Design and Analysis of Algorithms (CSCI 323)

Lecture #2

July 3rd, 2018

Notes by: Mark A Avila

Announcements

- HW #1 is due on Monday July 9th at the beginning of the class.
- HW #2 will be posted the day after Monday. (assuming…)
- Exam 1 will be held on Thursday July 12th, 2018 at 8:00AM – 9:34AM EST. Old exams will be sent via Google Drive which I will email this week.
- No school on Wednesday July 4th, 2018. On Thursday, we will continue our new topic Data Structures and hopefully Binary Trees and Sorting
- Have a good Fourth of July and see you on Thursday!

Topic 1: Time Complexity from Yesterday's class (7/2/2018)

Yesterday's class we talked about the best, average and worst case including their asymptotic notations.

Best case: is the function defined by the minimum number of steps taken on any instance of size *n*.

Average case: is the function defined by the average number of steps taken on any instance of size *n*.

Worst case: what is the worst case performance possible for a given size input n. **WHY?** … It's easier to analyze and it gives us a sense of scenarios.

Between these 3 we only use worst case and average case performance since best case makes it difficult to analyze and has a faster performance than the other 2.

From geeksforgeeks

Worst Case Analysis (Usually Done)
In the worst case analysis, we calculate upper bound on running time of an algorithm. We must know the case that causes maximum number of operations to be executed.

Average Case Analysis (Sometimes done)
In average case analysis, we take all possible inputs and calculate computing time for all of the inputs. Sum all the calculated values and divide the sum by total number of inputs. We must know (or predict) distribution of cases

Best Case Analysis (Bogus)
In the best case analysis, we calculate lower bound on running time of an algorithm. We must know the case that causes minimum number of operations to be executed

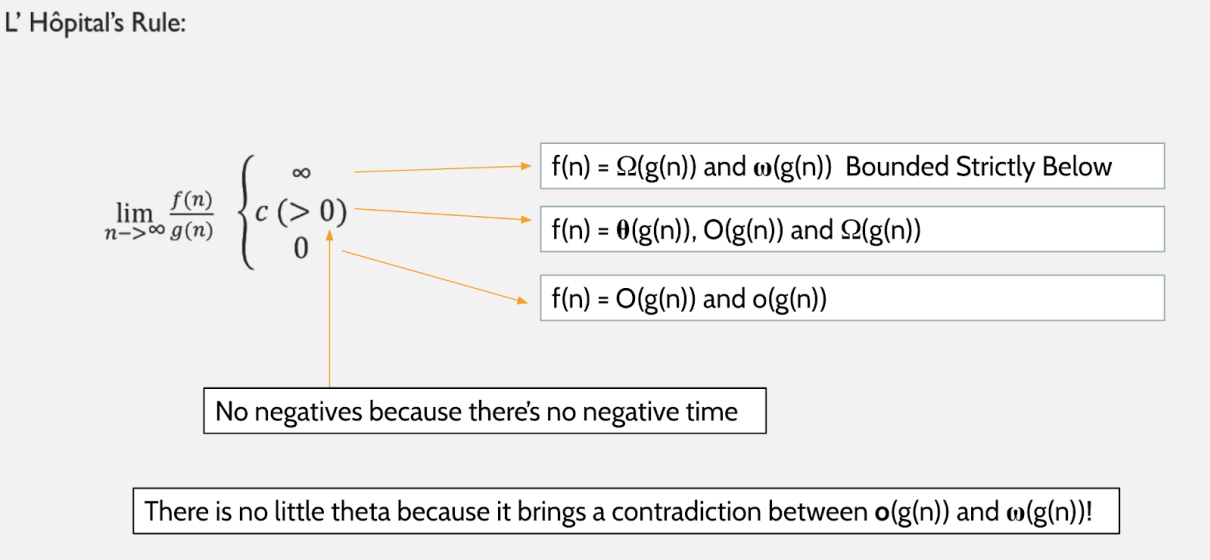
Topic 2: Asymptotic Notation for little-oh and omega

θ Big-Theta	f(n) is θ(g(n)) iff $\exists n_0 \in \mathbb{N}$ and $c \in R^+$, such that $\forall n \geq n_0$, $c_1 . g(n) \leq f(n) \leq c_2 . g(n)$
O Big -Oh	f(n) is O(g(n)) iff $\exists n_0 \in \mathbb{N}$ and $c \in R^+$, such that $\forall n \geq n_0$, $f(n) \leq c . g(n)$
o Little Oh	f(n) is o(g(n)) iff $\nexists n_0 \in \mathbb{N}$ and $c \in R^+$, such that $\forall n \geq n_0$, $f(n) \geq c . g(n)$
Ω Big- Omega	f(n) is Ω(g(n)) iff $\exists n_0 \in \mathbb{N}$ and $c \in R^+$, such that $\forall n \geq n_0$, $f(n) \geq c . g(n)$
ω Little-Omega	f(n) is ω(g(n)) iff iff $\exists n_0 \in \mathbb{N}$ and $c \in R^+$, such that $\forall n \geq n_0$, $f(n) \leq c . g(n)$
? Little- Theta	Look at the diagram below. Also, it's confusing and nonsense!

