Lecture 2

Asymptotic Analysis

Teera Siriteerakul

1

Comparing Growth Rate

$f_0(n) = 2n + 5 \qquad f_1(n) = 4\sqrt{n} + 6 \qquad f_2(n) = 7\frac{\sqrt{n}}{6} + 15$

- These are functions from size of input to number of computing operation of isPrime0, isPrime1, and isPrime2
- We can say that the growth rates of $f_1(n)$ and $f_2(n)$ are mathematically similar taking a limit as n approach infinity

$$\lim_{n\to\infty} \frac{f_1(n)}{f_2(n)} = \lim_{n\to\infty} \frac{4\sqrt{n} + 6}{7\frac{\sqrt{n}}{6} + 15} = \frac{24}{7}$$

- These means that, $f_1(n)$ and $f_2(n)$ grows together to infinity.

2

Comparing Growth Rate

- How about $f_0(n)$ comparing to $f_1(n)$?

$$\lim_{n\to\infty} \frac{f_0(n)}{f_1(n)} = \lim_{n\to\infty} \frac{2n + 5}{4\sqrt{n} + 6} = \infty$$

- This means that $f_0(n)$ is bigger than $f_1(n)$ if n is large enough.

- Now, let compare $f_2(n)$ to $f_0(n)$:

$$\lim_{n\to\infty} \frac{f_2(n)}{f_0(n)} = \lim_{n\to\infty} \frac{7\frac{\sqrt{n}}{6} + 15}{2n + 5} = 0$$

- This means $f_2(n)$ is small comparing to $f_0(n)$ if n is large enough.

3

Asymptotic Analysis

- We are to compare growth rates of functions, as their input increase (up to infinity), to another function.
- There are three asymptotic analysis commonly used in Computer Science:
  - $f(n) \in O(g(n))$ read **Big-O** of $f(n)$ is $g(n)$, means that g(n) is an upper bound of f(n) or **f(n) grows asymptotically no faster than g(n)**.
  - $f(n) \in \Omega(g(n))$ read **Big-Omega** of $f(n)$ is $g(n)$, means that g(n) is a lower bound of $f(n)$ **f(n) grows asymptotically faster than g(n)**.
  - $f(n) \in \theta(g(n))$ read **Big-Theta** of $f(n)$ is $g(n)$, means that $f(n)$ and g(n) are growing at that same rate or **f(n) grows asymptotically as faster as g(n)**.

  Note that $f(n)$ and $g(n)$ are defined on unbound subset of positive real number and $g(n)$ is strictly positive for all large enough n.

4

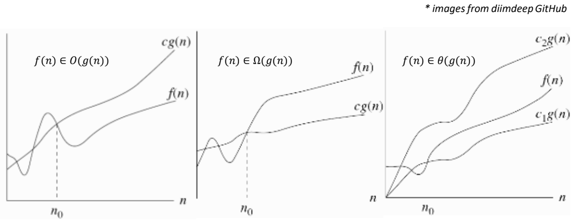
Formal Definition

Notation	Formal Definition	Limit Definition
$f(n) = O(g(n))$	$\exists k > 0 \exists n_0 \forall n > n_0 : \lvert f(n) \rvert \le k \cdot g(n)$	$\limsup_{n\to\infty} \frac{\lvert f(n) \rvert}{g(n)} < \infty$
$f(n) = \Theta(g(n))$	$\exists k_1 > 0 \exists k_2 > 0 \exists n_0 \forall n > n_0$: $k_1 \cdot g(n) \le f(n) \le k_2 \cdot g(n)$	$f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$ (Knuth version)
$f(n) = \Omega(g(n))$	$\exists k > 0 \exists n_0 \forall n > n_0 : f(n) \ge k \cdot g(n)$	$\liminf_{n\to\infty} \frac{f(n)}{g(n)} > 0$

5

Simple Illustrations

** images from diimdeep GitHub*



6

Simple g(n)

- Rather than comparing to each other, we will compare each function with a simple function, for example

$$f_0(n) = 2n + 5 \in O(n)$$
$$f_1(n) = 4\sqrt{n} + 6 \in O(\sqrt{n})$$
$$f_2(n) = 7\frac{\sqrt{n}}{6} + 13 \in O(\sqrt{n})$$

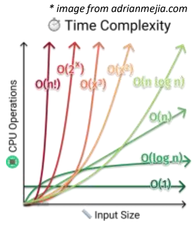
- So, we are saying that isPrime0 has $O(n)$ while isPrime1 and isPrime2 has $O(\sqrt{n})$.
  - Thus, we regards isPrime1 and isPrime2 as equal and superior to isPrime0

7

Typical Big-O

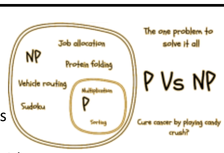
Big-O	Name
$O(1)$	Constant
$O(\log n)$	Logarithmic
$O(n)$	Linear
$O(n \log n)$	Just $n \log n$
$O(n^2)$	Quadratic
$O(n^k)$	Polynomial
$O(c^n)$	Exponential
$O(n!)$	Factorial
$O(n^n)$	Very bad!

** image from adrianmejia.com*

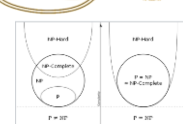


8

P vs NP problems



- If a problem is solvable with polynomial time, $O(n^k)$, or less, we call them a P class problem.
- NP-complete problem means a problem with no polynomial time solution.
- Although widely accept, there is **no definite proof** that $\mathrm{P} \neq NP$



NP is short for "nondeterministic polynomial time"

9

Note on Big-O

- In Computer Science, we often use only Big-O for asymptotic analysis.
- However, people are expecting the smallest Big-O.
  - For example, if you say $f_1(n) = 4\sqrt{n} + 6 \in O(n)$ **is technically correct**, but people will be expecting $O(\sqrt{n})$ and **tends to think you are wrong**.
  - Thus, to be the safe side, although we say Big-O, we should use Big-Theta whenever possible.

10

Summary

- Asymptotic analysis measures computational complexity by comparing them asymptotically to well known functions.
- There are several well known functions that we are typically compare our function to.
- Note that, although we say Big-O, we are expecting Big-Theta.

Data Structures & Algorithms

11