Goals for today:

- Characterize how functions grow as the inputs get really big.
- Use big-O notation to characterize the running time of algorithms.

Recall candy distribution problem from last class:

solution 1

```
def distribute_candies1(n: int, limit: int) -> int:
    """Counts the number of ways to distribute 'n' candies
       to 3 people so that each person has a maximum
       of 'limit' candies."""
    n_ways = 0
    for i in range(limit + 1):
        for j in range(limit + 1):
            for k in range(limit + 1):
                if i + j + k == n:
                    n_ways += 1
    return n_ways
```

n = l = 250 ? 5 sec.
= 500 ? 40 sec.
= 1000 ? 5-6 mins.

Possible solutions to the candy distribution problem from last class.

how many +, −, ∗, /, >, <, == are used?

```python
def distribute_candies1(n: int, limit: int) -> int:
    n_ways = 0
    limit_plus_one = limit + 1
    for i in range(limit_plus_one):
        for j in range(limit_plus_one):
            for k in range(limit_plus_one):
                if i + j + k == n:
                    n_ways += 1
    return n_ways
```

$1 + 4(\ell+1)(\ell+1)(\ell+1) \simeq 1 + 4(\ell+1)^3$

4 operations

solution 2

```python
def distribute_candies2(n: int, limit: int) -> int:
    n_ways = stars_and_bars(n, 3)
    limit_plus_one = limit + 1
    n_ways -= 3 * stars_and_bars(n - limit_plus_one, 3)
    n_ways += 3 * stars_and_bars(n - 2 * limit_plus_one, 3)
    n_ways -=     stars_and_bars(n - 3 * limit_plus_one, 3)
    return n_ways
```

$1 + 3 + 2 + 3 + 2 + 3(4) = 23$

no dependence on ℓ.

$k = 3$

$\binom{n+k-1}{k-1} = \binom{n+2}{2} = \frac{(n+2)!}{n!\,2!} = \frac{(n+2)(n+1)}{2}$ → how many ops? 4

math.comb

Try to do a detailed analysis of the number of operations performed by this algorithm? (count arithmetic & comparison operators).

+, -, *, / <, >, ==

Assume the *worst-case*: `value` is not in the list.

```
1 def linear_search(a: list[int], value: int) -> int:
2     n = len(a)
3     i = 0
4     while i < n:
5         if a[i] == value:
6             return i
7         i += 1
8     return -1 # not found
```

results:

2n-1

3n

2+2n

how do we analyze this?

as n gets big.

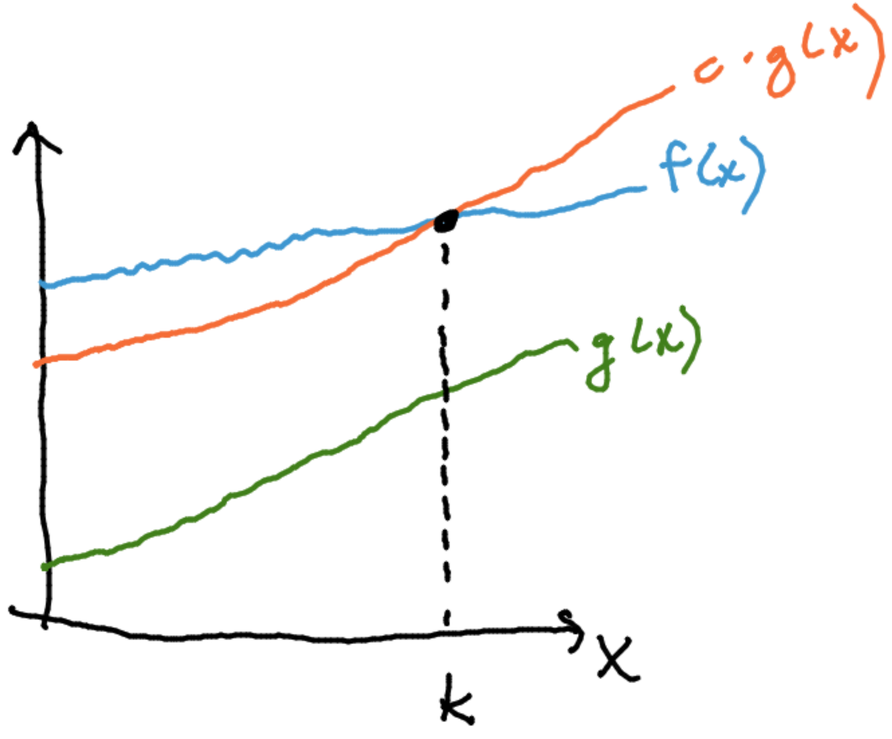
We need a better way to analyze running time of algorithms.

