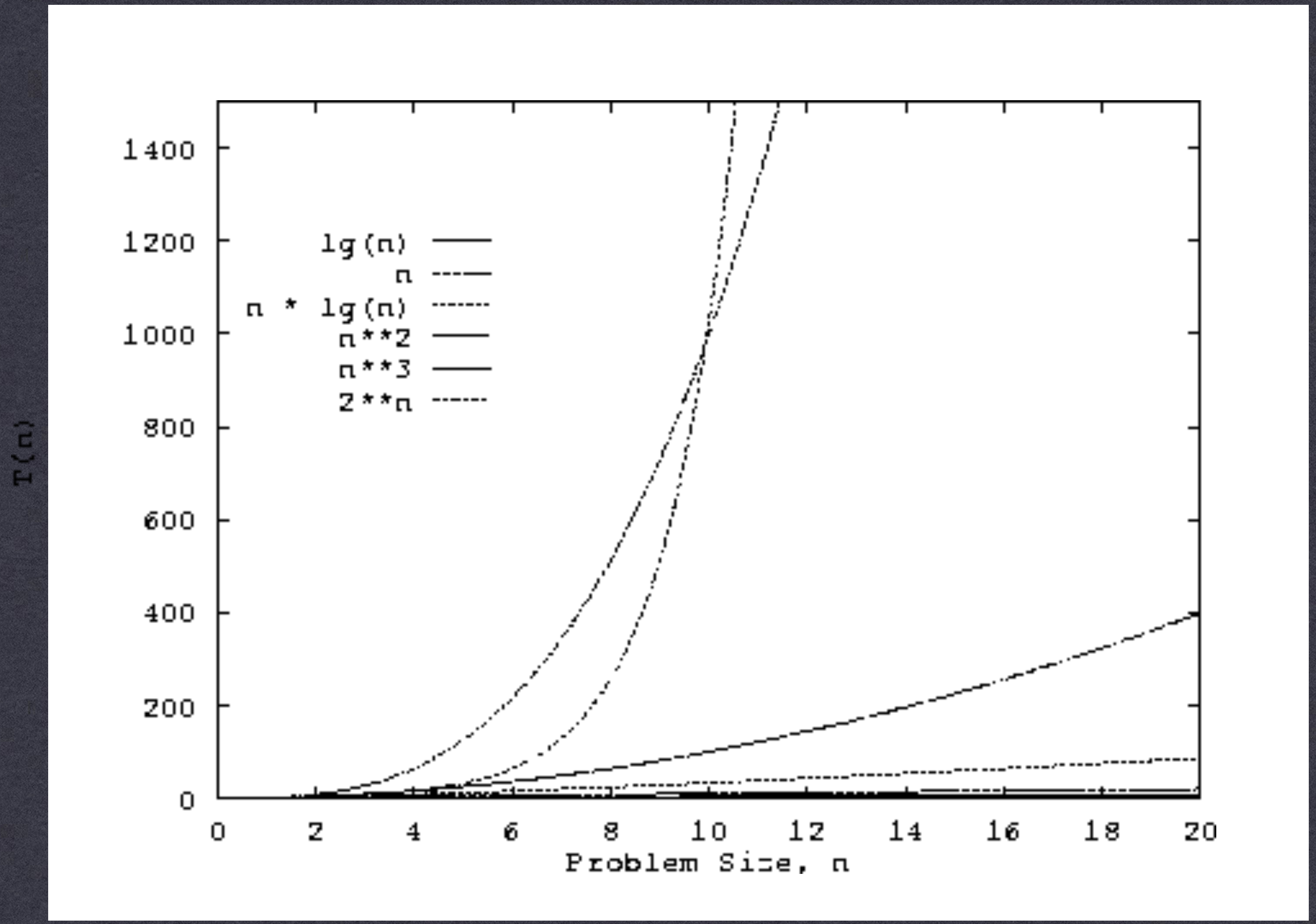
Asymptotic Analysis



Comparing Algorithms

- Linked Lists and Array List
  - Which one is best?

Comparing Algorithms

- Linked Lists and Array List
  - Which one is **fastest**?

Comparing Algorithms

- Linked Lists and Array List
  - Which one is **fastest**?
  - How can we compare?
  - How do we know if an algorithm/data structure is efficient?
  - What does it mean to be efficient?

Complexity

- time complexity
  - how many *operations* are needed
- space complexity
  - how much *memory* is needed

Let’s focus on speed

- We need to **count the operations**
  - Determine the cost per operation
  - Count the number of times an operation is executed
  - Sum over all operations

Measuring runtime

```
int arrayMax(int A[], int len)
{
        int currentMax = A[0];
        for (int i = 1; i < len; i++){
                if (currentMax < A[i]) {
                        currentMax = A[i];
                }
        }
        return currentMax;
}
```

executed once (2 ops)

ex. n - 1 times (2 * (n - 1) ops)

ex. n times (n ops)

ex. once (1 op)

ex. n - 1 times (2 * (n - 1) ops)

ex. at most n - 1 times (2 * (n - 1) ops)

Loop total: between 4 * (n - 1) and 6 * (n - 1)

ex. once (1 op)

- How many operations this procedure needs?

Let’s simplify

- Assume all operations need 1 unit of time
- Let’s count best, worst, and average worst case
    - Best Case:
    - Worst Case
    - Average?

Let’s simplify

- Assume all operations need 1 unit of time
- Let’s count best, worst, and average worst case
    - Best Case: $2 + 1 + n + 4(n-1) + 1 = \mathbf{5n}$
    - Worst Case $2 + 1 + n + 6(n-1) + 1 = \mathbf{7n - 2}$
    - Average? **Depends on input**

Problems

- Computing the exact number is tedious (unfeasible for big code)
- Exact number of operations depends on many parameters
  - Compiler (under the hood improvements)
  - Computer that it is run (allocate variables in register? RAM?)
  - Operating System
- Sometimes it is hard to even compute
  - How much time does a search take?

Impractical!

What is common about these functions?

$5n + \frac{1}{2}n^2 + \frac{1}{100} \cdot n^3$ $\quad$ $18n^3 - 12$ $\quad$ $40n^3 + \log n^{10}$

What is common about these functions?

$5n + \frac{1}{2}n^2 + \frac{1}{100} \cdot \underline{n^3}$ $\qquad$ $18\underline{n^3} - 12$ $\qquad$ $40\underline{n^3} + \log n^{10}$

The dominant term contains n^3

What is common about these functions?

