Growth rate, Big-O notation

CS110C
Max Luttrell, CCSF

growth rate

- let's say we know the following, for two algorithms acting on a given list of n items on a given computer:
  - algorithm A requires $n^2 / 5$ seconds
  - algorithm B requires $5 * n$ seconds

growth rate

- let's say we know the following, for two algorithms acting on a given list of n items on a given computer:
  - algorithm A requires $n^2 / 5$ seconds
  - algorithm B requires 5 * n seconds

n	A	B
5	5	25

growth rate

- let's say we know the following, for two algorithms acting on a given list of n items on a given computer:
  - algorithm A requires $n^2 / 5$ seconds
  - algorithm B requires $5 * n$ seconds

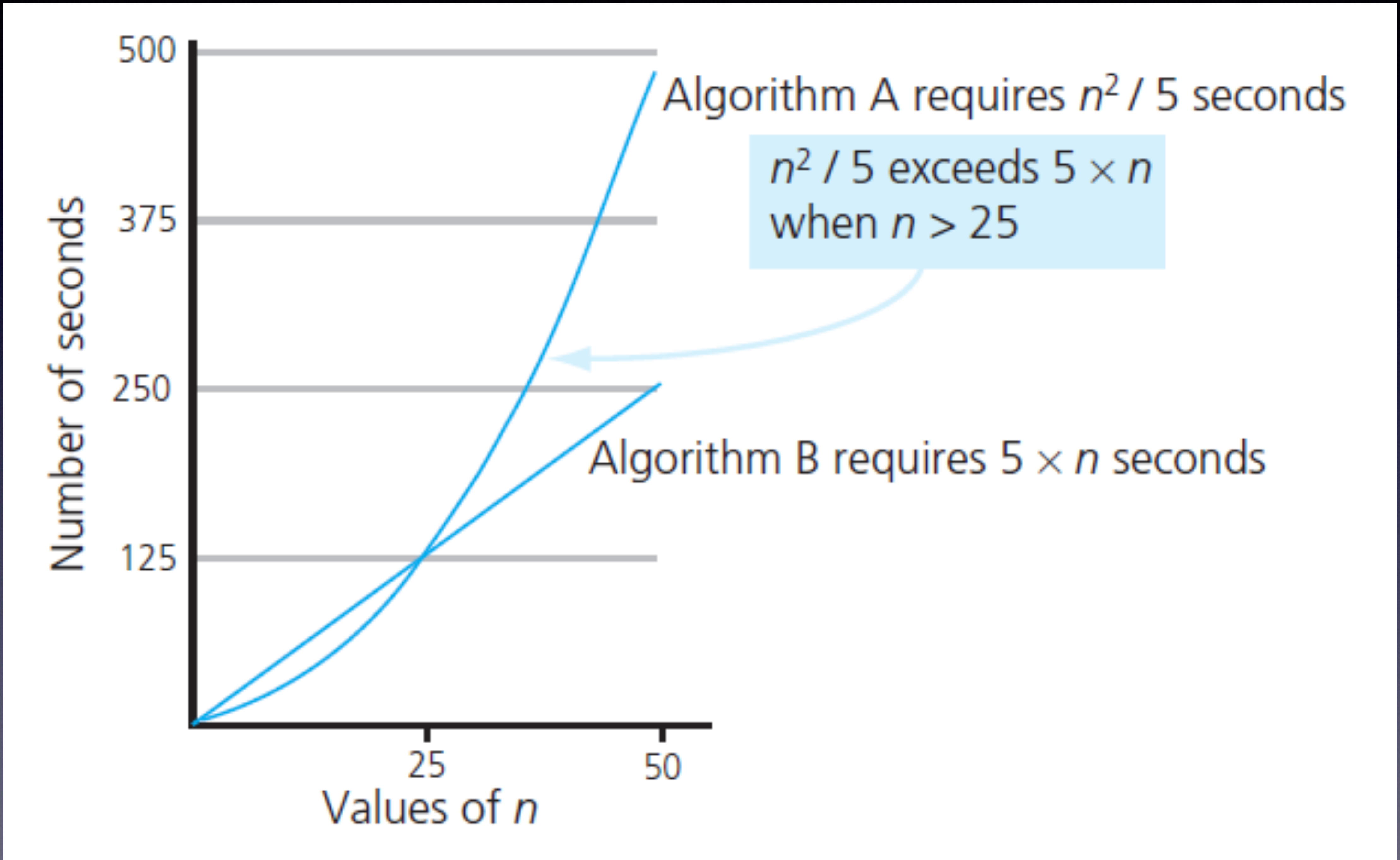
n	A	B
5	5	25
10	20	50

growth rate

- let's say we know the following, for two algorithms acting on a given list of n items on a given computer:
  - algorithm A requires $n^2 / 5$ seconds
  - algorithm B requires 5 * n seconds

n	A	B
5	5	25
10	20	50
100	2000	500

growth rate



500
375
250
125
Number of seconds
25
50
Values of n
Algorithm A requires $n^2 / 5$ seconds
$n^2 / 5$ exceeds $5 \times n$ when $n > 25$
Algorithm B requires $5 \times n$ seconds

order

- let's say we know the following, for three algorithms acting on a list of n items:
  - algorithm A is proportional to n^2
  - algorithm B is proportional to n
  - algorithm C is proportional to f(n)
- f(n) is the **growth rate function** of the algorithm, a.k.a. **order f(n)**, a.k.a. **O(f(n)) (big O notation)**

growth rates

- Some common orders in ascending order of growth rate

$$O(1) < O(\log_2 n) < O(n) < O(n \times \log_2 n) <$$
$$O(n^2) < O(n^3) < O(2^n)$$

determining order

1. You can ignore lower-order terms in an algorithm's growth rate function. An algorithm which is $O(n^2 + n)$ is also $O(n^2)$.

2. You can ignore constants. An algorithm which is $O(3n^2 + 2n)$ is also $O(n^2 + n)$, which from property 1 is also $O(n^2)$.

3. You can combine growth rate functions. An algorithm which is $O(n^2)+ O(n)$ can be rewritten $O(n^2 + n)$. We know this is $O(n^2)$ by property 1.

$$O(1) < O(\log_2 n) < O(n) < O(n \times \log_2 n) <$$
$$O(n^2) < O(n^3) < O(2^n)$$

worst case vs. average case

- Maximum amount of time to solve a problem of size n: **worst case**
- Average amount of time to solve a problem of size n: **average case**

some perspective

- if we know n is always small, e.g. less than 25, the order of the solution is likely insignificant
- choosing between implementations: consider what type of operations frequently occur (e.g. for a list ADT, dictionary vs. a list of online friends)
- note, however, that efficiency might still be critical even if the operation is infrequently used, i.e. an air traffic control emergency operation.