big-O notation: Given $f,\ g: \mathbb{R} \to \mathbb{R}$, we say that $f(x)$ is $\mathcal{O}(g(x))$ if-and-only-if there *exist* constants $c > 0$ and k such that

"eventually"

$$|f(x)| \leq c \cdot |g(x)|, \quad \forall x \geq k$$

in words: "f(x) is eventually no larger than some constant multiple of g(x)"



1.) to show find pair (c, k)

a.) pick k

b.) go back and calculate c

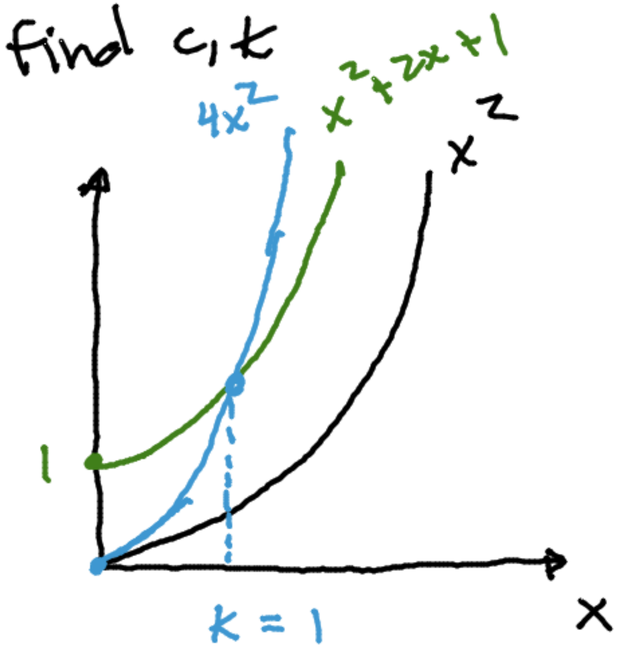
2.) use alternative definition using limits

show $\lim_{x \to \infty} \frac{f(x)}{g(x)} < \infty$ (finite)

Example 1: Show that $x^2 + 2x + 1$ is $\mathcal{O}(x^2)$

$f(x)$ $\quad$ $g(x)$

1.) find c, k



pick $c = 4$ $\quad 4x^2$

$k = 1$

2.) using limits

$$\lim_{x \to \infty} \frac{f(x)}{g(x)} = \lim_{x \to \infty} \frac{x^2 + 2x + 1}{x^2}$$

$$= \lim_{x \to \infty} 1 + \frac{1}{2x} + \frac{1}{x^2}$$

0 $\quad$ 0

what happens when x gets really big?

which terms remain?

$= 1 < \infty$? yes

(finite)

Which of the following statements is true?

"f is O(g)"

$f \in O(g)$

Which of the following statements are true?
(slido.com #1825775)

40

- [ ] x^2 is O(1000000000x) ✗
- [x] 100x^2 is O(x^2)
- [x] 1000000x is O(x^2)
- [x] 10 is O(1)

$\lim_{x \to \infty} \frac{x^2}{10^9 x} \to \infty$

$\lim_{x \to \infty} \frac{10^6 x}{x^2} = \lim_{x \to \infty} \frac{10^6}{x} = 0$

