Illustration of big-O notation

- f is $O(g)$ if $\exists C, k > 0 : f(x) \leq C \cdot g(x)$ for all $x \geq k$

Illustration of big-O notation

- f is $O(g)$ if $\exists C, k > 0 : f(x) \leq C \cdot g(x)$ for all $x \geq k$

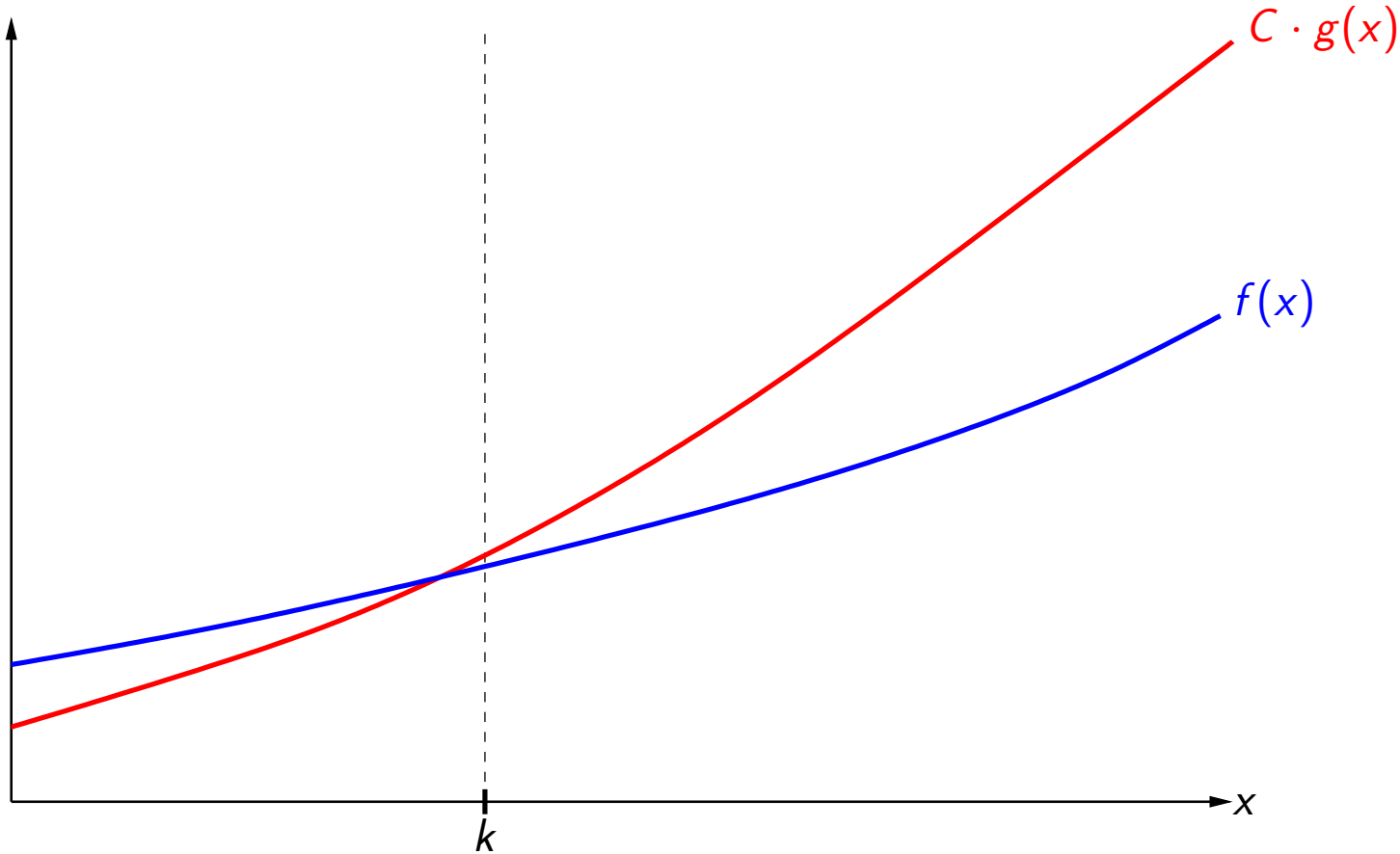
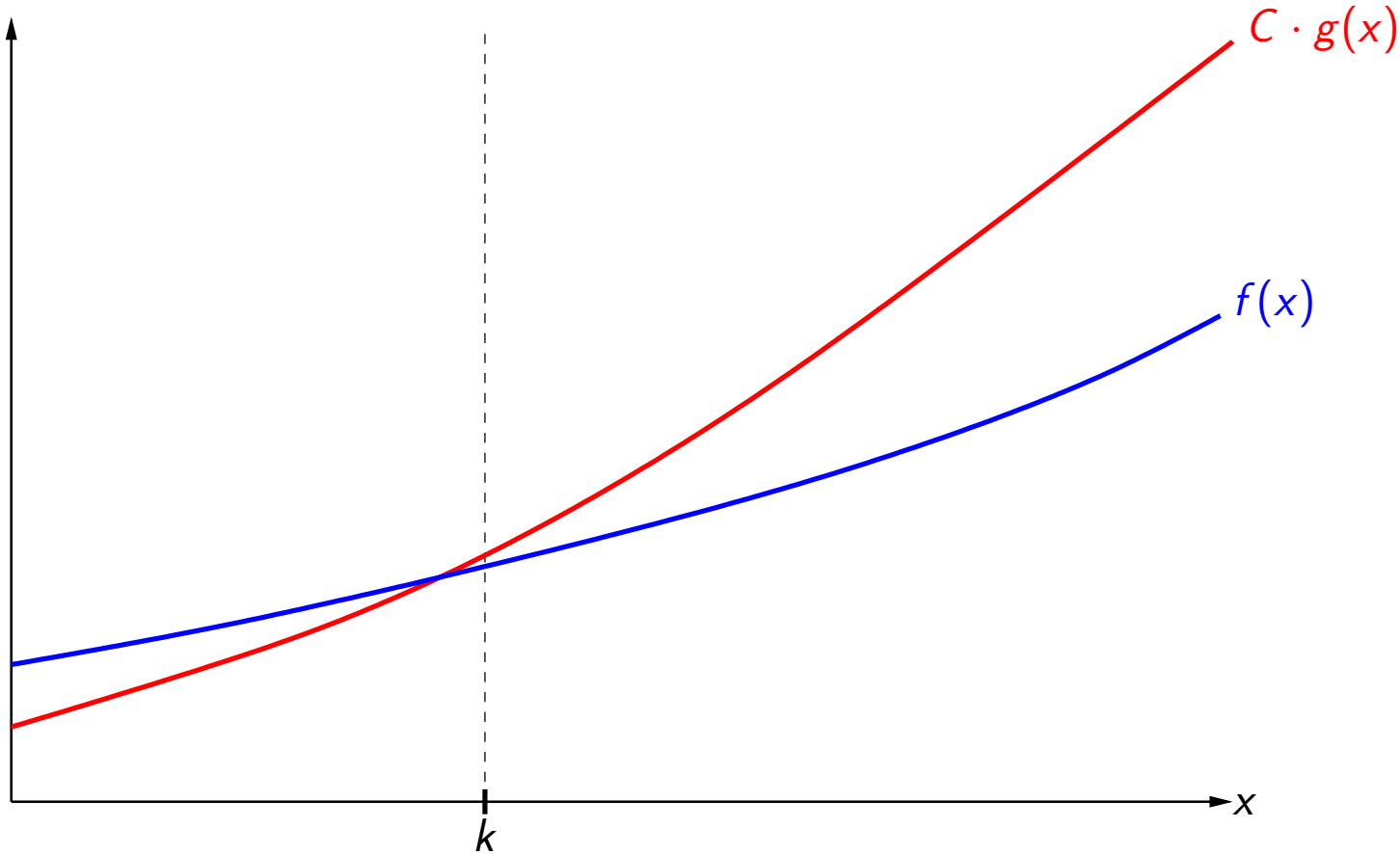


