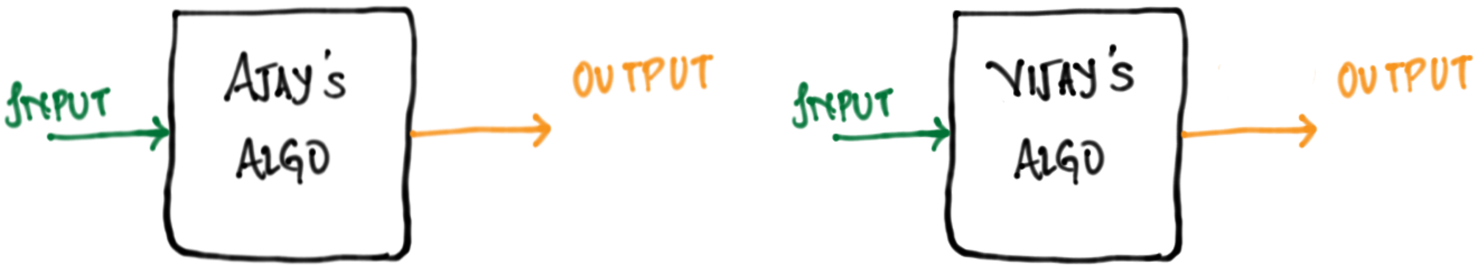
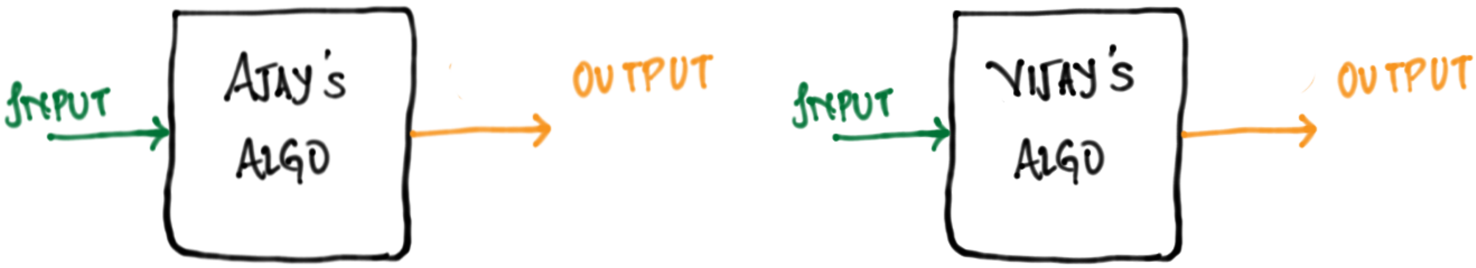
Problem Definition : Description of the problem

Problem Definition : Description of the problem



Q: Which algorithm will you choose: Ajay's or Vijay's ?

Problem Definition : Description of the problem



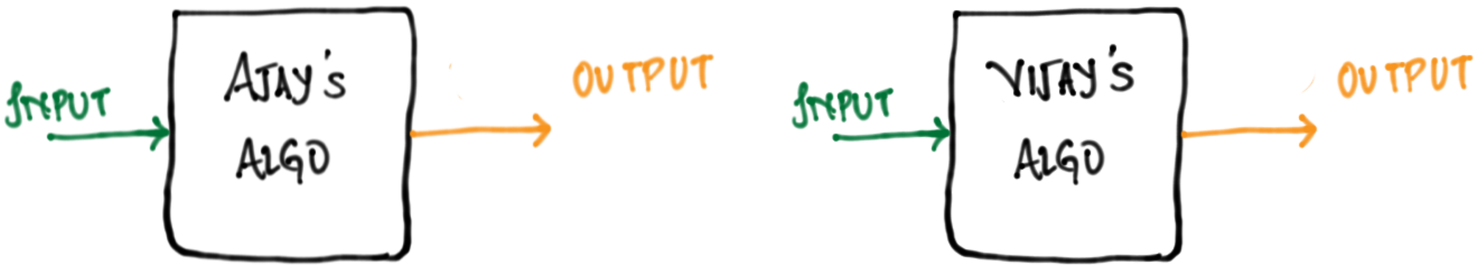
Q: Which algorithm will you choose : Ajay's or Vijay's ?

A : Correct : the algorithm should give the correct output for every valid input.

Running Time : the algorithm should be fast

Maintainence : The algorithm should be easy to read.

Problem Definition : Description of the problem



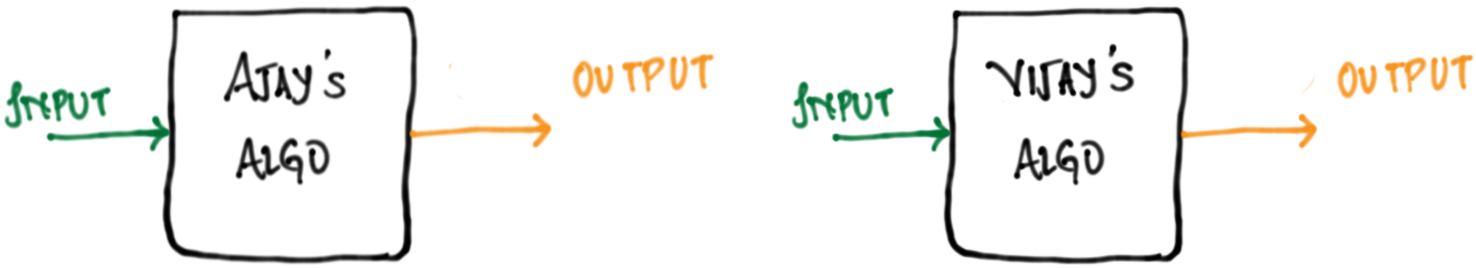
Q: Which algorithm will you choose: Ajay's or Vijay's ?