$5n + \frac{1}{2}n^2 + \frac{1}{100} \cdot n^3$ $\quad 18n^3 - 12$ $\quad 40n^3 + \log n^{10}$

The dominant term contains n^3

↳ what does this mean? For $n=5$, $\frac{1}{100} \cdot n^3$ doesn't dominate.

$< 5n$ & $\frac{1}{2}n^2$

What is common about these functions?

$5n + \frac{1}{2}n^2 + \frac{1}{100} \cdot n^3$ $\quad$ $18n^3 - 12$ $\quad$ $40n^3 + \log n^{10}$

The dominant term contains n^3

what does this mean? For $n=5$, $\frac{1}{100} \cdot n^3$ doesn't dominate.

$< 5n \;\&\; \frac{1}{2}n^2$

n^3 dominates for all n larger than some integer.

What is common about these functions?

$5n + \frac{1}{2}n^2 + \frac{1}{100}\cdot n^3$ $\quad$ $18n^3 - 12$ $\quad$ $40n^3 + \log n^{10}$

- The dominant term contains n^3

what does this mean? For $n=5$, $\frac{1}{100}\cdot n^3$ doesn't dominate.

$< 5n$ & $\frac{1}{2}n^2$

n^3 dominates for all n larger than some integer.

- All three functions have $50n^3$ as an upper bound, for $n \geq 1$

What is common about these functions?

$5n + \frac{1}{2}n^2 + \frac{1}{100} \cdot n^3$ $18n^3 - 12$ $40n^3 + \log n^{10}$

- The dominant term contains n^3

what does this mean? For $n=5$, $\frac{1}{100} \cdot n^3$ doesn't dominate.

$< 5n \; \& \; \frac{1}{2}n^2$

n^3 dominates for all n larger than some integer.

- All three functions have $50n^3$ as an upper bound, for $n \geq 1$
- All three functions have $\frac{1}{100}n^3$ as a lower bound, for $n \geq 1$

$\boxed{5n + \frac{1}{2}n^2 + \frac{1}{100} \cdot n^3}$ $\boxed{18n^3 - 12}$ $\boxed{40n^3 + \log n^{10}}$

- All three functions have $50n^3$ as an upper bound, for $n \geq 1$
- All three functions have $\frac{1}{100}n^3$ as a lower bound, for $n \geq 1$

$$\boxed{5n + \frac{1}{2}n^2 + \frac{1}{100}\cdot n^3} \qquad \boxed{18n^3 - 12} \qquad \boxed{40n^3 + \log n^{10}}$$

There exist constants $c > 0$, $n_0 > 0$ such that for all $n \geqslant n_0$:

All three functions have cn^3 as an upper bound

(all $\leqslant cn^3$)

- All three functions have $50n^3$ as an upper bound, for $n \geqslant 1$
- All three functions have $\frac{1}{100}n^3$ as a lower bound, for $n \geqslant 1$

$\boxed{5n + \frac{1}{2}n^2 + \frac{1}{100}\cdot n^3}$ $\boxed{18n^3 - 12}$ $\boxed{40n^3 + \log n^{10}}$

There exist constants $c > 0$, $n_0 > 0$ such that for all $n \geq n_0$:

All three functions have cn^3 as an upper bound

(all $\leq cn^3$)

There exist constants $d > 0$, $n_1 > 0$ such that for all $n \geq n_1$:

All three functions have dn^3 as a lower bound

(all $\geq dn^3$)

- All three functions have $50n^3$ as an upper bound, for $n \geq 1$
- All three functions have $\frac{1}{100}n^3$ as a lower bound, for $n \geq 1$

If there exist constants $c > 0$, $n_0 > 0$ such that for all $n \geqslant n_0$:

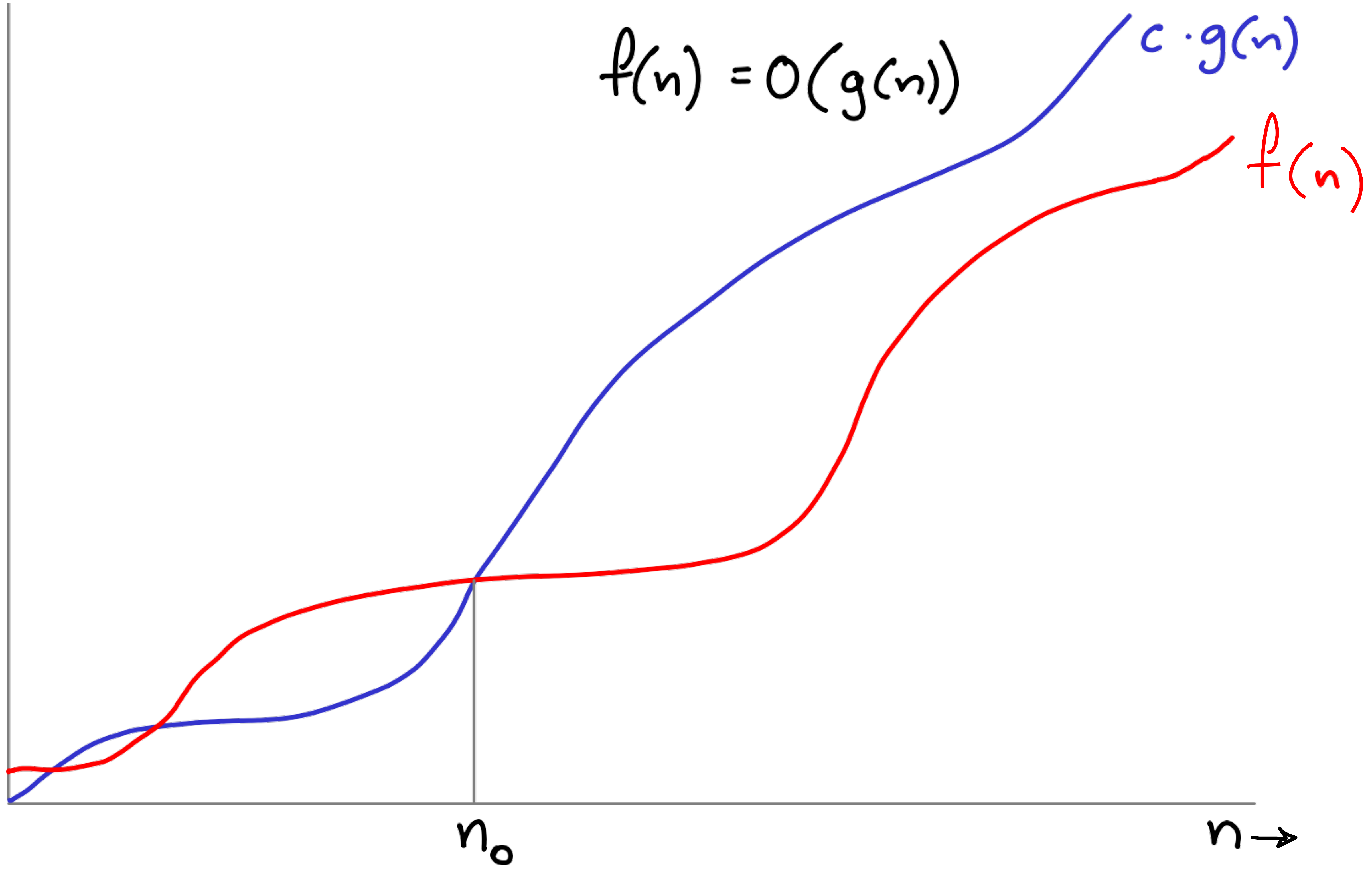
$$f(n) \leq c n^3$$

If there exist constants $c > 0$, $n_0 > 0$ such that for all $n \geqslant n_0$:

$$f(n) \leq cn^3$$

then we say $f(n) = O(n^3)$

$f(n) \leq cn^3$



If there exist constants $c > 0$, $n_0 > 0$ such that for all $n \geqslant n_0$:

$$f(n) \leqslant cn^3$$

then we say $f(n) = O(n^3)$

If there exist constants $d > 0$, $n_1 > 0$ such that for all $n \geqslant n_1$:

$$f(n) \geqslant dn^3$$

If there exist constants $c > 0$, $n_0 > 0$ such that for all $n \geqslant n_0$:

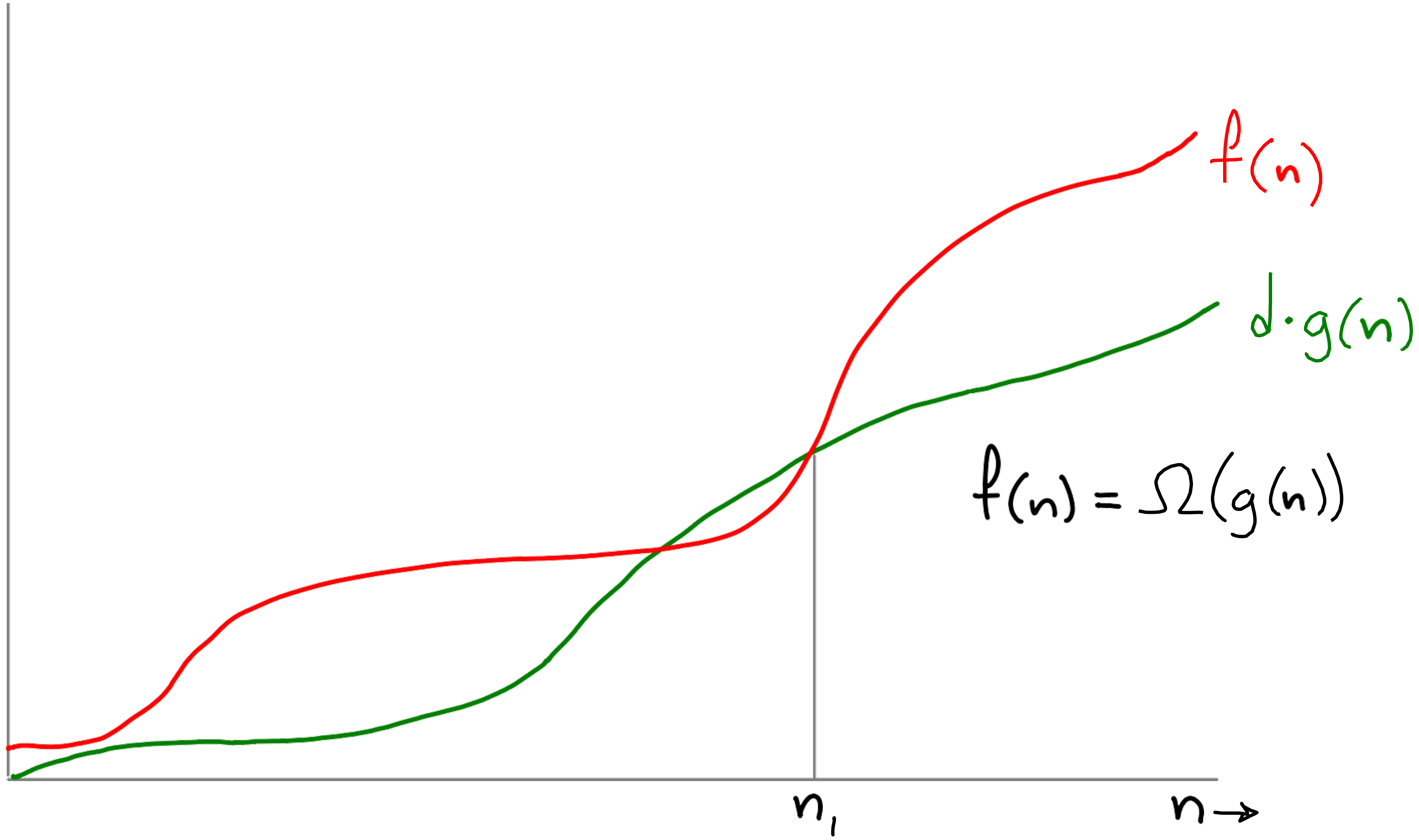
$$f(n) \leqslant cn^3$$

then we say $f(n) = O(n^3)$

If there exist constants $d > 0$, $n_1 > 0$ such that for all $n \geqslant n_1$:

$$f(n) \geqslant dn^3$$

then we say $f(n) = \Omega(n^3)$


f(n)
d·g(n)
f(n) = Ω(g(n))
n₁
n →

If there exist constants $c>0$, $n_0>0$ such that for all $n \geq n_0$:

$$f(n) \leq c \cdot g(n)$$

then we say $f(n) = O(g(n))$

If there exist constants $d>0$, $n_1>0$ such that for all $n \geq n_1$:

$$f(n) \geq d \cdot g(n)$$

then we say $f(n) = \Omega(g(n))$

If there exist constants $c > 0$, $n_0 > 0$ such that for all $n \geqslant n_0$:

