CSSE 220 Day 27

Analysis of Algorithms intro

Program “goodness”

- What is “goodness”?
- How to measure efficiency?
  - Profiling, Big-Oh
- Big-Oh:
  - Motivation
  - Informal examples
  - Informal definition
  - Formal definition
    - Mathematical
  - Application: examples
  - Best, worst, average case

What makes a program “good”

- Correct – meets specifications
- Easy to understand
- Easy to modify
- Easy to write
- Runs fast
- Uses reasonable set of resources
  - Time
  - Space (main memory)
  - Hard-drive space
  - Peripherals
  - …

Measuring program effciency

- What kinds of things should we measure?
  - CPU time
  - memory used
  - disk transfers
  - network bandwidth
- Mostly in this course, we focus on the first two, and especially on CPU time
- One way to measure CPU time: ***profiling***
  - Run the program in a variety of situations / inputs
  - Call **`System.currentTimeMillis()`**
- What are the problems with profiling?

Big-Oh motivation: why profiling is not enough

- Results from profiling depend on:
  - Power of machine you use
    - CPU, RAM, etc
  - State of machine you use
    - What else is running? How much RAM is available? …
  - What inputs do you choose to run?
    - Size of input
    - Specific input

Big-Oh motivation: what it provides

- Big-Oh is a mathematical definition that allows us to:
  - Determine how fast a program is (in big-Oh terms)
  - Share results with others in terms that are universally understood
- Features of big-Oh
  - Allows paper-and-pencil analysis
  - Is much easier / faster than profiling
  - Is a function of the *size of the input*
  - Focuses our attention on *big* inputs
  - Is machine independent

Familiar example:
Linear search of a sorted array of Comparable items

```
for (int i=0; i < a.length; i++) {
  if ( a[i].compareTo(soughtItem) > 0 )
      return NOT_FOUND; // Explain why this is NOT cohesive.
                        // NOT_FOUND must be …?
  else if ( a[i].compareTo(soughtItem) == 0 )
      return i;
}
return NOT_FOUND;
```

- What should we count?
- Best case, worst case, average case?

Q5

Another algorithm analysis example

Does the following method actually create and return a copy of the string `s`?

What can we say about the running time of the method? (where **N** is the length of the string **`s`**)

What should we count?

```
public static String stringCopy(String s) {
  String result = "";
  for (int i=0; i<s.length(); i++)
     result += s.charAt(i);
  return result;
}
```

Don’t be too quick to make assumptions when analyzing an algorithm!

How can we do the copy more efficiently?

Q6

Interlude

- Always code as if the guy who ends up maintaining your code will be a violent psychopath who knows where you live.
  --Martin Golding