A :

In this course

- Correct : the algorithm should give the correct output for every valid input.
- Running Time : the algorithm should be fast
- Maintainence : The algorithm should be easy to read.

Problem Definition : Description of the problem



Q: Which algorithm will you choose : Ajay's or Vijay's ?

A : Correct : the algorithm should give the correct output for every valid input.

Initially we will focus on running time → Running Time : the algorithm should be fast

Maintainence : The algorithm should be easy to read.

Input A[1...n]

```
min ← A[1] ;
for  i ← 2 to n
{
      if ( A[i] < min )
          min ← A[i] ;
}
return min ;
```

What is this program doing ?

Input A[1...n]

```
min ← A[1] ;
for i ← 2 to n
{
    if (A[i] < min)
        min ← A[i] ;
}
return min ;
```

What is this program doing ?

Finding the Minimum.

```
Input A[1...n]

min ← A[1] ;
for i ← 2 to n
{
        if (A[i] < min)
             min ← A[i] ;
}
return min ;
```

What is this program doing?

Finding the Minimum.

Q: What is the running time of this algorithm?

Input A[1...n]

```
min ← A[1] ;
for i←2 to n
{
      if (A[i] < min)
            min ← A[i] ;
}
return min ;
```

What is this program doing?

Finding the Minimum.

Q: What is the running time of this algorithm?

Q: What do you mean by running time?

Input A[1...n]

```
min ← A[1] ;
for i ← 2 to n
{
    if ( A[i] < min)
        min ← A[i] ;
}
return min ;
```

What is this program doing ?

Finding the Minimum.

Q: What is the running time of this algorithm ?

Q: What do you mean by running time?

<u>Good Input</u>

1 2 3 4 5 6

<u>Bad Input</u>

6 5 4 3 2 1

Q: What do you mean by running time?

A: Compute the worst case running time (on the worst case input).

Q: What do you mean by running time?

A: Compute the worst case running time (on the worst case input).

Running Time : ($n = 10^6$, hypothetical example)

1950	1980	2010
1ms	1us	1ns

Q: What do you mean by running time?

A: Compute the worst case running time (on the worst case input).

Running Time : ($n = 10^6$, hypothetical example)

1950	1980	2010
1ms	1us	1ns

<u>Problem</u> : These running time are machine dependent.

Can we find out a running time that is machine independent?

Input A[1...n]

```
min ← A[1] ;                          C1
for  i ← 2 to n                       C2
{
      if ( A[i] < min)
           min ← A[i] ;
}
return min ;
```

Input A[1...n]

```
min ← A[1] ;                          C1
for  i ← 2 to n                       C2(n-1)
{
      if ( A[i] < min)
            min ← A[i] ;
}
return min ;
```

Input A[1 ... n]

```
min ← A[1] ;                    c1
for  i ← 2 to n                 c2(n-1)
{
      if (A[i] < min)           c3(n-1)
          min ← A[i] ;          c4(n-1)
}
return min ;                    c5
```

Running Time $= c_1 + (c_2 + c_3 + c_4)(n-1) + c_5$

Input $A[1 \dots n]$

```
min ← A[1] ;                      c1
for  i ← 2  to  n                 c2(n-1)
{
      if ( A[i] < min )           c3(n-1)
          min ← A[i] ;            c4(n-1)
}
return min ;                      c5
```

$$\text{Running Time} = c_1 + (c_2 + c_3 + c_4)(n-1) + c_5$$

$$= 2c + 3c(n-1) \quad (\text{each } c_i = c)$$

$$= (2 + 3(n-1))c$$

$$= (3n-1)c$$

Input $A[1 \ldots n]$

```
min ← A[1];                    c1
for  i ← 2 to n                c2(n-1)
{
      if (A[i] < min)          c3(n-1)
          min ← A[i];          c4(n-1)
}
return min;                    c5
```

$$\text{Running Time} = c_1 + (c_2 + c_3 + c_4)(n-1) + c_5$$
$$= 2c + 3c(n-1) \qquad (\text{each } c_i = c)$$
$$= (2 + 3(n-1))c$$
$$= (3n-1)c$$

Tells us that as n increases the running time of our algorithm increases

Machine dependent constant

Since we wanted a machine independent running time, we say that

Running Time $\propto (3n-1)$

Since we wanted a machine independent running time, we say that

Running Time $\propto$ $(3n-1)$

Running Time $= (3n-1)$

Compare two algorithms.

A

$f(n) = 2n^2+5$

B

$g(n) = 50n + 5$

Q: Which algorithm is better?

Compare two algorithms.

A

$f(n) = 2n^2+5$

B

$g(n) = 50n+5$

Q: Which algorithm is better?