Example: f(n) = $2n^2$+5n+100 and g(n) = n^2+1000n

We can look at the limit of f(n) and g(n) using L'Hôpital's Rule.

Topic 3: L'Hôpital's Rule



When comparing the limit of the function and getting the result, ω(g(n)), O(g(n)), o(g(n)) are strong arguments since they are strictly below!

- ∞ and –c does not exist in the Computer Science Department, ONLY IN MATH DEPT!!

From Topic 2, Example:

f(n) = $2n^2$+5n+100 and g(n) = n^2+1000n.

Finding the limit of both functions we can just derive then until we get a result.

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{f'(n)}{g'(n)} = \frac{4n+5}{2n+100} =$$

$$f''(n) \quad 4$$

And now since 2 is a constant we can say that f(n) = O(g(n)), Ω(g(n), and θ(g(n))

Example 2: f(n) $= n^2$ and g(n) = n log n

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{f\prime(n)}{g\prime(n)} = \frac{1}{1/n} =$$

$n \to \infty$

$n^2 = \Omega(n \log n)$

n log n = ω(n log n) ← Strong argument since it's strictly below!

Example 3: f(n) = n^3 and g(n) = 2^n

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{f\prime(n)}{g\prime(n)} =$$

n^3

It's 0 since the denominator is getting smaller and it will remain exponential!

$n^3 = O(2^n)$

$2^n = o(2^n)$ ← Stronger argument… √

Topic #4: Ranking Functions

Ranking functions is putting them in order in increasing or decreasing order in polynomial.

Example: 5, … , n!

log n and log log n.

$$\log_{10} 10^{100} = 100$$

$$\log_{10} \log_{10} 100 = \log_{10} 100 = 2$$

But sadly this won't work in the case for whole numbers since n can exceed as much as it can.

Example 2: n^2+1000000 and 2^n

2^n will be bigger than n^2+1000000 no matter how bigger it gets, since 2^n is exponential

$n^2+1000000^{1000000}$ and 2^n ← still bigger and can exceed!

Review for Logarithms

log (a x b) = log a+log b

log n! = logn + log(n-1)+ … +log1 < n logn

Topic #5: Recurrence Relations

f(0) = 0

f(n) = f(n-1) +1

If we want an answer; we want a closed form solution like the following…

f(n) = n-1

f(n) = n^2+5

f(n) = 2^n

Example: Linear Search

Finding the min/max

f(n) = f(n-1) +1

= [f(n-2) +1] +1

= [f(n-3) +1+1] +1

...

= f(0) + n

Assume that f(n)= ∞

0: f(0) = 0

n >0: n =n (n-1) +1

We expand it out using the expansion method!

Topic 6: Telescoping and the Characteristic equation

n: f(n) - f(n-1) = 1

n-1: f(n-1) - f(n-2) = 1

n-2: f(n-2) - f(n-3) = 1

…

1: f(1) - f(0) = 1

RHS = n

LHS = f(n) - f(0)

f(n) =n; The f(0) is cancelled out since f(0) = 0, so the answer is just n. The reason why it's called telescoping since any like terms like the example above can be cancelled out and be reduced to a much simplified function

Example: Fibonacci Numbers

f(0) = 0

f(1) = 1

f(n) = f(n-1) + f(n-2) → f(n) – f(n-1) – f(n-2) = 0

Characteristic equation: $F_n – F_{n-1} – F_{n-2} = 0$ OR $X^n – X^{n-1} – X^{n-2} = 0$

$X^{n-2}(x^2 - x - 1) = 0$

$x^2 - x - 1 = 0$ ← We use quadratic formula

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

The solution to this problem is going to be a linear combination to the characteristic equation (power of the 2 solutions).

$$x = \frac{-(-1) \pm \sqrt{(-1)^2 - 4(1)(-1)}}{2(1)}$$

$$x = \frac{1 \pm \sqrt{5}}{2}$$

$$\alpha\left(\frac{1+\sqrt{5}}{2}\right)^n + \beta\left(\frac{1+\sqrt{5}}{2}\right)^n$$

$$\mathrm{f}(0) = 0:\ \alpha\left(\frac{1+\sqrt{5}}{2}\right)^0 + \beta\left(\frac{1+\sqrt{5}}{2}\right)^0 = 0$$

$$\alpha + \beta = 0 \ \rightarrow\ \alpha = -\beta$$

$$\mathrm{f}(1) = 1:\ -\beta\left(\frac{1+\sqrt{5}}{2}\right)^1 + \beta\left(\frac{1+\sqrt{5}}{2}\right)^1 = 0$$