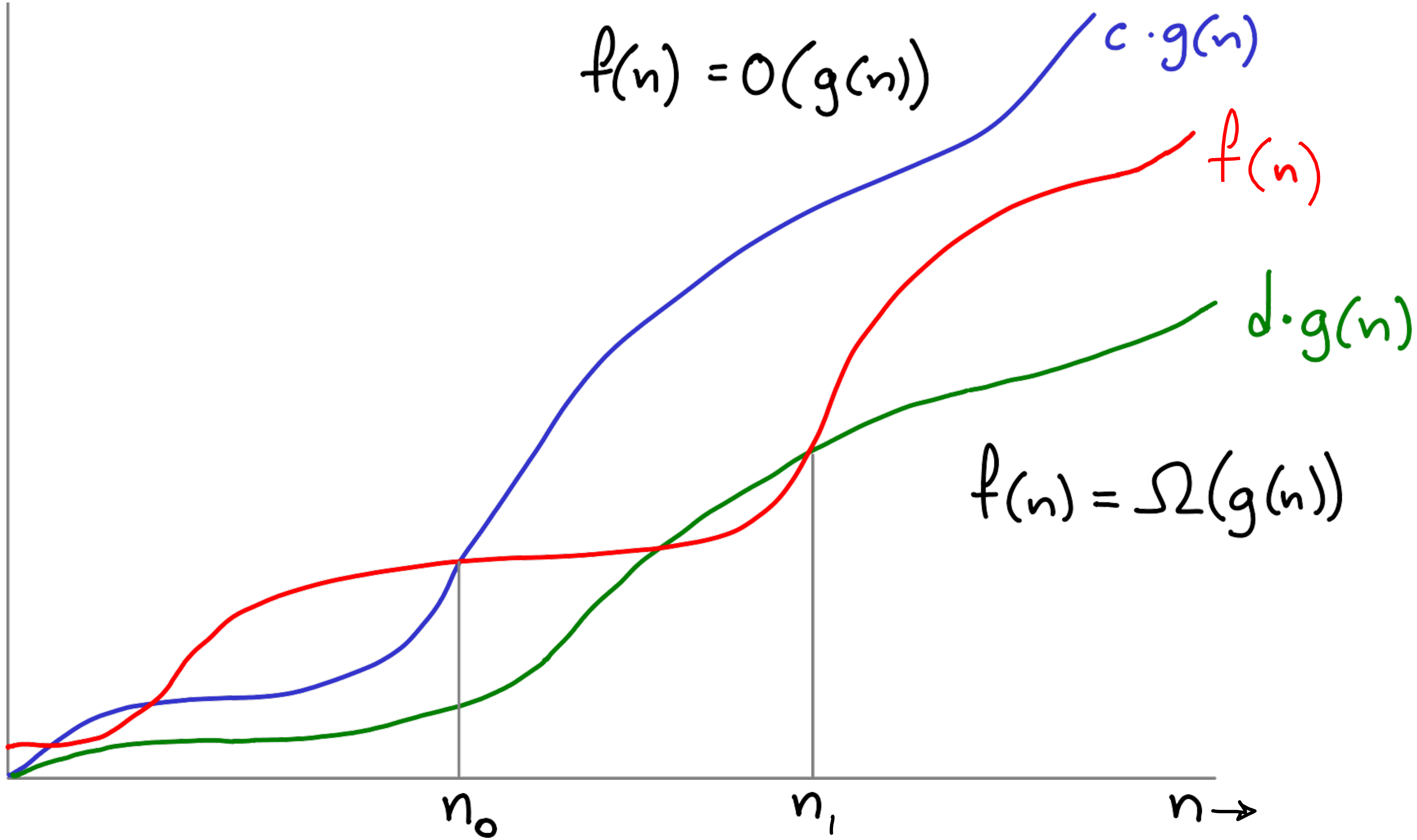
$$f(n) \leq c \cdot g(n)$$

then we say $f(n) = O(g(n))$ Big-O

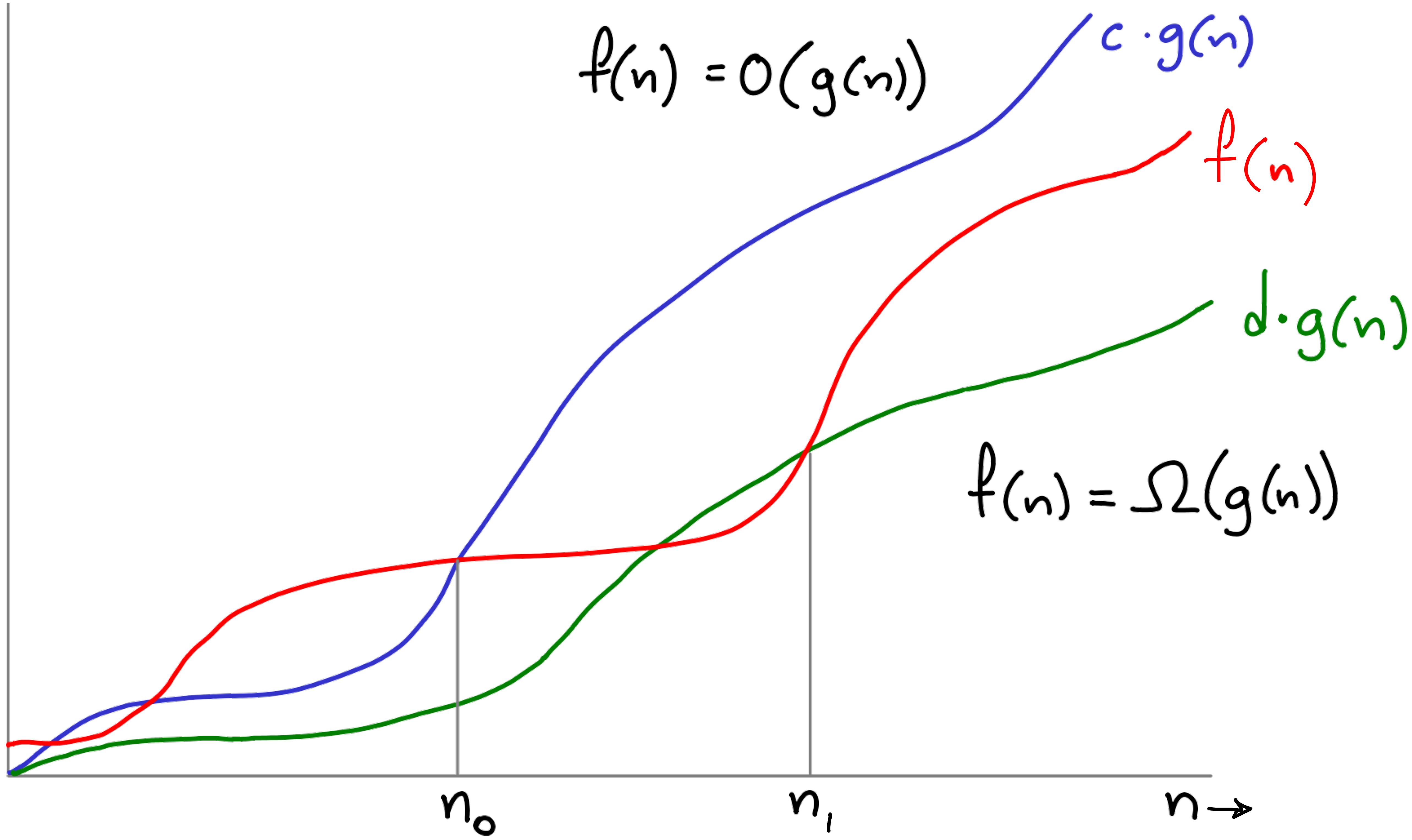
If there exist constants $d > 0$, $n_1 > 0$ such that for all $n \geqslant n_1$:

$$f(n) \geqslant d \cdot g(n)$$

then we say $f(n) = \Omega(g(n))$ Omega

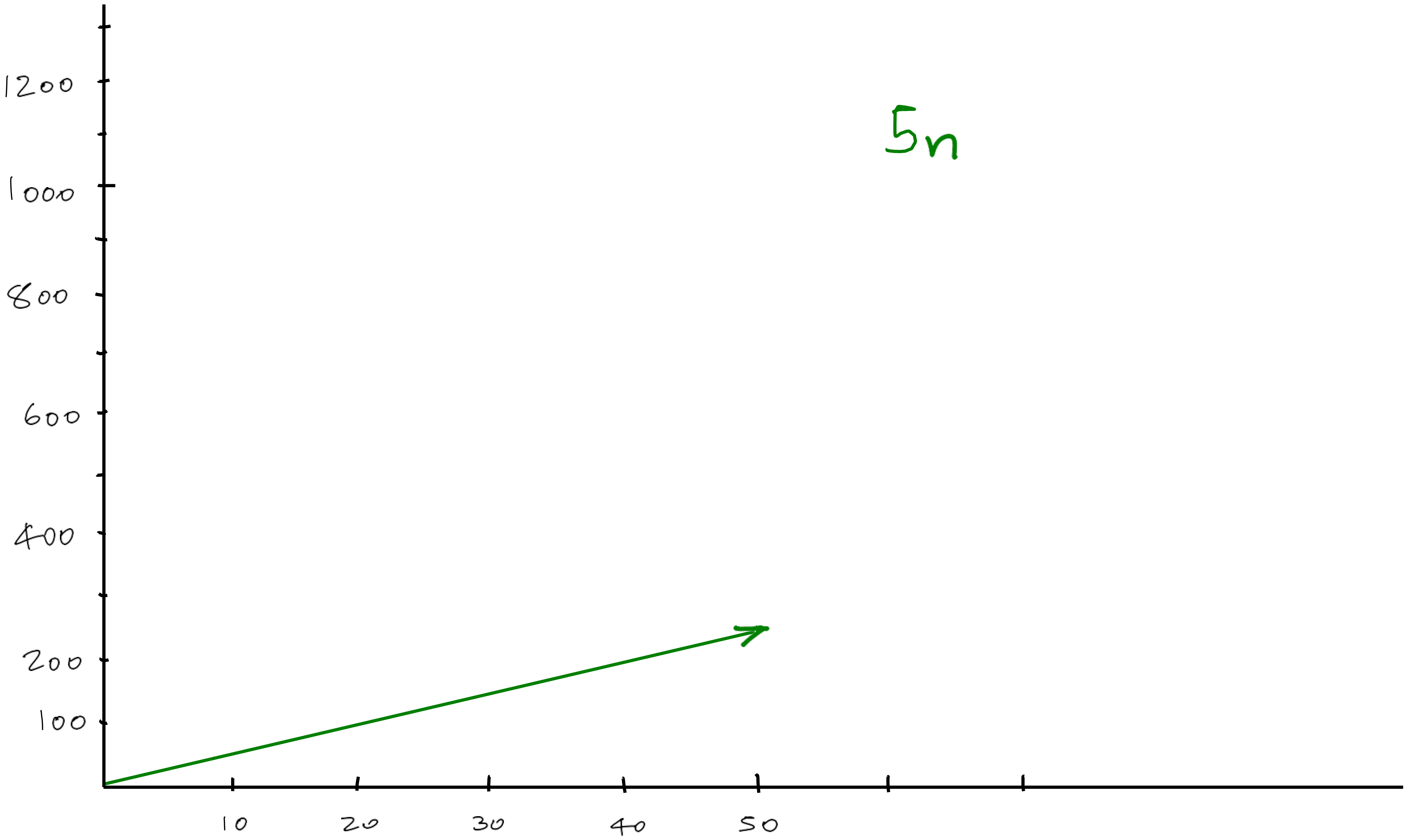

$f(n) = O(g(n))$
$c \cdot g(n)$
$f(n)$
$d \cdot g(n)$
$f(n) = \Omega(g(n))$
n_0
n_1
$n \rightarrow$