(Case 1) $n \leq 25$

$f(25) = g(25) = 1225$

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

Compare two algorithms.

A

$f(n) = 2n^2+5$

B

$g(n) = 50n+5$

Q: Which algorithm is better ?

(Case 1) $n \leq 25$

$f(25) = g(25) = 1225$

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

Compare two algorithms.

A

$f(n) = 2n^2 + 5$

B

$g(n) = 50n + 5$

Q: Which algorithm is better?

(Case 1) $n \leq 25$

$f(25) = g(25) = 1225$

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

(case 2) $n > 25$

$f(n) > g(n)$

Compare two algorithms.

A

$f(n) = 2n^2+5$

B

$g(n) = 50n + 5$

Q: Which algorithm is better?

(Case 1) $n \leq 25$

$f(25) = g(25) = 1225$

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

(case 2) $n > 25$

$f(n) > g(n)$

$$\lim_{n \to \infty} \frac{g(n)}{f(n)} = \lim_{n \to \infty} \frac{50n+5}{2n^2+5} = 0$$

<u>Observation</u> :

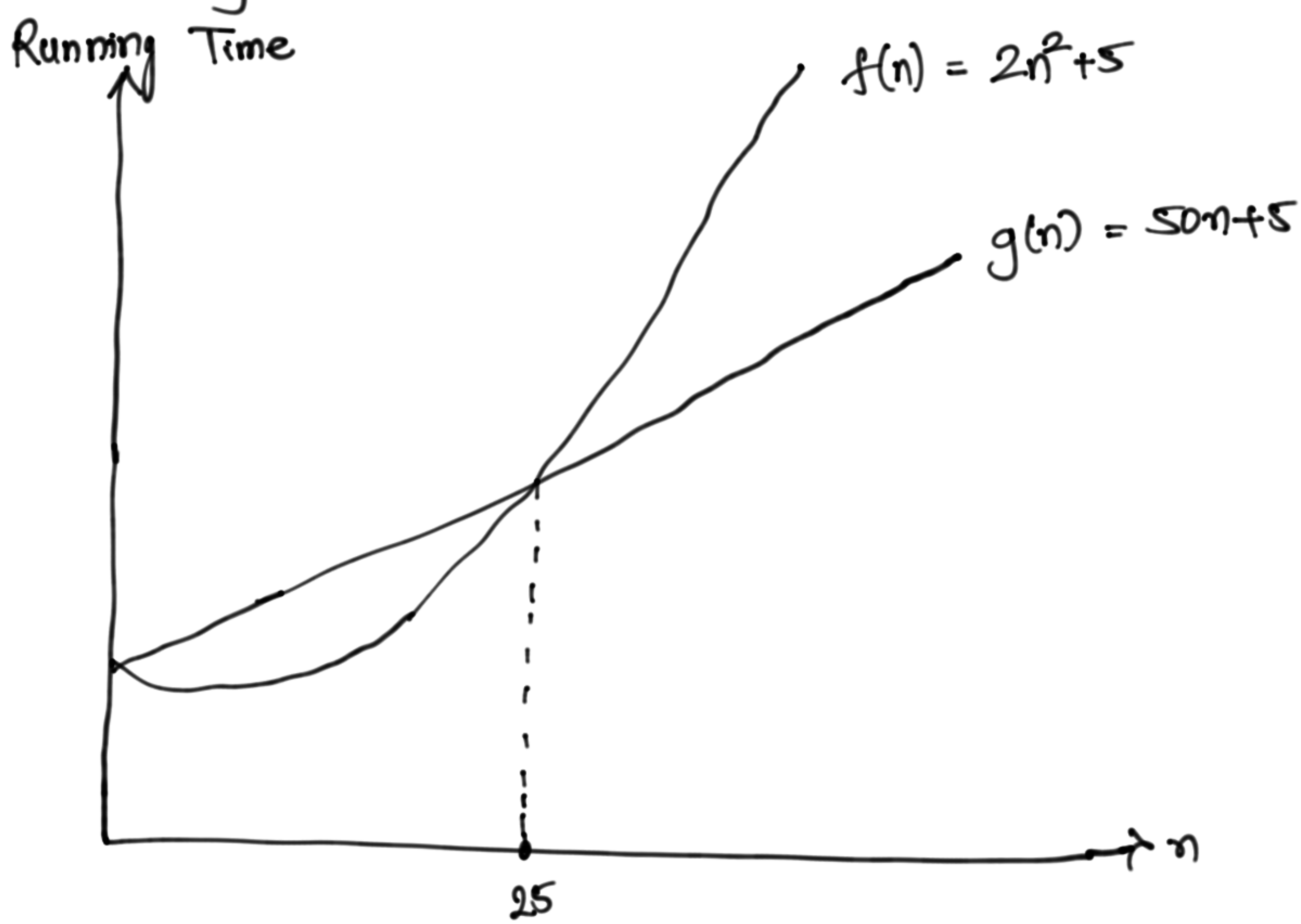
For higher values of n , $g(n) \lll f(n)$

Observation :

