Design and Analysis
of Algorithms I

Introduction

Guiding Principles

Guiding Principle #1

"worst-case analysis": our running time bound holds for every input of length n.
- particularly appropriate for "general-purpose" routines

As opposed to

- "average-case" analysis
- benchmarks

} requires domain knowledge

BONUS: worst case usually easier to analyze.

Guiding Principle #2

Won't pay much attention to constant factors, lower-order terms.

Justifications

(1) way easier

(2) constants depend on architecture / compiler / programmer anyways

(3) lose very little predictive power (as we'll see)

Guiding Principle #3

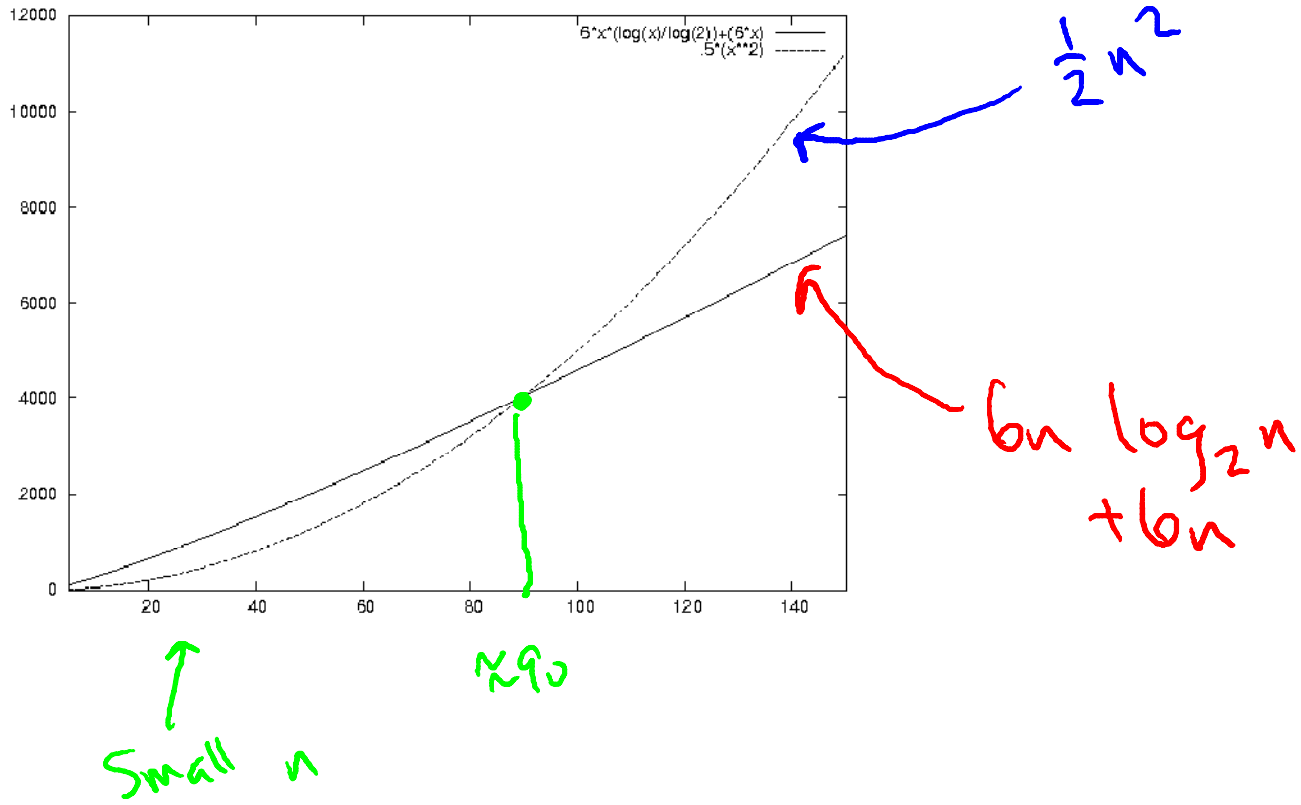
Asymptotic analysis: focus on running time for large input sizes n.

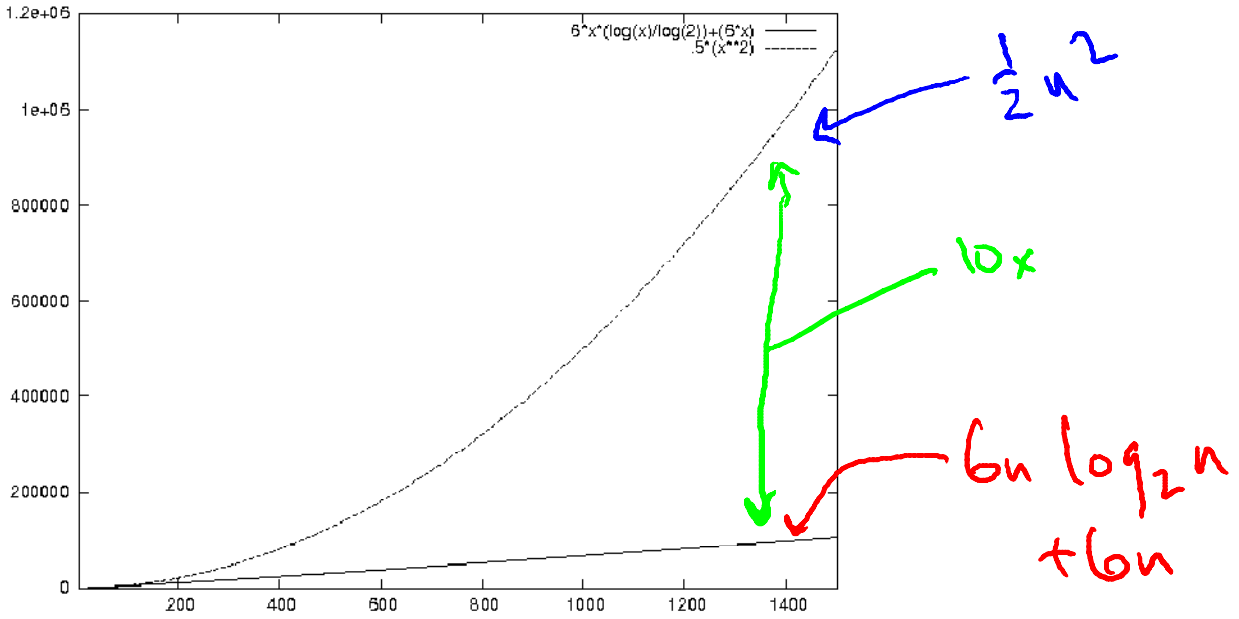
E.g.: $6n \log_2 n + 6n$ "better than" $\frac{1}{2} n^2$

Merge Sort

≈ Insertion Sort

Justification: only big problems are interesting!


6*x*(log(x)/log(2))+(6*x)
.5*(x**2)
12000
10000
8000
6000
4000
2000
0
20
40
60
80
100
120
140
$\frac{1}{2}n^2$
$6n \log_2 n + 6n$
n_0
Small n


6*x*(log(x)/log(2))+(6*x)
.5*(x**2)
1.2e+06
1e+06
800000
600000
400000
200000
0
200
400
600
800
1000
1200
1400
$\frac{1}{2}n^2$
10x
$6n \log_2 n$
$+6n$

What Is a "Fast" Algorithm?

This course: adopt these three biases as guiding principles.

> Fast algorithm $\approx$ worst-case running time grows slowly with input size

Usually: want as close to linear ($O(n)$) as possible.