Illustration of big-O notation

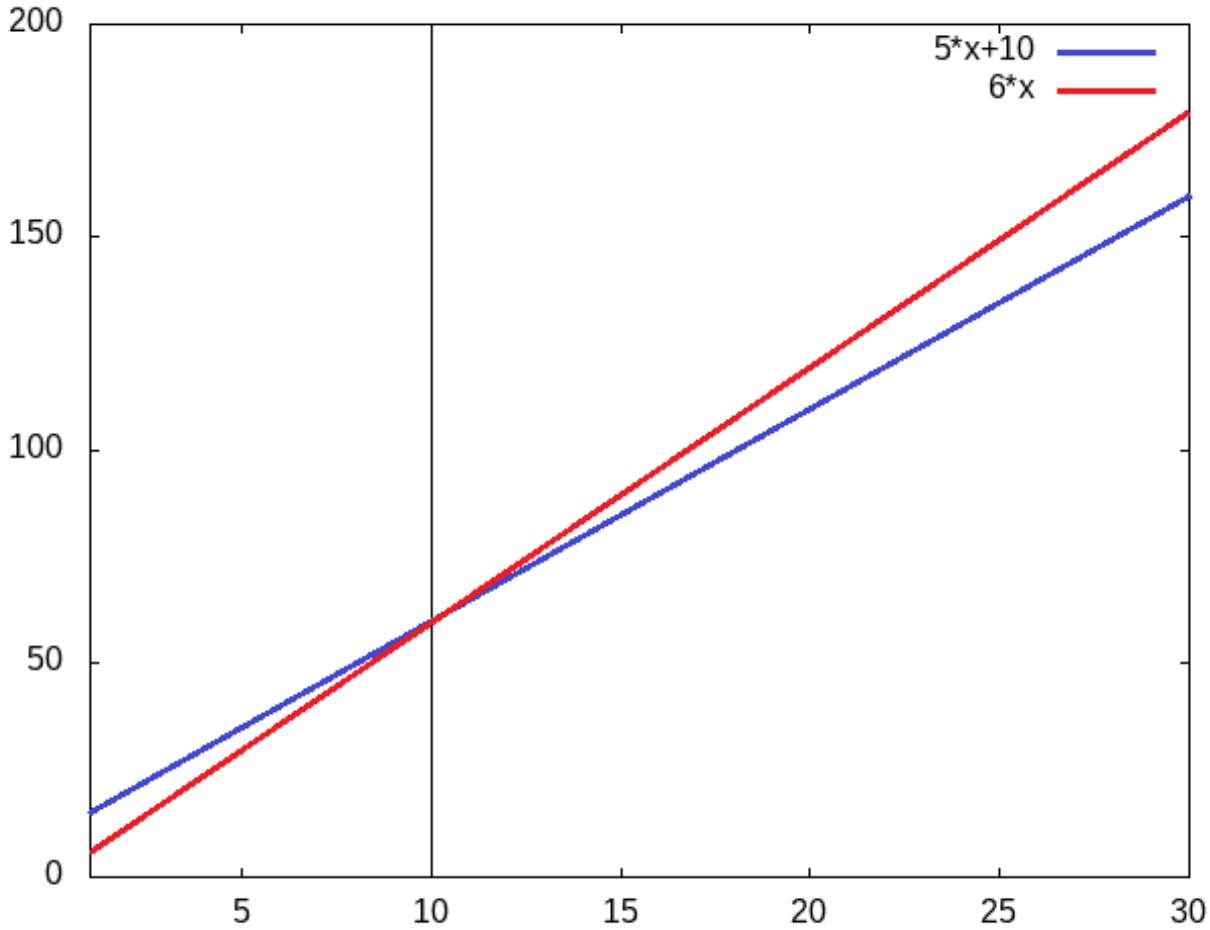
- f is $O(g)$ if $\exists C, k > 0 : f(x) \leq C \cdot g(x)$ for all $x \geq k$



- From $x = k$ onward, $f(x)$ never exceeds $C \cdot g(x)$.