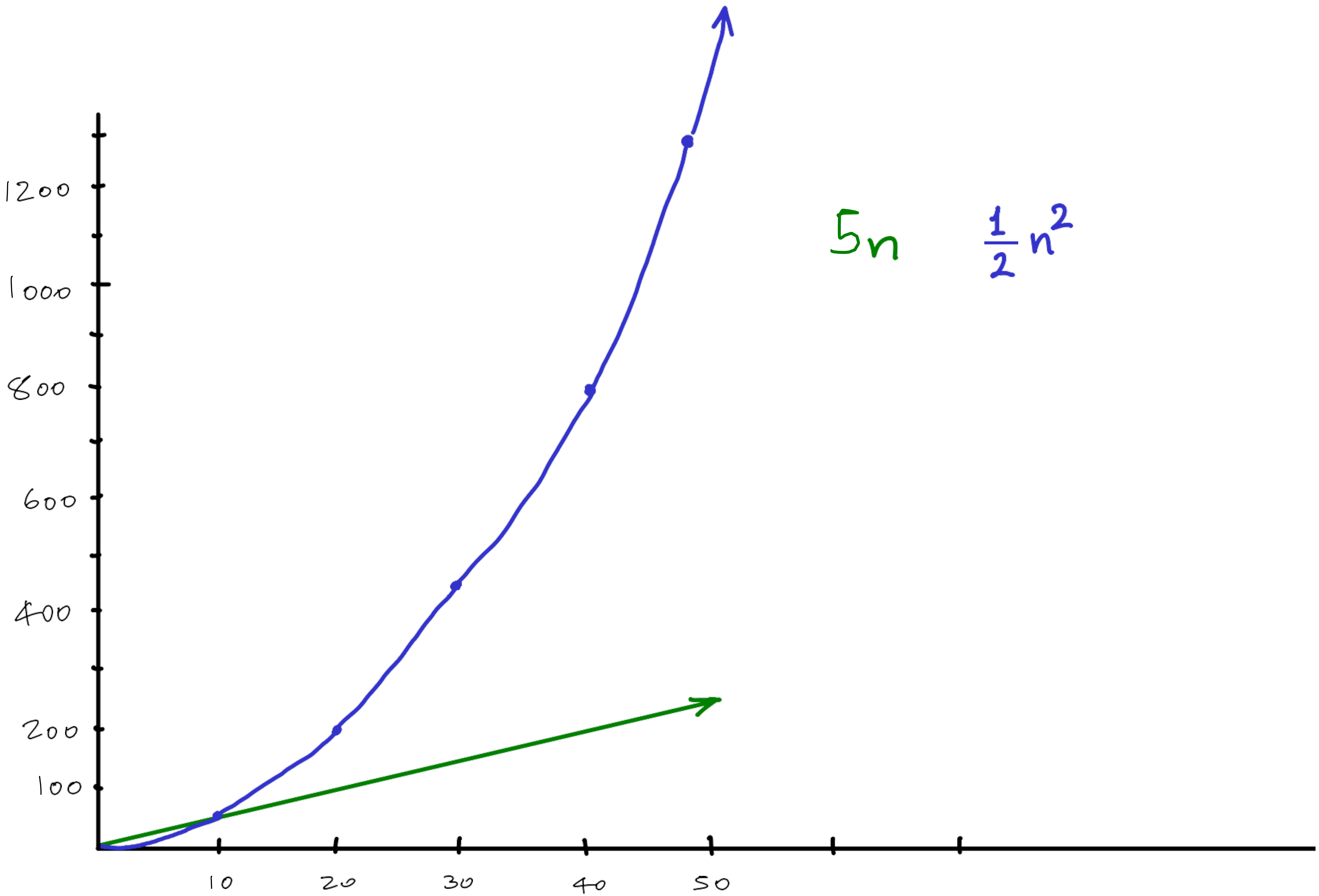
If $f(n) = O(g(n))$ AND $f(n) = \Omega(g(n))$ then $f(n) = \Theta(g(n))$ [Theta]