Voting as Anonymous

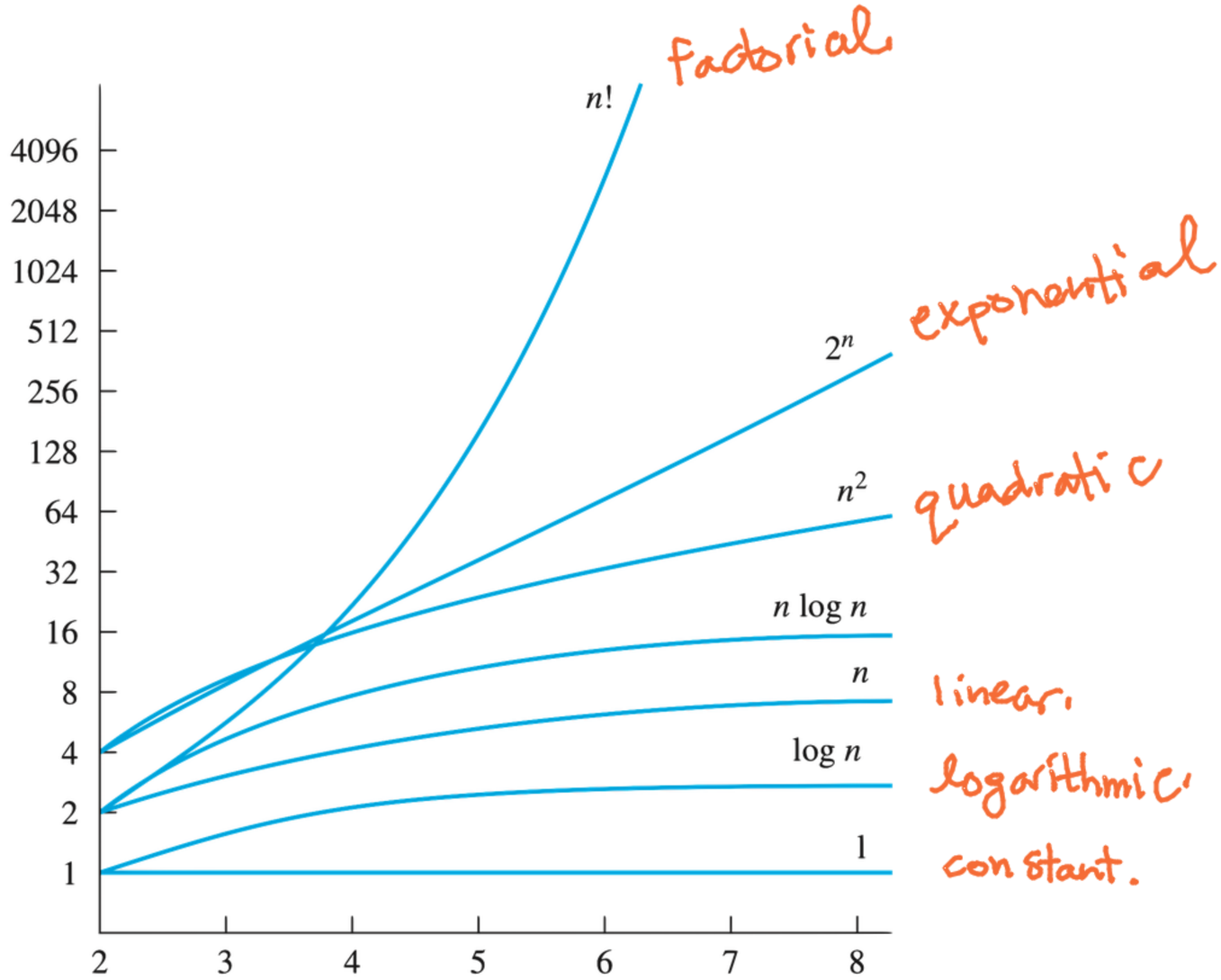
Send

Example 2: Show that $\underbrace{\log_3(n^2)}_{f(n)}$ is $\mathcal{O}(\underbrace{\log_2 n}_{g(n)})$.

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{\log_3(n^2)}{\log_2 n} = \lim_{n\to\infty} \frac{2\log_3 n}{\log_2 n} \overset{\lim}{=} 2\,\frac{\frac{\log n}{\log 3}}{\frac{\log n}{\log 2}} =$$

$$= \lim_{n\to\infty} 2\,\frac{\frac{\log n}{\log 3}}{\frac{\log n}{\log 2}} = \frac{2\log 2}{\log 3} < \infty \quad \text{(finite)} \quad \checkmark$$

Common functions used in big-O estimates.



(Discrete Mathematics and Its Applications 7th Ed., Rosen)