)  \( f(n) < c \cdot g(n) \quad \forall n > n_0 \)

Give or take a constant

If \( c \) and \( n_0 \) exist, we say \( f(n) \in O(g(n)) \)

Your choice of constant.

Can be as large as you want!
\[ f(n) < c \cdot g(n) \quad \forall n > n_0 \] means \( f(n) \in O(g(n)) \)

\[ f_1(n) = 2n \]

\[ f_2(n) = n \]

Let us know: [http://goo.gl/VHT2q](http://goo.gl/VHT2q)
\( f(n) < c \cdot g(n) \ \forall n > n_0 \) means \( f(n) \in O(g(n)) \)

\[
\begin{align*}
  f_0(n) &= n & f_1(n) &= n^2 + 10 & f_2(n) &= n!
  \\
f_3(n) &= n \log n & f_4(n) &= n^n & f_5(n) &= n \log(n^2)
  \\
f_6(n) &= \log n & f_7(n) &= \frac{1}{10^6} n^3 + 10^6 n^2 + 12n + 4
  \\
f_8(n) &= n^2 \log n & f_9(n) &= \sum_{i=1}^{n} i & f_{10}(n) &= 2^n
  \\
f_{11}(n) &= 3^n
\end{align*}
\]

Fill out our form: [http://goo.gl/9vpuE](http://goo.gl/9vpuE)