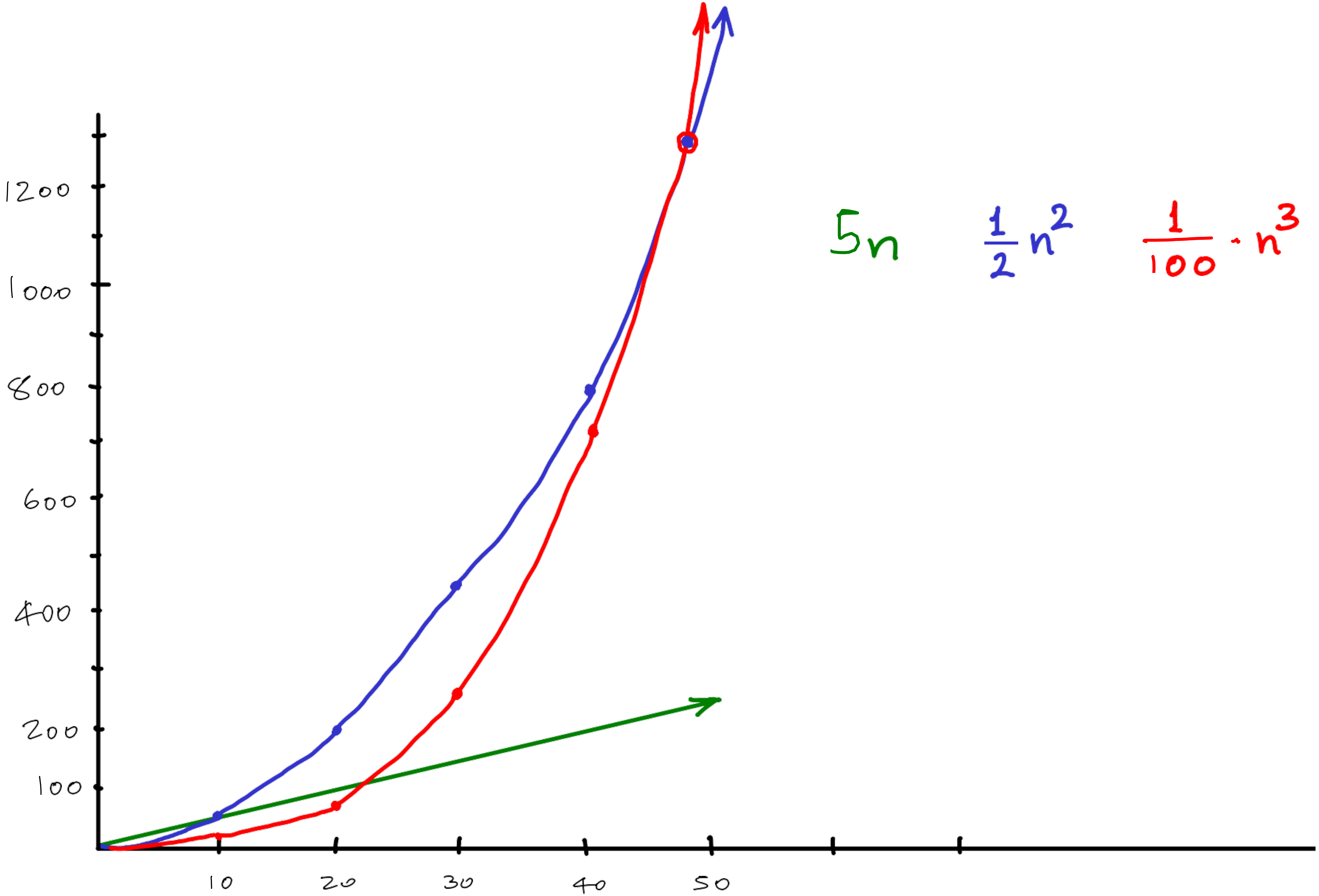


Let's compare runtimes

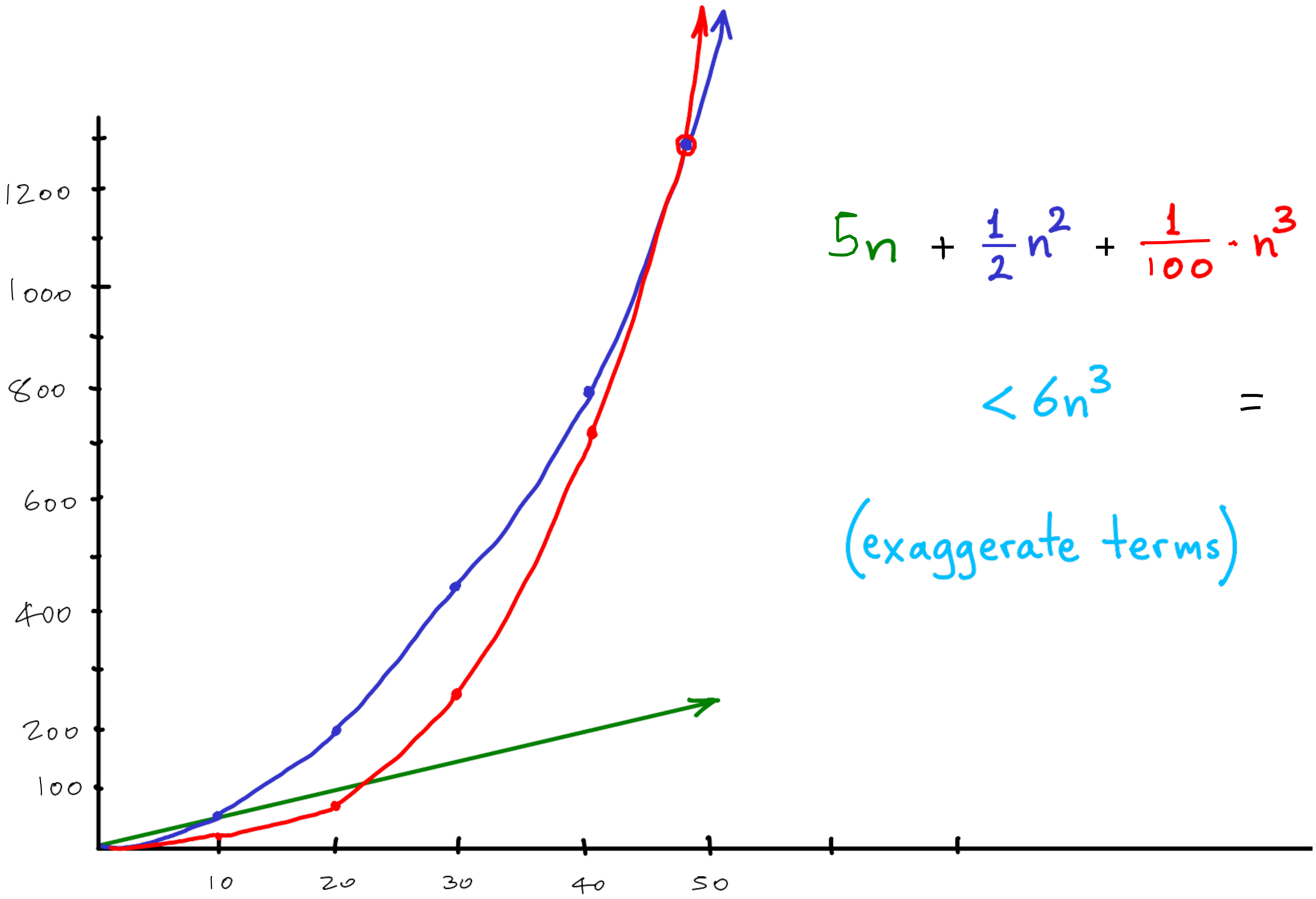
constant	*logarithmic*	*linear*	*quadratic*	*polynomial*	*exponential*
O(1)	***O(log n)***	***O(n)***	$O(n^2)$	$O(n^k)$ $(k \geq 1)$	$O(a^n)$ $(a > 1)$