Figure 5.1
Running times for small inputs

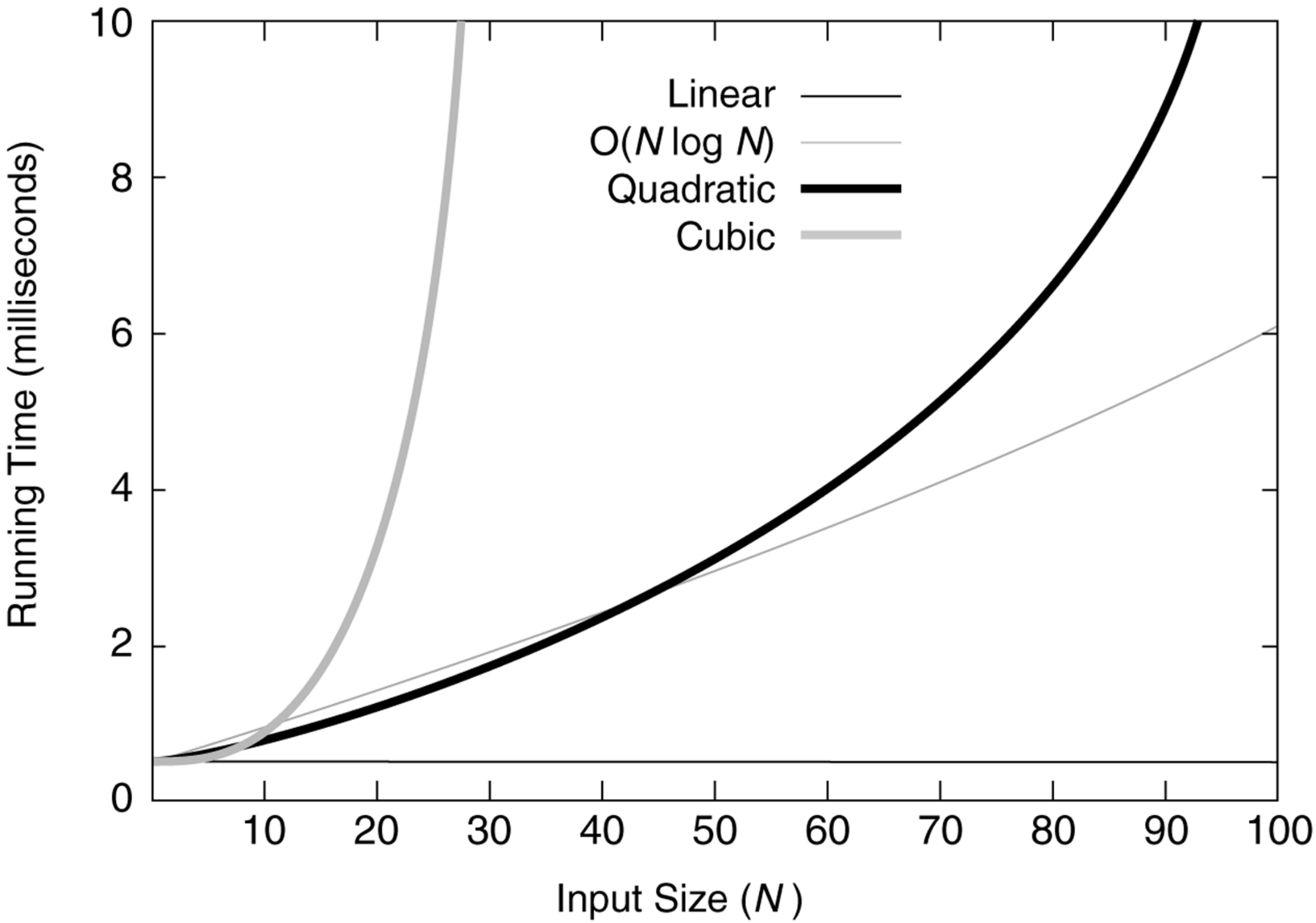


Figure 5.2
Running times for moderate inputs

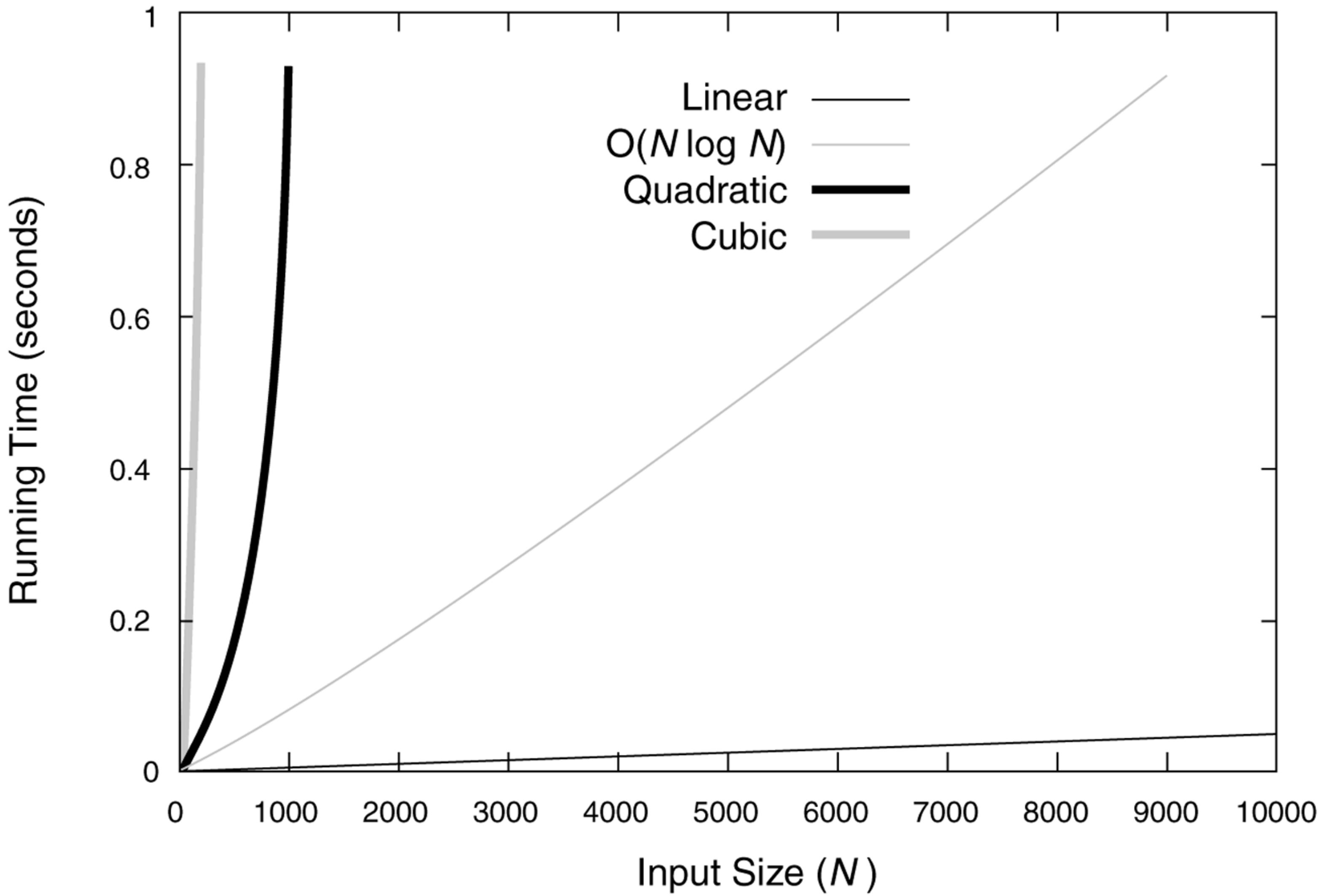


Figure 5.3

Functions in order of increasing growth rate

FUNCTION	NAME
c	Constant
$\log N$	Logarithmic
$\log^2 N$	Log-squared
N	Linear
$N \log N$	N log N
N^2	Quadratic
