Analysis Basics: Counting Things

Consider the following algorithm:

```
a = f(0)
for (i = 1; i<=n; i++){
     a = a + f(i)
}
```

How many calls are made to the function f?

Analysis Basics: Counting Things

Consider the following algorithm:

```
a = f(0)
for (i = 1; i<=n; i++){
     for (j = 1; j<=n; j++){
          a = a + f(i) + f(j)
     }
}
```

How many calls are made to the function f?

Analysis Basics: Counting Things

Consider the following algorithm:

```
a = f(0)
for (i = 1; i<=n; i++){
     for (j = i; j<=n; j++){
          a = a + f(i)
     }
}
```

How many calls are made to the function f?

Analysis Basics: Counting Things

Consider the following algorithm:

```
a = f(0)
for (i = 1; i<=n; i++){
 for (j = 1; j<=n; j++){
     for (k = j; k<=n; k++){
          a = a + f(i) + j + k
}}}
```

How many calls are made to the function f?

Rates of Growth

Consider an algorithm A that takes $3n^3$ nanoseconds to process input of size n versus an algorithm B that takes 20,000,000n nanoseconds to process input of size n:

If n = 10 then A takes 3 microseconds and B 200 millisecs.
If n = 100 then A takes 3 ms and B takes 2 secs.
If n = 1000 then A takes 3 secs. and B takes 20 secs.

How many minutes would each algorithm take for n=10,000?

Rates of Growth

Consider an algorithm A that takes $3n^3$ nanoseconds to process input of size n versus an algorithm B that takes 20,000,000n nanoseconds to process input of size n:

If n = 10 then A takes 3 microseconds and B 200 millisecs.
If n = 100 then A takes 3 ms and B takes 2 secs.
If n = 1000 then A takes 3 secs. and B takes 20 secs.

How much time would each algorithm take for n=1,000,000?

What is Big Oh about?

Intuition: Avoid details when they don't matter and they don't matter when input size, n, is big enough.

What do these functions have in common?

n^2

$.001n^2$

$1000n^2$

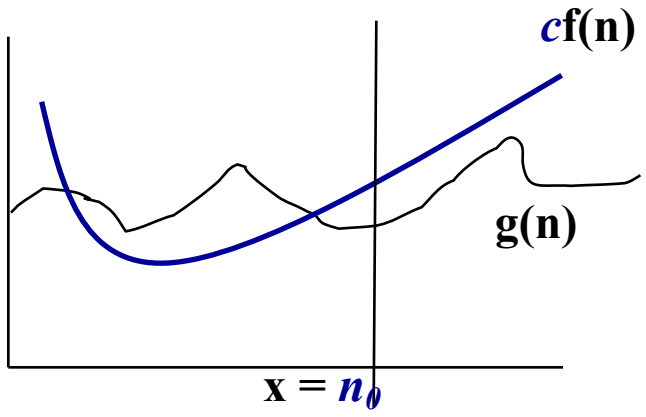
$5n^2 + 3n + 2\log n$

A more formal definition

Big Oh-notation is an upper-bound

Formally:

A function `g(n)` is `O(f(n))` if there exist constants c and n_0 such that `g(n) < cf(n)` for all `n >` n_0



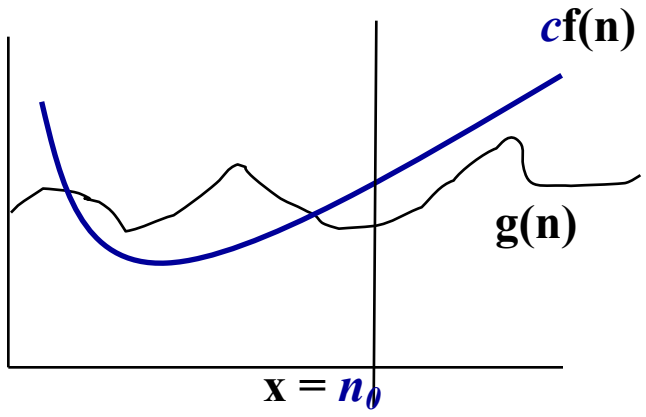
Give possible c and n_0 to show that $T(n) = n^2 + 2n + 1$ is $O(n^2)$

A more formal definition

Big Oh-notation is an upper-bound

Formally:

A function `g(n)` is `O(f(n))` if there exist constants c and n_0 such that `g(n) < cf(n)` for all `n >` n_0



Give possible c and n_0 to show that $T(n) = 2^n + 5n$ is $O(2^n)$