5n
1200
1000
800
600
400
200
100
10
20
30
40
50


1200
1000
800
600
400
200
100
10
20
30
40
50
5n
$\frac{1}{2}n^2$



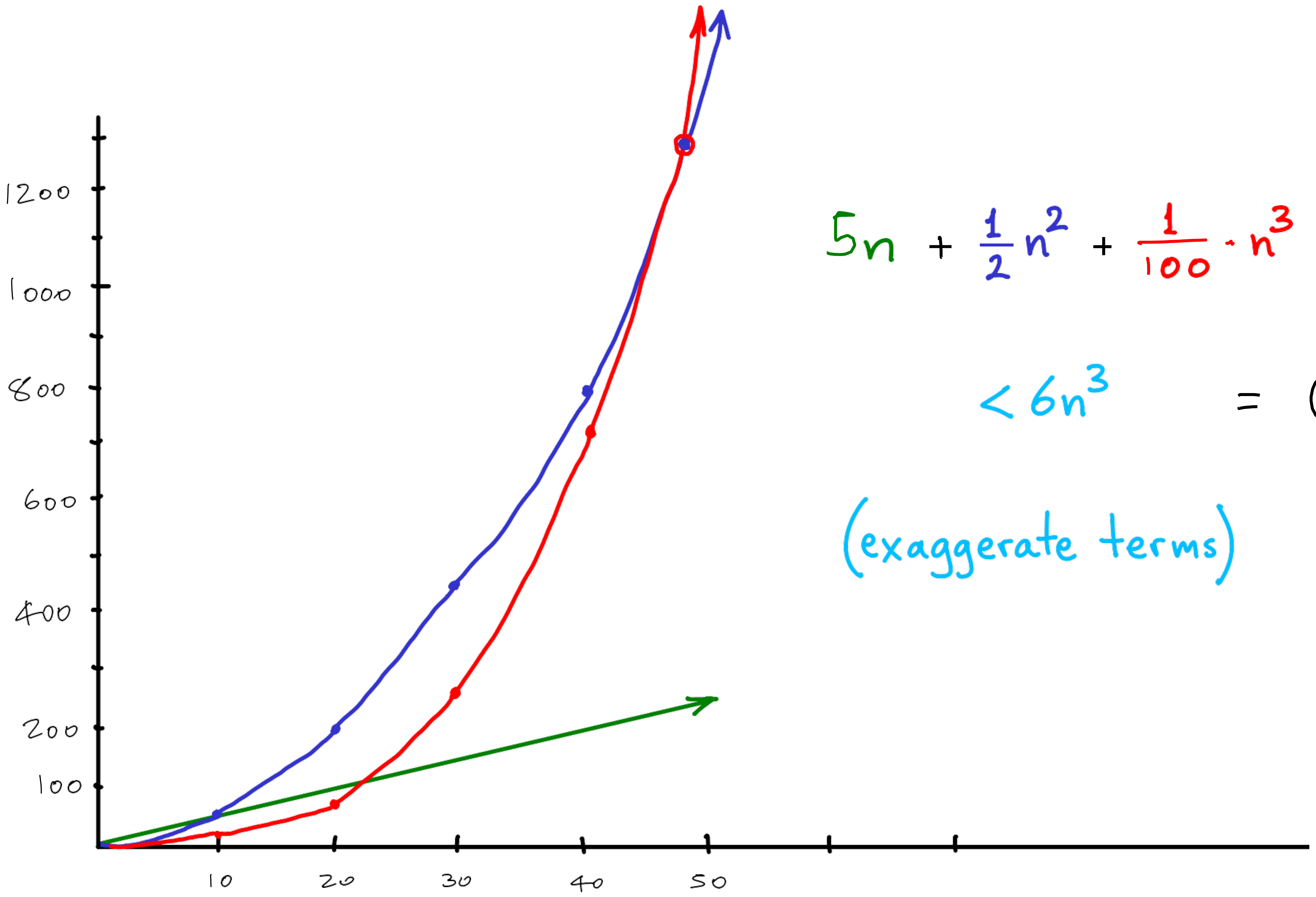
1200
1000
800
600
400
200
100
10
20
30
40
50
$5n$
$\frac{1}{2}n^2$
$\frac{1}{100} \cdot n^3$



$$5n + \frac{1}{2}n^2 + \frac{1}{100}\cdot n^3$$

$$< 6n^3 \qquad =$$

(exaggerate terms)



$$5n + \frac{1}{2}n^2 + \frac{1}{100}\cdot n^3$$

$$< 6n^3 \qquad = O(n^3)$$

(exaggerate terms)

Polynomials: $a + bn + cn^2 + dn^3 \cdots + zn^k$

$a, b, c, d, \ldots, z$: constants

Polynomials: $a + bn + cn^2 + dn^3 \cdots + \underline{zn}^k = O(\underline{n}^k)$

$a, b, c, d, \ldots, z$: constants

Also assuming one of each term (compare to: $a_1 n + a_2 n + \cdots + a_n n$)

Polynomials: $a + bn + cn^2 + dn^3 \dots + z\underline{n}^k = O(\underline{n}^k)$

$a, b, c, d, \dots, z$: constants

Also assuming one of each term (compare to: $a_1 n + a_2 n + \dots + a_n n$)

Logarithms: $50 \cdot \log n^3 + \log^{20} n + \underline{n}^{0.1} = O(\underline{n^{0.1}})$

"weaker" than polynomial

Polynomials: $a + bn + cn^2 + dn^3 \cdots + zn^k = O(n^k)$

$a, b, c, d, \ldots, z$: constants

Also assuming one of each term (compare to: $a_1 n + a_2 n + \cdots + a_n n$)

Logarithms: $50 \cdot \log n^3 + \log^{20} n + n^{0.1} = O(n^{0.1})$

"weaker" than polynomial

Exponential: $100 \cdot n^{50} + 3^n + 40 \cdot 2^n = O(3^n)$

"stronger" than polynomial

Ordering some common functions

Within each row, subdivide

n^n

$n!$

exponential: k^n $(k>1)$ — 1.1^n, 2^n, 3^n etc

polynomial: n^k — $n^{0.1}$, $\sqrt{n}$, n, $n^{1.1}$, n^2, n^3, etc

powers of logs: $\log^k n = (\log n)^k$ — $\log n$, $\log^2 n$, $\log^3 n$, etc

base doesn't matter!

constants: $1,\ 50,\ 2^{100} = O(1)$

Summary

If runtime is…

- Constant/Logarithmic runtime **Amazing**
- Linear **Great**
- Quadratic **Ok**
- Anything bigger? **We are in trouble**

Let’s Practice!!!

```
void silly(int n) {
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < i; ++j) {
            System.out.println("j = " + j);
        }
        for (int k = 0; k < n * 3; ++k) {
            System.out.println("k = " + k);
        }
    }
}
```

Runtime:

$\boxed{O(n^2)}$

```
void silly(int n, int x, int y) {
    for (int i = 0; i < n; ++i) {
        if (x < y)
            for (int k = 0; k < n * n; ++k){
                System.out.println("k = " + k);
            }
        else
            System.out.println("i = " + i);
    }
}
```

$\boxed{O(n^3)}$

```
void silly(int n) {
    if (n <= 0) return;
    System.out.println("n = " + n);
    silly(n-1);
}
```

$\boxed{O(n)}$

```
void silly(int n) {
    if (n <= 0) return;
    System.out.println("n = " + n);
    silly(n/2);
}
```

$\boxed{O(\log n)}$

$f(N) = N \log(N^2) + N$ ____________________

$f(N) = 20000^2 + N \log N + N$ ____________________

$f(N) = 100N + N \log N + N/2$ ____________________

$f(N) = N \cdot (\log (N^4) - \log N) + N^2$ ____________________

$f(N) = N^2 \cdot \log_2 (N/2) + N^2$ ____________________