Illustration of witnesses that work

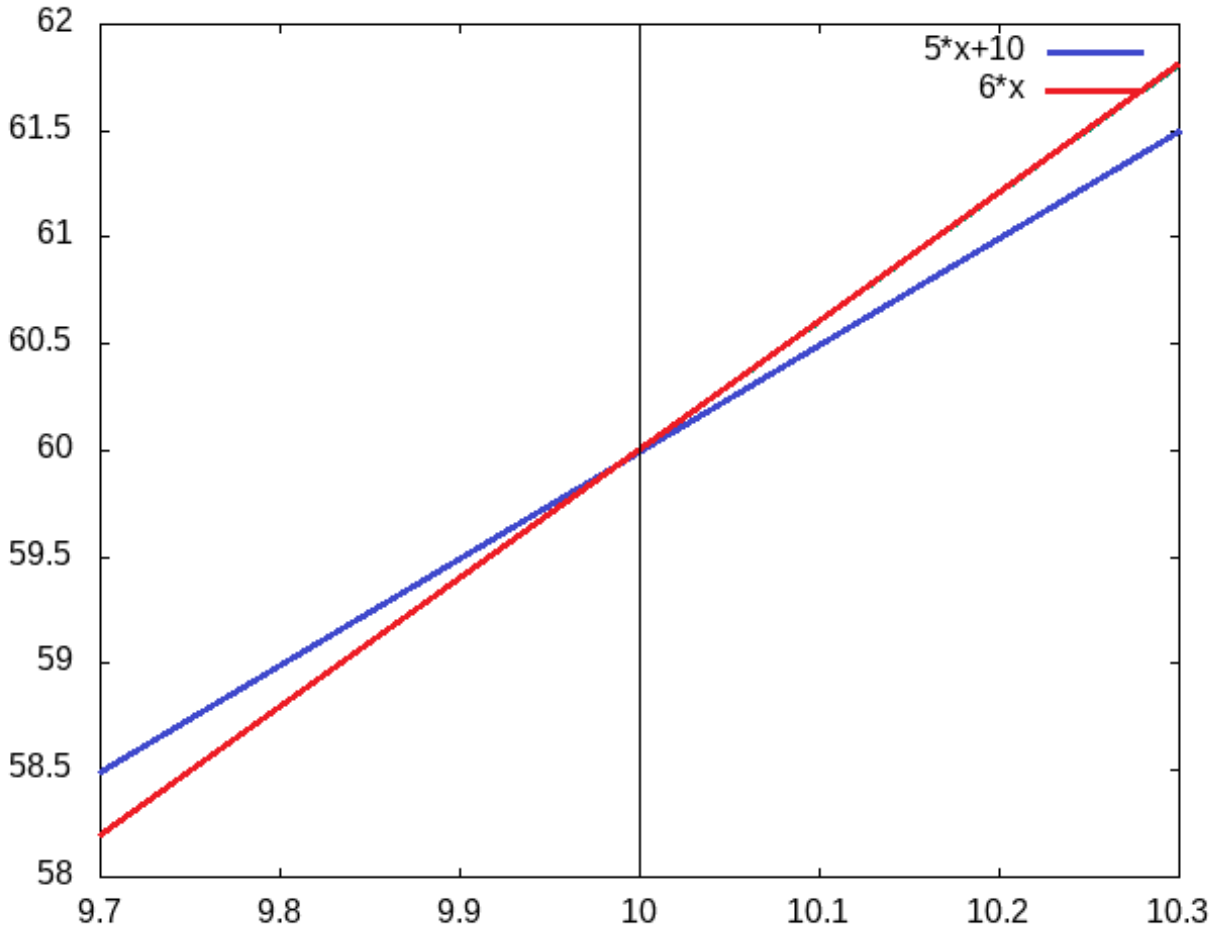
- $C = 6$, $k = 10$ are witnesses for showing that $5x + 10$ is $O(x)$



- From $x = 10$ onward, $5x + 10$ never exceeds $6 \cdot x$.