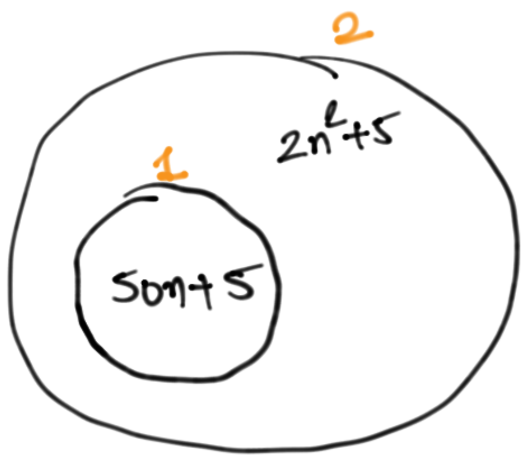
For higher values of n , $\underbrace{g(n) \lll f(n)}$

$$\lim_{n \to \infty} \frac{g(n)}{f(n)} = 0$$

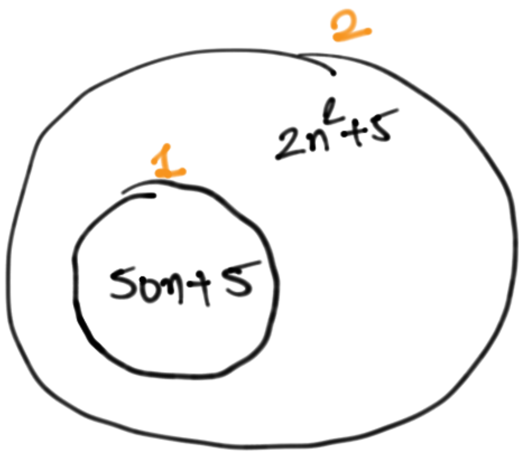
Pictorially



Running Time Classes

