CSI33 Data Structures

Sharon Persinger

Fall 2019

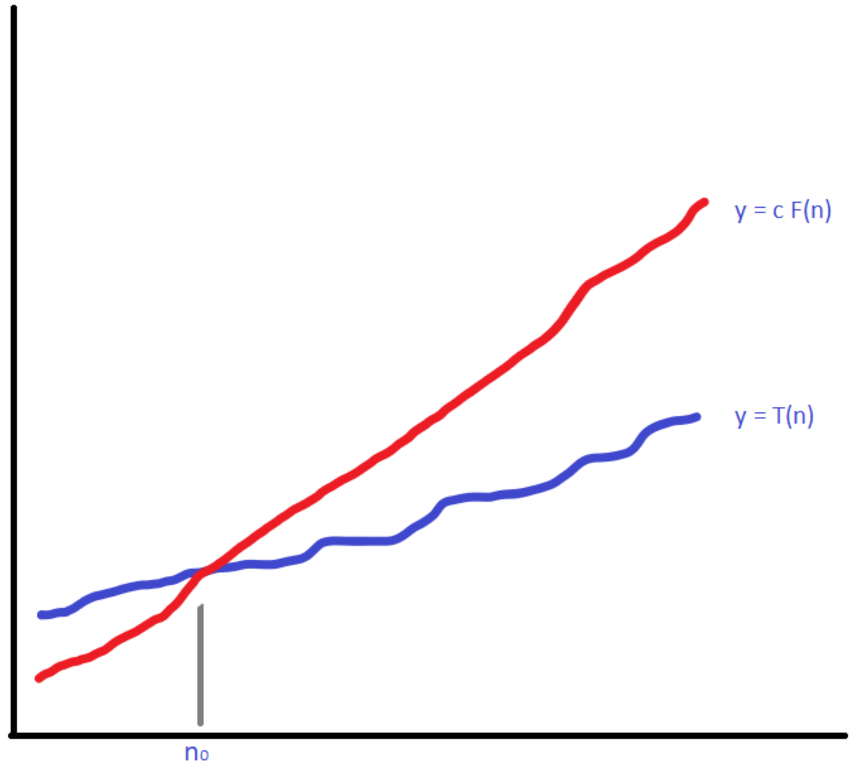
September 4, 2019 Day 2

Analysis of Algorithms - terminology

- The size of a problem is n - a natural number. Usually n is the number of pieces of data for the problem.
- There is some algorithm that solves the problem.
- $T(n)$ is a function of n defined as maximum number of steps needed for this algorithm to solve the problem for any set of data with n elements – the worst case situation.
- We want to find bounds on $T(n)$ for large values of n.

$O(f(n))$ - Big O

- An algorithm is called $O(f(n))$ - Big O of $f(n)$ if its number of steps function $T(n)$ is bounded above by some constant multiple of f(n) for large enough values of n. Formally –
- If there is some constant c and some natural number n_0 such that for all $n > n_0$, $T(n) < cf(n)$