N^3	Cubic
2^N	Exponential

a.k.a "log linear"

Asymptotic analysis

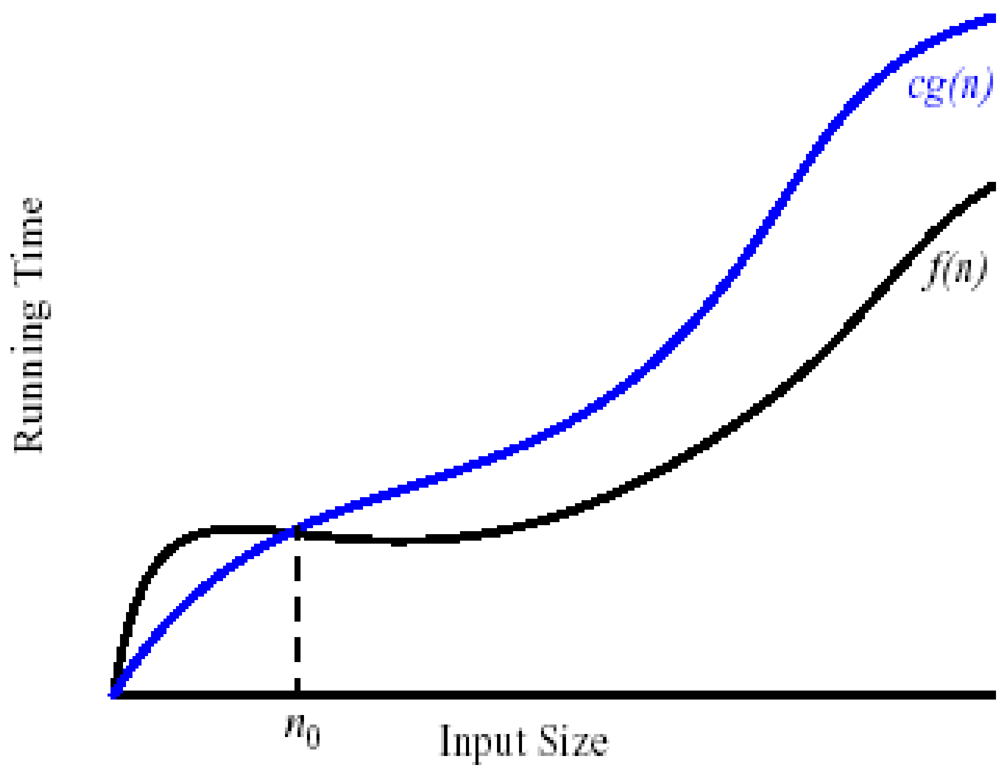
- We only really care what happens when N (the size of a problem) gets large
- Is the function basically linear, quadratic, etc. ?
- For example, when **n** is large, the difference between $\mathbf{n}^2$ and $\mathbf{n}^2 - 3$ is negligible