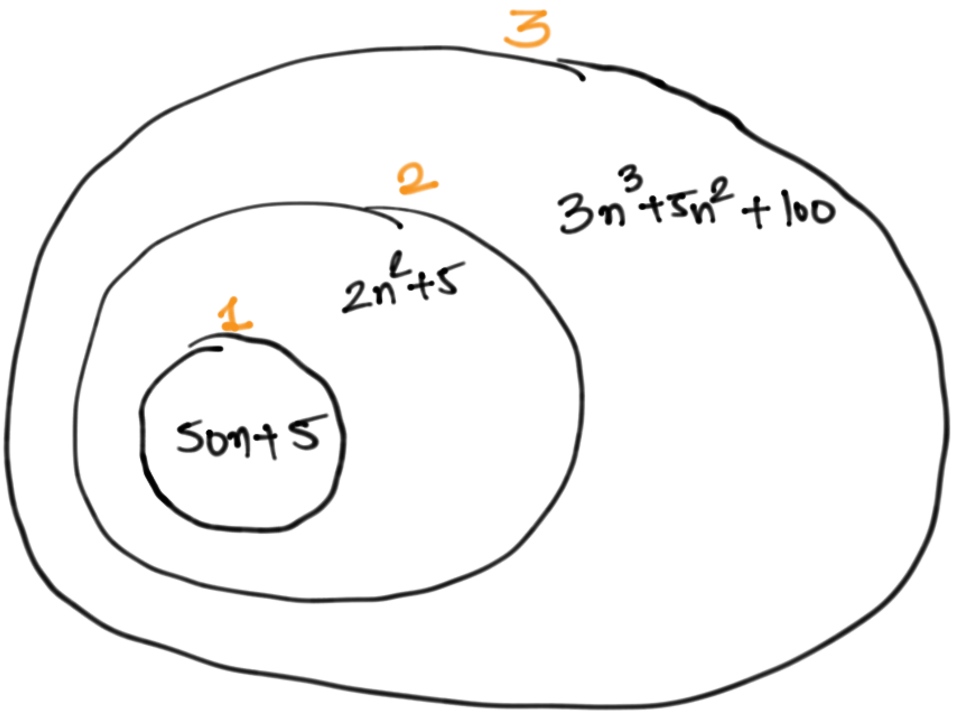


Running Time Classes



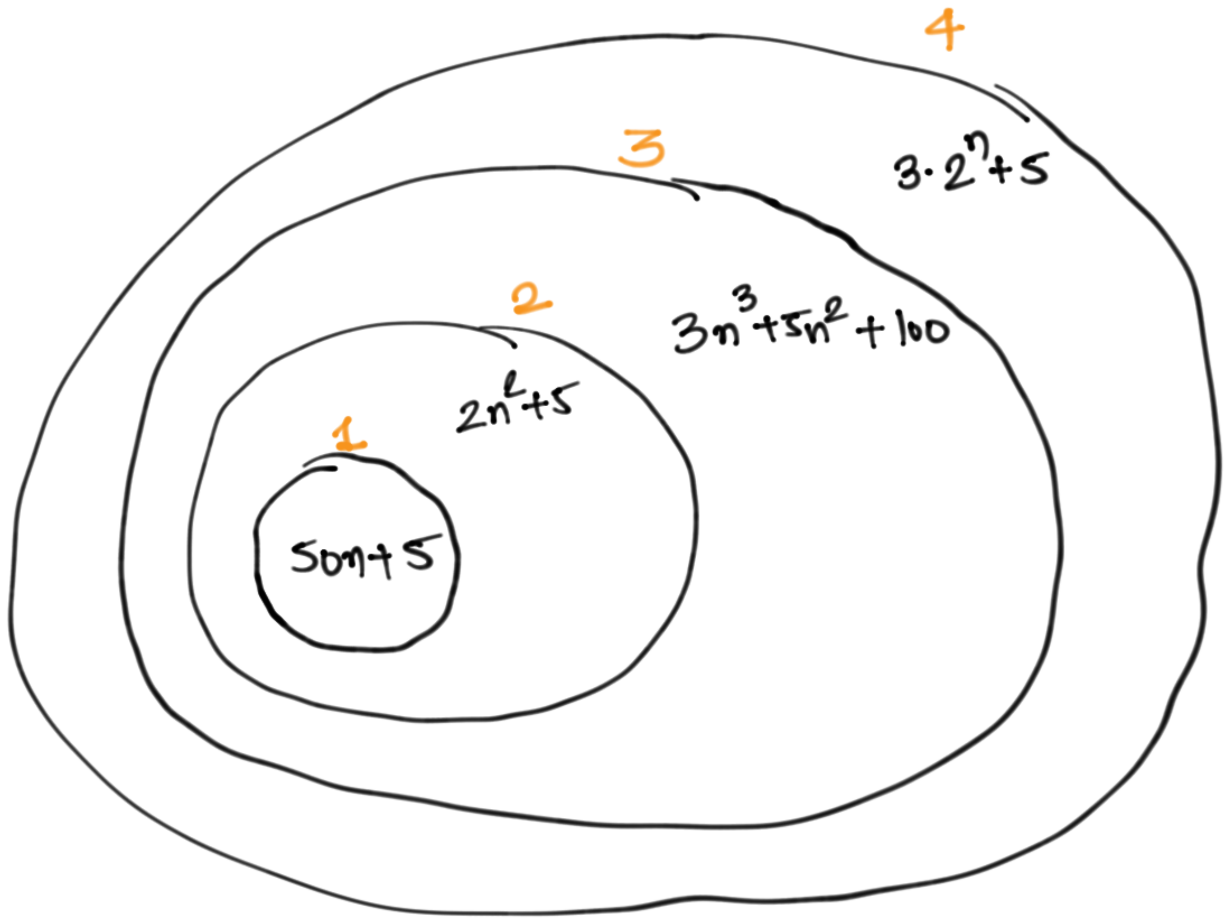
$3n^3+5n^2+100$

Running Time Classes

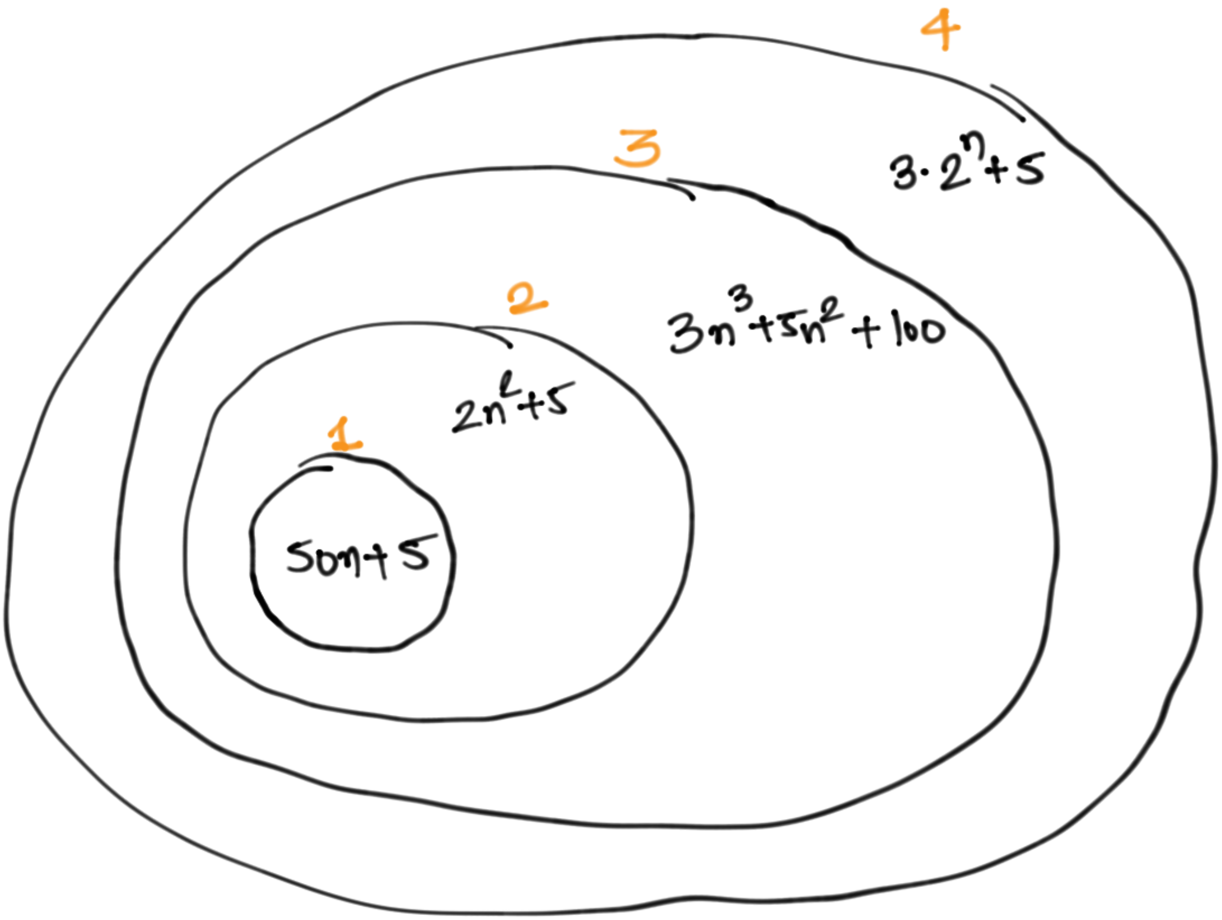


$3n^3+5n^2+100$