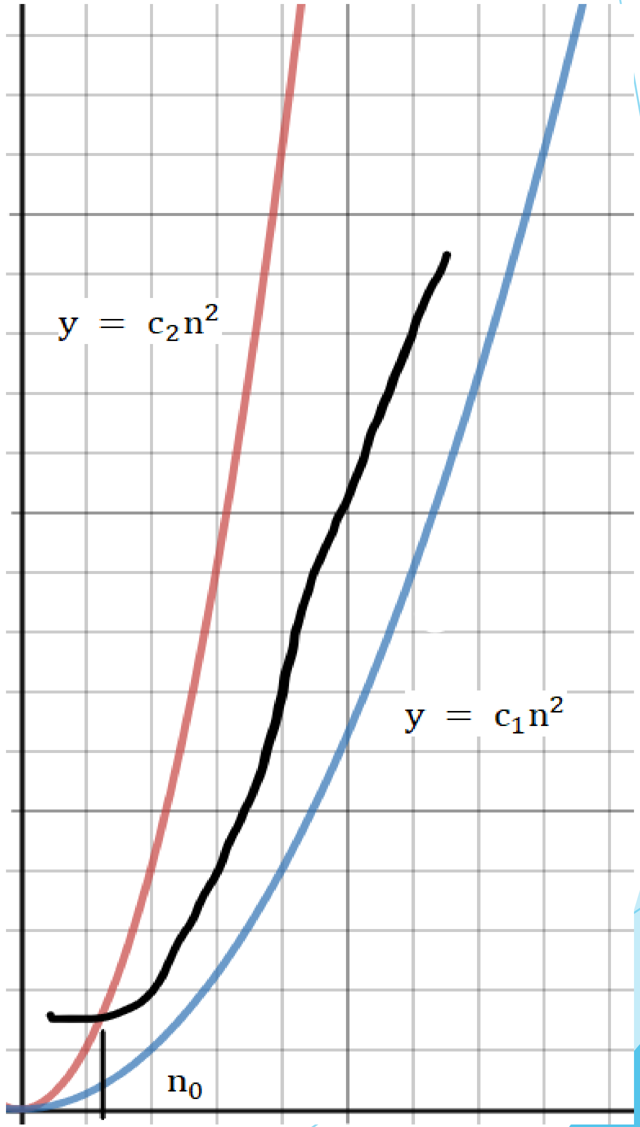


Big O bounds aren't tight

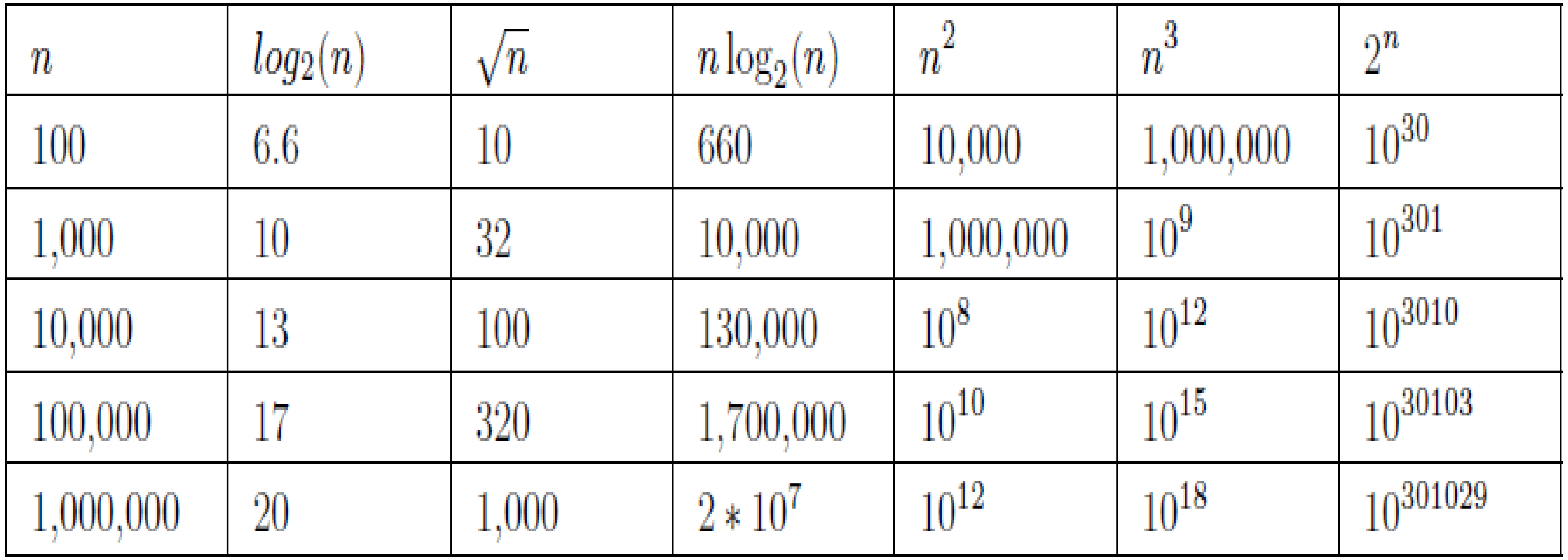
- Any algorithm that is $O(n)$ is also $O(n^2)$.
- Explain. A Big O bound just gives an upper bound.

Big Theta - $\Theta(f(n))$ bounds

- An algorithm is called $\Theta(f(n))$ - Big *Theta* of $f(n)$ if its number of steps function $T(n)$ is bounded above and below by some constant multiples of f(n) for large enough values of n. Formally -
- If there are some constants c_1 and c_2 and some natural number n_0 such that for all $n > n_0$, $c_1 f(n) < T(n) < c_2 f(n)$



Some familiar f(n)



n	$log_2(n)$	$\sqrt{n}$	$n\log_2(n)$	n^2	n^3	2^n
100	6.6	10	660	10,000	1,000,000	10^{30}
1,000	10	32	10,000	1,000,000	10^9	10^{301}
10,000	13	100	130,000	10^8	10^{12}	10^{3010}
100,000	17	320	1,700,000	10^{10}	10^{15}	10^{30103}
1,000,000	20	1,000	$2 * 10^7$	10^{12}	10^{18}	10^{301029}

Figure 1.4: Approximate growth rate of common functions

Estimating time for solving problems

- If a $\Theta(n^2)$ algorithm takes 4 seconds to execute on an input of 1,000,000 data points, about how long should it take on an input of 3,000,000 data points?
- If a $\Theta(\mathrm{n})$ algorithm takes 4 seconds to execute on an input of 1,000,000 data points, about how long should it take on an input of 3,000,000 data points?
- If a $\Theta(log_2(n)$ algorithm takes 4 seconds to execute on an input of 1,000,000 data points, about how long should it take on an input of 2,000,000 data points?
- How does all this change if we know only Big O bounds?

Is the processor fast enough? What can Big Θ analysis tell us?

- Current processors perform 2 to 5 billion operations per second – 2GHz to 5GHz.
- About how many seconds does it take to carry out a $\Theta(n^2)$ algorithm on a million data points with a 2GHz processor?
- About how many seconds does it take to carry out a $\Theta(2^n)$ algorithm on a hundred data points?

Program code and time analysis

- examples.py

Abstract Data Types

- Abstract Data type (ADT) is a type (or class) for objects whose behavior is defined by a set of values and a set of operations.
- The definition of an ADT tells what operations are to be performed on the data but not how these operations will be implemented. It does not specify how data will be organized in memory and what algorithms will be used for implementing the operations.
- Providing only the essentials and hiding the details is known as abstraction.
- Python list – a sequence of data elements with these operations list.append(x), list.extend(iterable), list.insert(i, x), list.remove(x), list.pop([i]), list.clear(), list.index(x[, start[, end]]), list.count(x),list.sort(key=None, reverse=False), list.reverse(), list.copy()

ADT for a Playing Card – object oriented version

- A playing card has a rank: Ace, Two, Three, Four, …, Ten, Jack, Queen, King; and a Suit: Clubs, Diamonds, Hearts, Spades
- What operations do we need?
- Constructor, accessors – get functions for rank and suit, functions to present rank and suit as strings, string representation function
- How to represent the data?
- Specification in module cardspec.py

One implementation of Card ADT

- Card.py