Informal definition of big-Oh
As applied to run-time analysis

- Run-time of the algorithm of interest on a worst-case input of size n is:
  - at most a constant times *blah*, for large n
- Example: run-time of the linear search algorithm on a worst-case input of size n is:
  - O(n)
- Alternatives to:
  - Run-time: space required, …
  - Algorithm of interest: Problem of interest
  - Worst-case input: Average-case, best-case
  - At most: At least => Ω and "exactly" (i.e. one constant for at least and another for at most) => Θ

- The "Big-Oh" Notation
  - given functions f(n) and g(n), we say that f(n) is O(g(n)) if and only if there exists a c such that f(n) $\leq$ c g(n) for $n \geq n_0$ for all n >= n0
  - c and n_0 are constants, f(n) and g(n) are functions over non-negative integers

In this course, we won't be so formal . We'll just say that f(N) is O(g(N) means that f(n) is eventually smaller than a constant times g(n).



Q7

- Simple Rule: Drop lower order terms and constant factors.
  - $7n - 3$ is $\boldsymbol{O}(n)$
  - $8n^2\log n + 5n^2 + n$ is $\boldsymbol{O}(n^2\log n)$

- Special classes of algorithms:
  - logarithmic: $\boldsymbol{O}(\log n)$
  - linear $\boldsymbol{O}(n)$
  - quadratic $\boldsymbol{O}(n^2)$
  - polynomial $\boldsymbol{O}(n^k)$, $k \geq 1$
  - exponential $\boldsymbol{O}(a^n)$, $n > 1$

- "Relatives" of the Big-Oh
  - $\Omega(f(n))$: Big Omega
  - $\Theta(f(n))$: Big Theta

Recap: O, Ω, Θ

- **f(N) is O(g(N))** if there is a constant c such that for sufficiently large N, $f(N) \leq cg(N)$
  - Informally, as N gets large the growth rate of **f** is bounded above by the growth rate of **g**
- **f(N) is Ω(g(N))** if there is a constant c such that for sufficiently large N, $f(N) \geq cg(N)$
  - Informally, as N gets large the growth rate of **f** is bounded below by the growth rate of **g**
- **f(N) is Θ(g(N))** if f(N) is O(g(n)) **and** f(N) is **Ω**(g(N))
  - Informally, as N gets large the growth rate of **f** is the same as the growth rate of **g**

Limits and asymptotics

- consider the limit $$\lim_{n \to \infty} \frac{f(n)}{g(n)}$$
- What does it say about asymptotics if this limit is zero, nonzero, infinite?
- We could say that knowing the limit is a sufficient but not necessary condition for recognizing big-oh relationships.
- It will be all we need for all examples in this course.

Q8

Apply this limit property to the following pairs of functions

1. N and N^2
2. $N^2 + 3N + 2$ and N^2
3. $N + \sin(N)$ and N
4. $\log N$ and N
5. $N \log N$ and N^2
6. N^a and N^n
7. a^N and b^N $(a < b)$
8. $\log_a N$ and $\log_b N$ $(a < b)$
9. $N!$ and N^N

Q9

Big-Oh Style

- **Give tightest bound you can**
  - Saying that $3N+2$ is $O(N^3)$ is true, but not as useful as saying it's $O(N)$ [What about **$\Theta(N^3)$** ?]
- **Simplify:**
  - You *could* say:
  - $3n+2$ is $O(5n-3\log(n) + 17)$
  - and it would be technically correct…
  - It would also be poor taste … and put me in a bad mood.
- **But… if I ask "true or false: $3n+2$ is $O(n^3)$", what's the answer?**
  - True!
  - There may be "trick" questions like this on assignments and exams.
  - But they aren't really tricks, just following the big-Oh definition!

Examples / practice

- Sorting and searching
  - Why we study these
- See project: SortingAndSearching
  - Counting: Loops