Running Time Classes

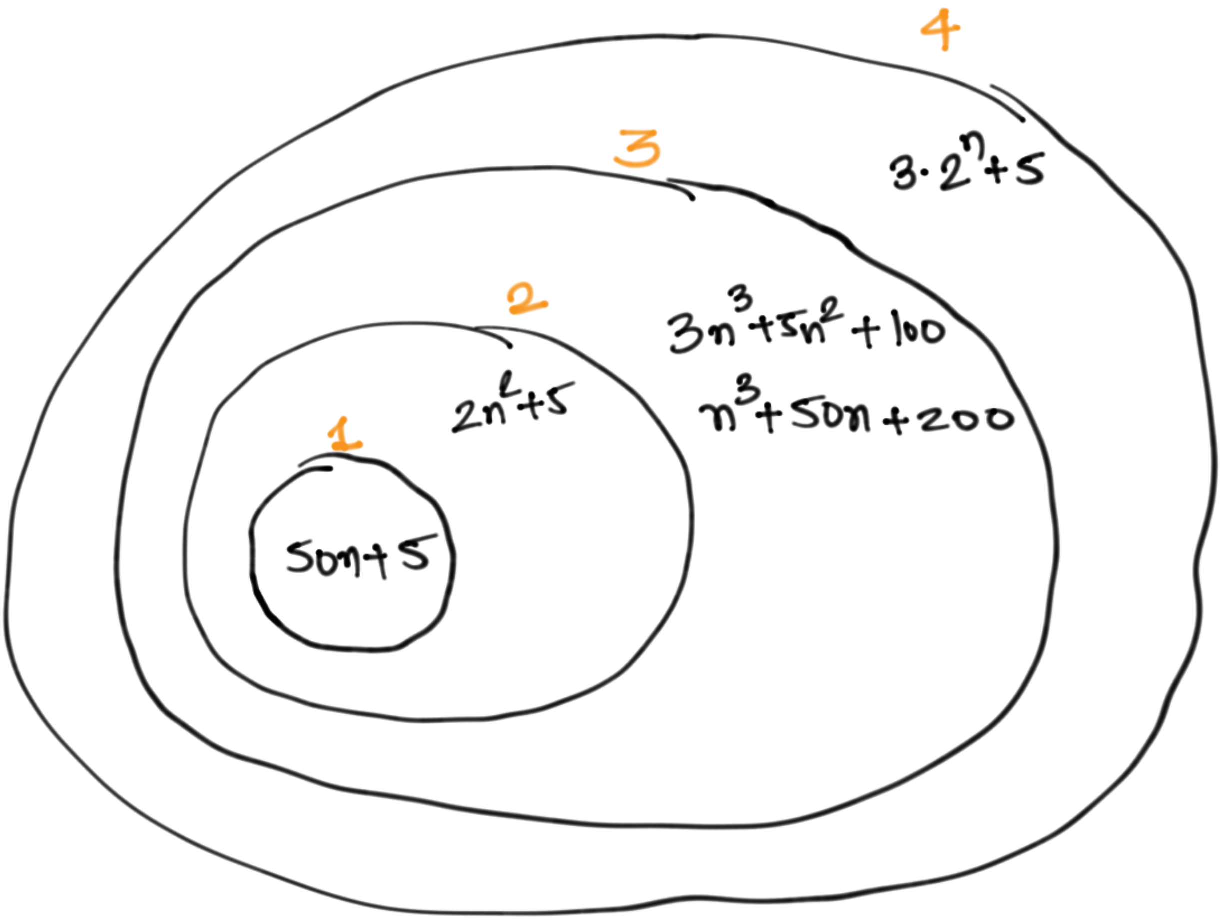


Running Time Classes

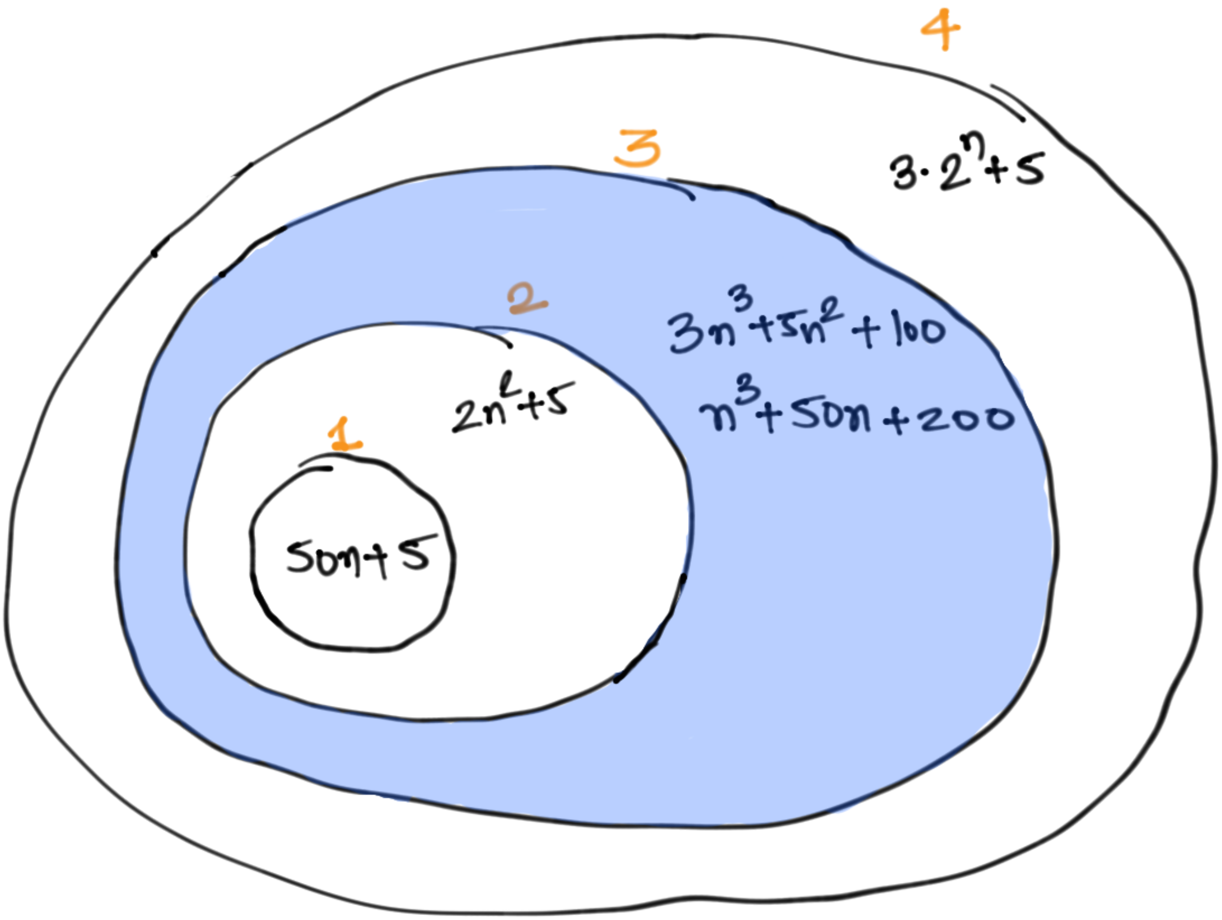


$n^3 + 50n + 200$

Running Time Classes

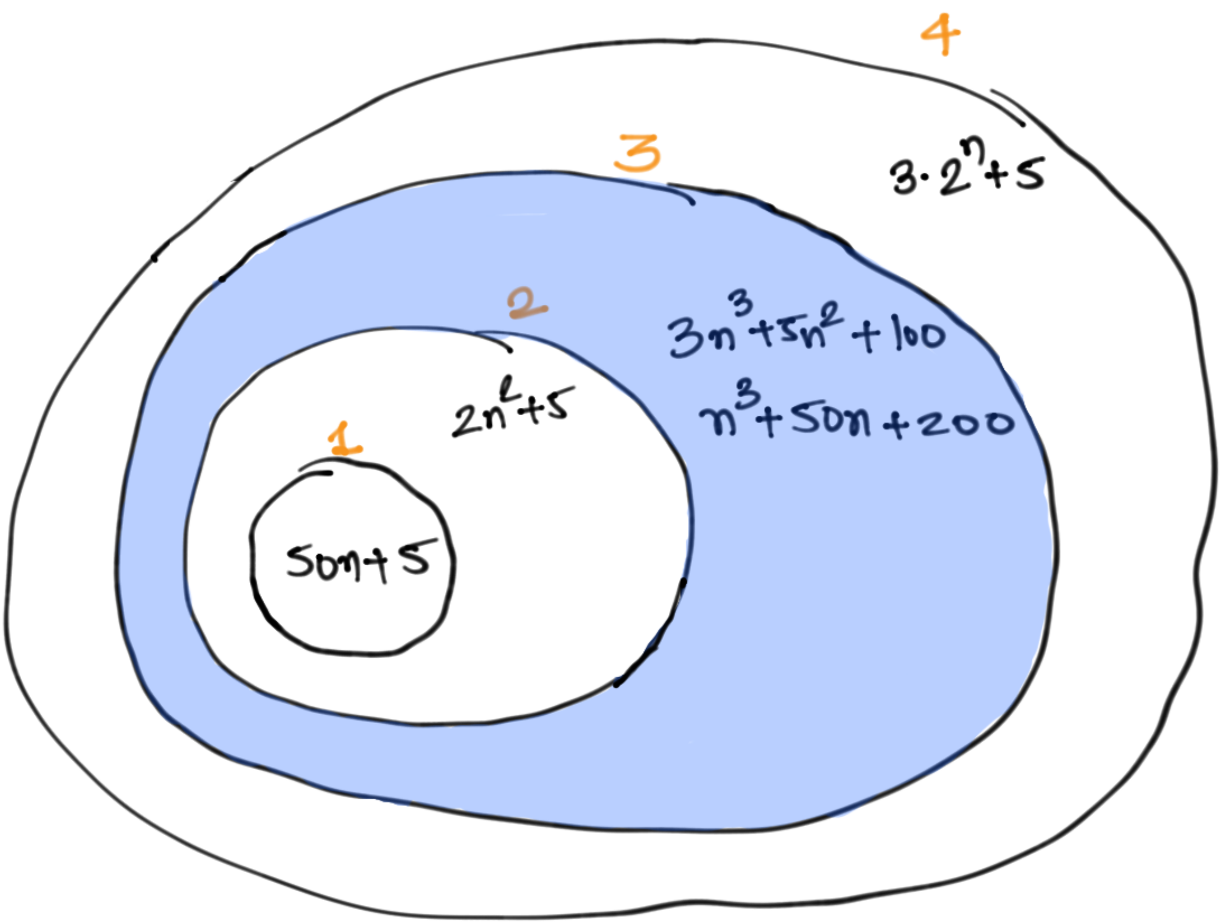


