TCSS 343 - Week 1

Jake McKenzie

August 3, 2018

Asymptotics

"You're always you, and that don't change, and you're always changing, and there's nothing you can do about it." Neil Gaiman

"Perhaps thinking should be measured not by what you do but by how you do it." Richard Hamming

"Can we do better?" Tim Roughgarden

Exact answers are nice when you can find them. Often times we can't or don't care to find them. As computer scientists we care about the behaivor of the algorithms we employ. If we run into a recurrence relation or summation that expressed loosly the behaivor of these algorithms we'd like to know something about their runtime.

Asymptotic Notation in Seven Words
suppress constant factors and lower-order terms

Constant factors are things that will be language and system dependent while lower order terms are irrelevant for large inputs.

Now consider these definitions:
Big-O Notation
If $\lim\limits_{n\to\infty} \frac{f(n)}{g(n)} \to 0$ then $f(n) \in O(g(n))$

This is another way of saying that $f(n) \leqslant c \cdot g(n)$ for some position c.
Big-Ω Notation
If $\lim\limits_{n\to\infty} \frac{f(n)}{g(n)} \to \infty$ then $f(n) \in \Omega(g(n))$

This is another way of saying that $f(n) \geqslant c \cdot g(n)$ for some position c.
Big-Θ Notation
If $\lim\limits_{n\to\infty} \frac{f(n)}{g(n)} \to k$ where k is a positive finite number then $f(n) \in \Theta(g(n))$

This is another way of saying that $f(n) \in O(g(n))$ and $f(n) \in \Theta(g(n))^{\cdot}$
Now while we're in this course, we will typically need to show that these relationships are true when prompted, but it's always good to know that the following is true where $0 < \epsilon < 1 < c$.

$$1 < \log\log n < \log n < n^{\epsilon} < n^{c} < n^{logn} < c^{n} < n^{n} < c^{c^{n}}$$

This asymptotic pecking order above is from Don Knuth's Concrete Mathematics.

Now consider the following truths I found in Tim Roughgarden's Algorithms Illuminated:

$$\max\{f(n), g(n)\} \leqslant f(n) + g(n) \wedge 2 \cdot \max\{f(n), g(n)\} \geqslant f(n) + g(n)$$

(reminder that $\wedge$ is the logical "and" symbol.)

1. Prove the following by using the definitions that I've given prior and the information immediately above to show that $\max\{f(n), g(n)\} \in \Theta(f(n)+g(n))$

2. Arrange the following functions in order of increasing growth rate, with $g(n)$ following $f(n)$ in your list if and only if $f(n) \in O(g(n))$. That means using the limit definitions I gave you or by induction. You cannot simply use Don's asymptotic pecking order I stated but I strongly suggest you use the asymptotic pecking order as a guide to the neighborhood of a correct answer.

a) $\sum_{i=0}^{n} 2^i$

b) n^2

c) $n^{0.9999999} \log n$

d) 1.00001^n

e) $\log 2^{\frac{n}{2}}$

f) $1000000n$

3. Arrange the following functions in order of increasing growth rate, with $g(n)$ following $f(n)$ in your list if and only if $f(n) \in O(g(n))$. That means using the limit definitions I gave you or by induction. You cannot simply use Don's asymptotic pecking order I stated but I strongly suggest you use the asymptotic pecking order as a guide to the neighborhood of a correct answer.

a) $n^{\frac{5}{3}}$

b) $\sum_{i=0}^{n}(i+1)$

c) $n\sqrt{n}$

d) $2^{\frac{n}{2}}$

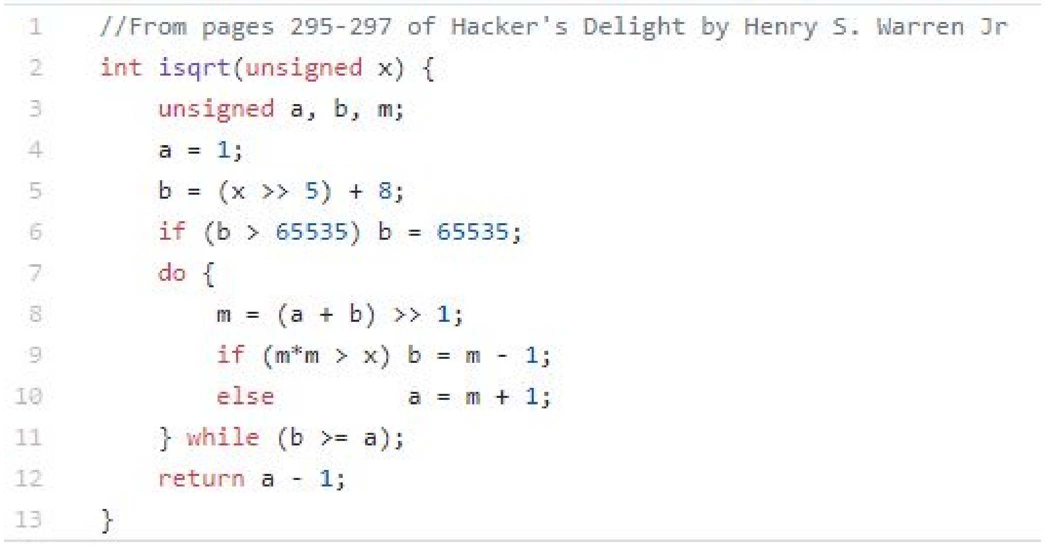
e) $\log\log n$

f) $n^{1.5}$

4. Using what you know prove or disprove that $f(n)+O(f(n)) \in \Theta(f(n))$.

5. Using what you know prove or disprove that $f(n) \in O(f(n)^2)$.

6. Find the worst case runtime of the code below. What is the code doing? (hint: follow what the code is doing by letting $x = 100$ and returning the result. You may use a calculator.)



```c
//From pages 295-297 of Hacker's Delight by Henry S. Warren Jr
int isqrt(unsigned x) {
    unsigned a, b, m;
    a = 1;
    b = (x >> 5) + 8;
    if (b > 65535) b = 65535;
    do {
        m = (a + b) >> 1;
        if (m*m > x) b = m - 1;
        else         a = m + 1;
    } while (b >= a);
    return a - 1;
}
```

Interview Question: Binary Search

7. Given an already sorted ArrayList of integers write a function in Java that finds the value t from n keys via binary search. Return -1 if t is not in the list otherwise return the index.