$$\beta\left[\frac{-1+\sqrt{5}}{2} + \frac{1-\sqrt{5}}{2}\right] = 1$$

$$\beta\left(\frac{-2\sqrt{5}}{2}\right) = 1$$

$$\beta = \frac{-1}{\sqrt{5}} = \frac{-\sqrt{5}}{5}$$

$$\alpha = \frac{\sqrt{5}}{5}$$

$$\mathrm{f}(2) = \mathrm{f}(1) = \mathrm{f}(0)$$

$$= \frac{\sqrt{5}}{5}\left(\frac{1+\sqrt{5}}{2}\right)^2 - \frac{\sqrt{5}}{5}\left(\frac{1+\sqrt{5}}{2}\right)^2$$

$$= \frac{\sqrt{5}}{5}\left(\frac{1+2\sqrt{5}+5}{4} - \frac{1-2\sqrt{5}+5}{4}\right)$$

$$= \frac{\sqrt{5}}{5} * \frac{4\sqrt{5}}{4} = \frac{5}{5} = 1$$

Since we found out that the Fibonacci number is 1, we can try to find out what's the 1000th Fibonacci number using logarithms and basic math…

1.6^{1000} ← It won't work because we don't know what it looks like.

$1.6^{x} \rightarrow^{10}$

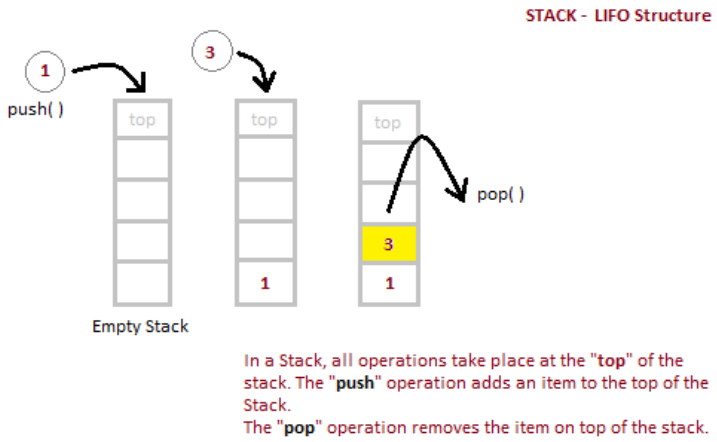
$$\log_{1.6} 10 = \frac{\ln 10}{\ln 1.6} = 4.89 \approx 5$$

$$1.6^{1000} \approx 10^{\frac{1000}{5}} = 10^{200}$$

Topic #7: Data Structures (Stacks, Queue, List)

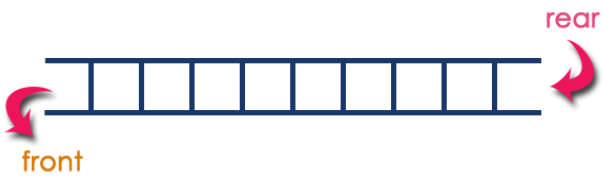
Stack: is a list with the restriction that insertion and deletions can be performed in only one position, namely, the end of the list, called the top. Also known as LIFO (Last In, First Out)

Push: Adds an item in the stack. If the stack is full, then it is said to be an Overflow condition.
Pop: Removes an item from the stack. The items are popped in the reversed order in which they are pushed. If the stack is empty, then it is said to be an Underflow condition.
Top: Returns top element of stack.
isEmpty: Returns true if stack is empty, else false.



Queue: is open at both its ends. One end is always used to insert **data**(enqueue) and the other is used to remove **data**(dequeue).

Enqueue: Adds an item to the queue. If the queue is full, then it is said to be an Overflow condition.
Dequeue: Removes an item from the queue. The items are popped in the same order in which they are pushed. If the queue is empty, then it is said to be an Underflow condition.
Front: Get the front item from queue.
Rear: Get the last item from queue.



PSA: Stack and Queue have constant time operations θ(1).

List: represents a countable number of ordered values, where the same value may occur more than once.

i^{th} element, θ(i), since you are looking for that value within the list.

find(x), delete(x); while x is the key of the element in the list

Worst Case: θ(n)
Avg Case: θ(n/2) = θ(n)
} linear
Best case: θ(n) → constant

References/Links

https://www.geeksforgeeks.org/analysis-of-algorithms-set-2-asymptotic-analysis/

Asymptotic Notation Table by Mark Avila from last semester CS 320 Final Review Doc 5/3/2018

L'Hôpital's Rule image: Last semester's lecture notes for CS320 by Jeffrey Meza (5/1/2018)

https://www.geeksforgeeks.org/stack-data-structure-introduction-program/

https://en.wikipedia.org/wiki/List_(abstract_data_type)

https://www.studytonight.com/data-structures/queue-data-structure