CSCA48 Winter 2015

Week 10 - Worst Case Complexity

Anna Bretscher

March 16, 2015

Last Week

- Finding an upper bound on *T(n)*, the number of steps an algorithm takes in the *worst case*.
- Gives us "*Big Oh*" or the *asymptotic upper bound*.
- Find *T(n)* for *insertion sort* by only looking at the actual code.
- We looked at *insertion sort*

TODAY

- Review “Big Oh”, O().
- Understand *insertion sort* and calculate *T(n)* again.
- Define the *lower bound*, Ω() on the worst case *T(n)*.
- Find Ω() for *insertion sort*.

Big Oh

- $f(n)$ belongs to $\mathcal{O}(g(n))$ if
- there are constants c, b
- such that $f(n) \leq c \cdot g(n)$
- whenever $n > b$ (for n big enough).
- *Note*: we only care about the term with the largest exponent. Why?

Upper Bounds

- Looking at the code we have shown *insertion sort* in the worst case takes *at most* $O(n^2)$ steps.
- In fact, our analysis was a bit *sloppy*.
- We assumed the inner loop *always* loops n times, but in fact, it doesn't.
- Does our over counting matter?
- Not this time...why?

Lower Bounds

There are two steps to proving the complexity of an algorithm.

- Find an *upper bound* $O(g(n))$ for $T(n)$.
- Find a *"bad"* input that forces the algorithm to take at least $g(n)$ steps.
- For *insertion sort*, is there an input that forces the algorithm to take the *most steps*?

INSERTION SORT

```
def insertion_sort (L):
  i = 1                                  // 1: >1      steps
  while (i < len(L)):                    // 2:  >1     steps
     t = L[i]                            // 3:  >1     steps
     j = i                               // 4:   >1    steps
     while (j > 0 and L[j-1] > t):      // 5:  >1     steps
        L[j] = L[j-1]                    // 6:  >1     steps
        j = j-1                          // 7:  >1     steps
     L[j] = t                            // 8:   >1    steps
     i = i+1                             // 9:  >1     steps
```

Let's look at what it's really doing!

OMEGA: $\Omega()$

We say that $T(n)$ is bounded from below or

- $T(n)$ belongs to $\Omega(g(n))$ if
- there exists constant $d \in \mathbb{R}^+$, and $b \in \mathbb{N}$ such that
- $T(n) \geq d \cdot g(n)$ whenever $n > b$.

PRACTICE

Prove each of the following:

$$n^3 - 4n^2 + 5 \in \mathcal{O}(n^3)$$

$$n^2 + n \log n \in \Omega(n^2)$$

$\frac{n^3 - n}{n^2} \in \mathcal{O}(n)$ and $\Omega(n)$.

$$n^3 - n^2 + 5 \in \Omega(n^3)$$