Illustration of witnesses that work

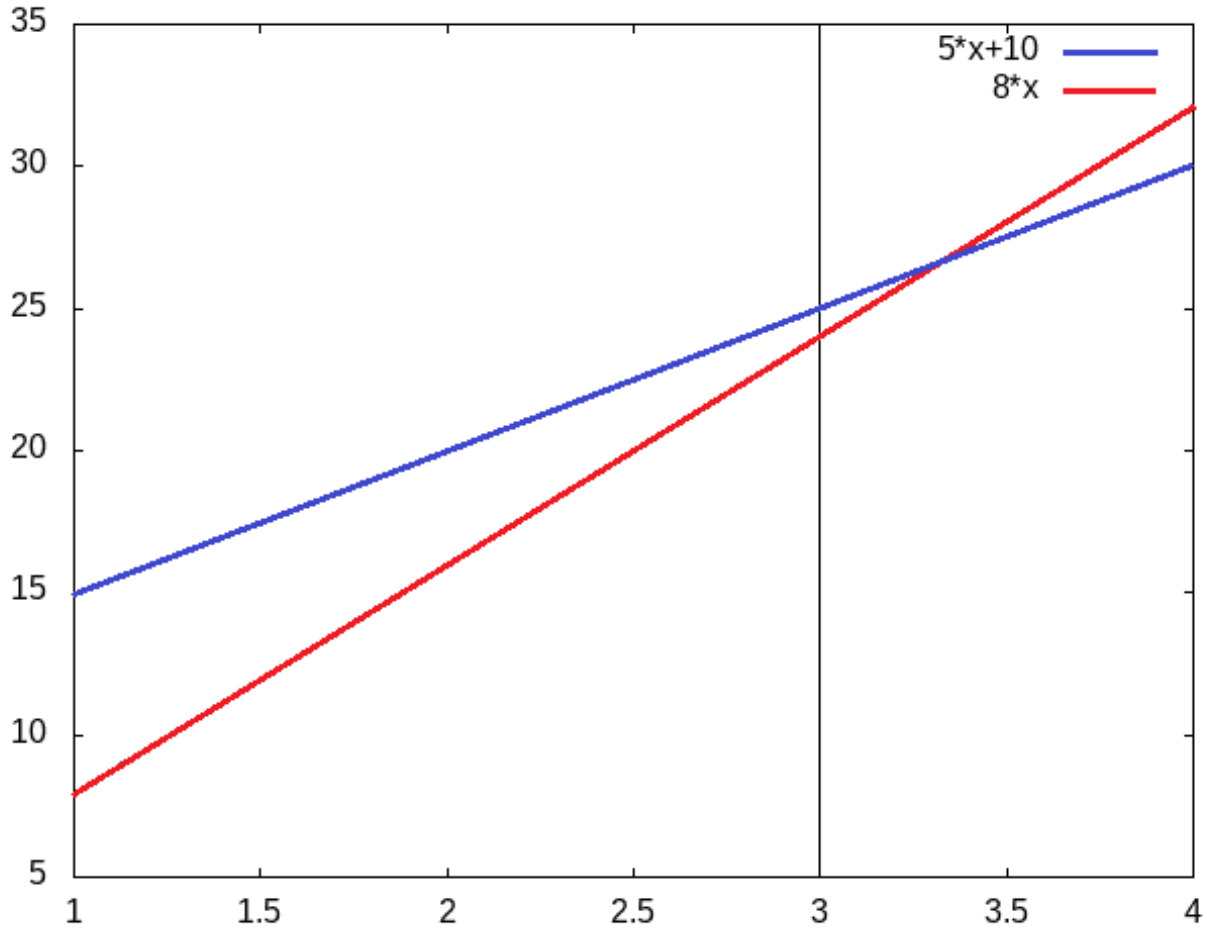
- $C = 6$, $k = 10$ are witnesses for showing that $5x + 10$ is $O(x)$



- From $x = 10$ onward, $5x + 10$ never exceeds $6 \cdot x$.

Illustration of witnesses that don't work

- $C = 8$, $k = 3$ are **NOT** witnesses for $5x + 10 = O(x)$



- At $x = 3$, $5x + 10 = 25$ exceeds $8 \cdot x = 24$.

Why do we use big-O notation?

- We don't want to say: The time complexity of our algorithm is $15n^2 + 10n \log n + 15n + 37$.

Why do we use big-O notation?

- We don't want to say: The time complexity of our algorithm is $15n^2 + 10n \log n + 15n + 37$.
  (If we count basic operations a bit differently, it might be something like $20n^2 + 12n \log n + 19n + 48$, anyway.)

Why do we use big-O notation?

- We don't want to say: The time complexity of our algorithm is $15n^2 + 10n \log n + 15n + 37$.
  (If we count basic operations a bit differently, it might be something like $20n^2 + 12n \log n + 19n + 48$, anyway.)
- If the time complexity is $15n^2 + 10n \log n + 15n + 37$, the main thing that matters when n gets large is that the dominant (fastest growing) term is $15n^2$.

Why do we use big-O notation?

- We don't want to say: The time complexity of our algorithm is $15n^2 + 10n \log n + 15n + 37$.
  (If we count basic operations a bit differently, it might be something like $20n^2 + 12n \log n + 19n + 48$, anyway.)
- If the time complexity is $15n^2 + 10n \log n + 15n + 37$, the main thing that matters when n gets large is that the dominant (fastest growing) term is $15n^2$.
- The fact that the factor in front of n^2 is 15 is less important to us (besides, it might be 20 or something else if we count basic operations a bit differently).

Why do we use big-O notation?

