Eggs, Logs, Etc.

CMS 170, Spring 2019

Warm-Up

Review the formal definition of Big-O notation.

State the order of growth in Big-O notation for the following functions

- $f(n) = 3n^2 + 2n$
- $f(n) = 3n^{.01} + 2n$
- $f(n) = \log n + \sin n$

Loops

What is the Big-O complexity of the following algorithm?

```
input: array a
output: prod, a 2-D array of pairwise products of a's elements

for (i = 0; i < a.length; i++) {
    for (j = 0; j < a.length; j++) {
        prod[i][j] = a[i] * a[j]
    }
}
```

How about this one?

```
input: array a
output: nothing

for (i = 0; i < a.length; i++) {
    for (j = 0; j < 1000000; j++) {
        // Run a simulation using a[i] as a seed value
    }
}
```

Tips: identify the size of the input, then think about how many times each statement executes as a function of the input size. For example, a loop that touches each item once will execute N times. Constant factors, like adding or subtracting 1 or multiplying by a fixed value, don't change the complexity analysis, so you don't need to worry about them.

The Two-Egg Problem

You have some eggs and a 100-story building. These eggs (laid, no doubt, by a genetically-engineered monster terror chicken) have a very strong shell. So strong, in fact, that if they are dropped from any floor lower than an unknown breaking floor N they will be completely unharmed.

Your challenge: develop an algorithm that finds the breaking floor N in an efficient number of drops without breaking too many eggs.

Consider a linear search approach to finding the breaking floor. Begin at floor 1 and drop an egg. If it breaks, $N = 1$ and you're done. If it doesn't break, move up to floor 2 and drop it again. Repeat until the egg finally breaks. This algorithm has the advantage of using only one egg, but the number of drops required is linear in N.

Fewer drops are possible if you use a binary search. Drop the first egg from floor 50. If it breaks, binary search the floors from 1 to 49. If it survives, binary search the floors from 51 to 100. This will require only seven drops in the worst case, but may break more eggs.

Can you find an algorithm that is better than linear search using only two eggs? How many worst-case drops does your method require as a function of N?

Polynomials

Prove, using the definition of Big-O notation, that the polynomial

$$f(n) = a_m n^m + a_{m-1} n^{m-1} + a_{m-2} n^{m-2} + \ldots + a_2 n^2 + a_1 n$$

is $O(n^m)$.

It's Log! It's Log! It's Better Than Bad, It's Good!

Prove that $f(n) = \log n!$ is $O(n \log n)$.

Speedup Factors

Suppose you have a computer running a certain program that implements an algorithm that has complexity $O(N^2)$. In some length of time T, the computer can process a problem of size n using your code.

Suppose you purchase a new computer that is 100 times as fast as your current computer. What size problem would you expect to be able to solve in time T with the new machine without changing the code?

What if the algorithm had been $O(N)$?

What about $O(N \log N)$?