- We don't want to say: The time complexity of our algorithm is $15n^2 + 10n \log n + 15n + 37$.
  (If we count basic operations a bit differently, it might be something like $20n^2 + 12n \log n + 19n + 48$, anyway.)
- If the time complexity is $15n^2 + 10n \log n + 15n + 37$, the main thing that matters when n gets large is that the dominant (fastest growing) term is $15n^2$.
- The fact that the factor in front of n^2 is 15 is less important to us (besides, it might be 20 or something else if we count basic operations a bit differently).
- We want to express in a simple but rigorous way that the time complexity grows with the square of the size of the input.

Why do we use big-O notation?

- We don't want to say: The time complexity of our algorithm is $15n^2 + 10n \log n + 15n + 37$.
  (If we count basic operations a bit differently, it might be something like $20n^2 + 12n \log n + 19n + 48$, anyway.)
- If the time complexity is $15n^2 + 10n \log n + 15n + 37$, the main thing that matters when n gets large is that the dominant (fastest growing) term is $15n^2$.
- The fact that the factor in front of n^2 is 15 is less important to us (besides, it might be 20 or something else if we count basic operations a bit differently).
- We want to express in a simple but rigorous way that the time complexity grows with the square of the size of the input.
- We could say: The time complexity of our algorithm is bounded by a constant factor times n^2.

Why do we use big-O notation?

- We don't want to say: The time complexity of our algorithm is $15n^2 + 10n \log n + 15n + 37$.
  (If we count basic operations a bit differently, it might be something like $20n^2 + 12n \log n + 19n + 48$, anyway.)
- If the time complexity is $15n^2 + 10n \log n + 15n + 37$, the main thing that matters when n gets large is that the dominant (fastest growing) term is $15n^2$.
- The fact that the factor in front of n^2 is 15 is less important to us (besides, it might be 20 or something else if we count basic operations a bit differently).
- We want to express in a simple but rigorous way that the time complexity grows with the square of the size of the input.
- We could say: The time complexity of our algorithm is bounded by a constant factor times n^2.
- We do say: The time complexity of our algorithm is $O(n^2)$.

Using big-O notation in a natural way

If the time complexity of our algorithm is:

$$12 + 17\sqrt{n} + 5n + 17n \log n$$

How should we express this with big-O notation? Should we say that the time complexity is in:

- $O(n)$
- $O(n \log n)$
- $O(17n \log n)$
- $O(n \log n + n)$
- $O(n^2)$

Using big-O notation in a natural way

If the time complexity of our algorithm is:

$$12 + 17\sqrt{n} + 5n + 17n \log n$$

How should we express this with big-O notation? Should we say that the time complexity is in:

- $O(n)$: WRONG
- $O(n \log n)$
- $O(17n \log n)$
- $O(n \log n + n)$
- $O(n^2)$

Using big-O notation in a natural way

If the time complexity of our algorithm is:

$$12 + 17\sqrt{n} + 5n + 17n \log n$$

How should we express this with big-O notation? Should we say that the time complexity is in:

- $O(n)$: WRONG
- $O(n \log n)$: ✓ correct and natural
- $O(17n \log n)$
- $O(n \log n + n)$
- $O(n^2)$

Using big-O notation in a natural way

If the time complexity of our algorithm is:

$$12 + 17\sqrt{n} + 5n + 17n \log n$$

How should we express this with big-O notation? Should we say that the time complexity is in:

- $O(n)$: WRONG
- $O(n \log n)$: ✓ correct and natural
- $O(17n \log n)$: correct but not natural: why not omit the factor 17?
- $O(n \log n + n)$
- $O(n^2)$

Using big-O notation in a natural way

If the time complexity of our algorithm is:

$$12 + 17\sqrt{n} + 5n + 17n \log n$$

How should we express this with big-O notation? Should we say that the time complexity is in:

- $O(n)$: WRONG
- $O(n \log n)$: ✓ correct and natural
- $O(17n \log n)$: correct but not natural: why not omit the factor 17?
- $O(n \log n + n)$: correct but not natural: why add the $+n$ that is not necessary?
- $O(n^2)$

Using big-O notation in a natural way

If the time complexity of our algorithm is:

$$12 + 17\sqrt{n} + 5n + 17n \log n$$

How should we express this with big-O notation? Should we say that the time complexity is in:

- $O(n)$: WRONG
- $O(n \log n)$: ✓ correct and natural
- $O(17n \log n)$: correct but not natural: why not omit the factor 17?
- $O(n \log n + n)$: correct but not natural: why add the $+n$ that is not necessary?
- $O(n^2)$: correct but not natural: why give a larger bound than necessary?