Running Time Classes



Q: Can you identify a unique property of running times lying in the same runtime class?

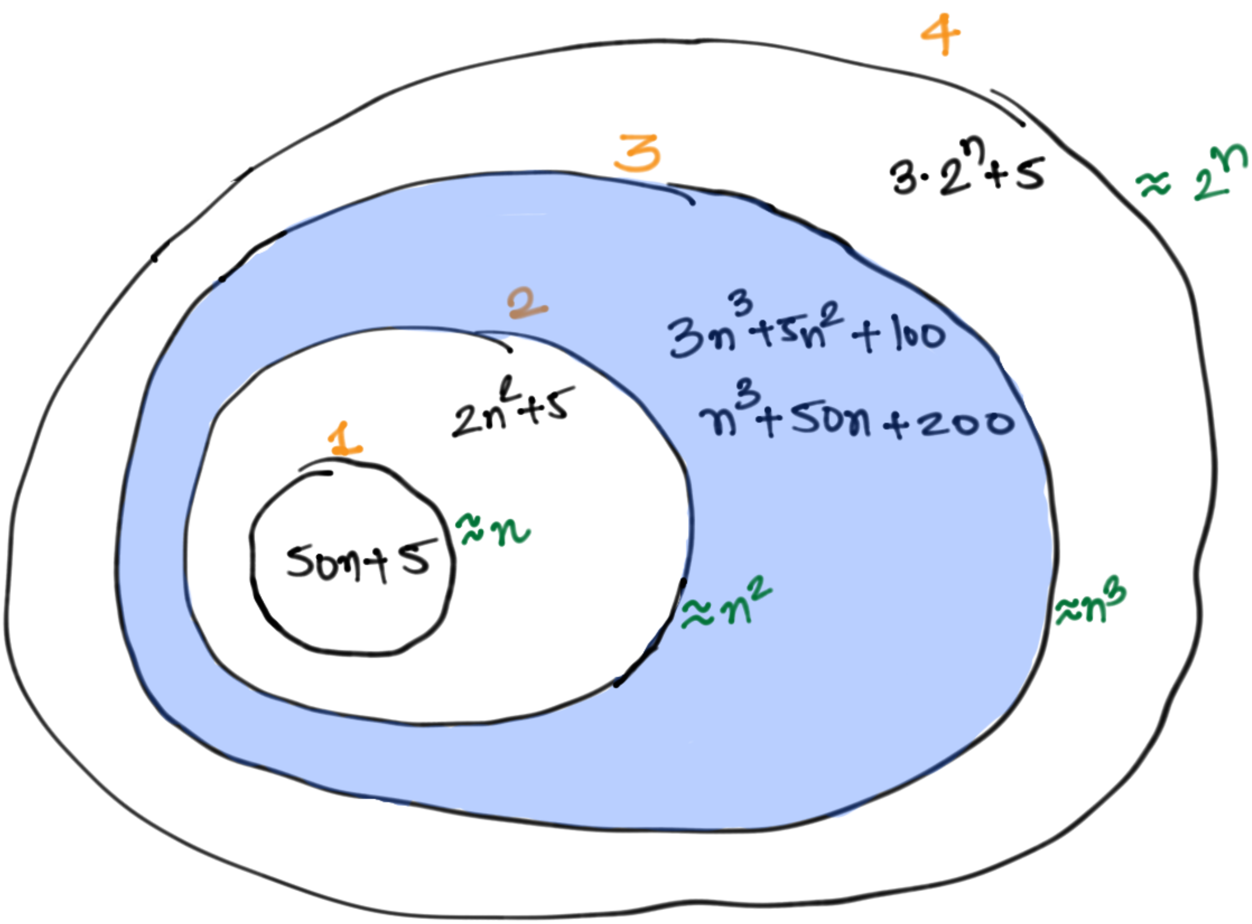
Running Time Classes



Q: Can you identify a unique property of running times lying in the same runtime class?

A: The highest order term is same

Running Time Classes



Q: Can you identify a unique property of running times lying in the same runtime class?

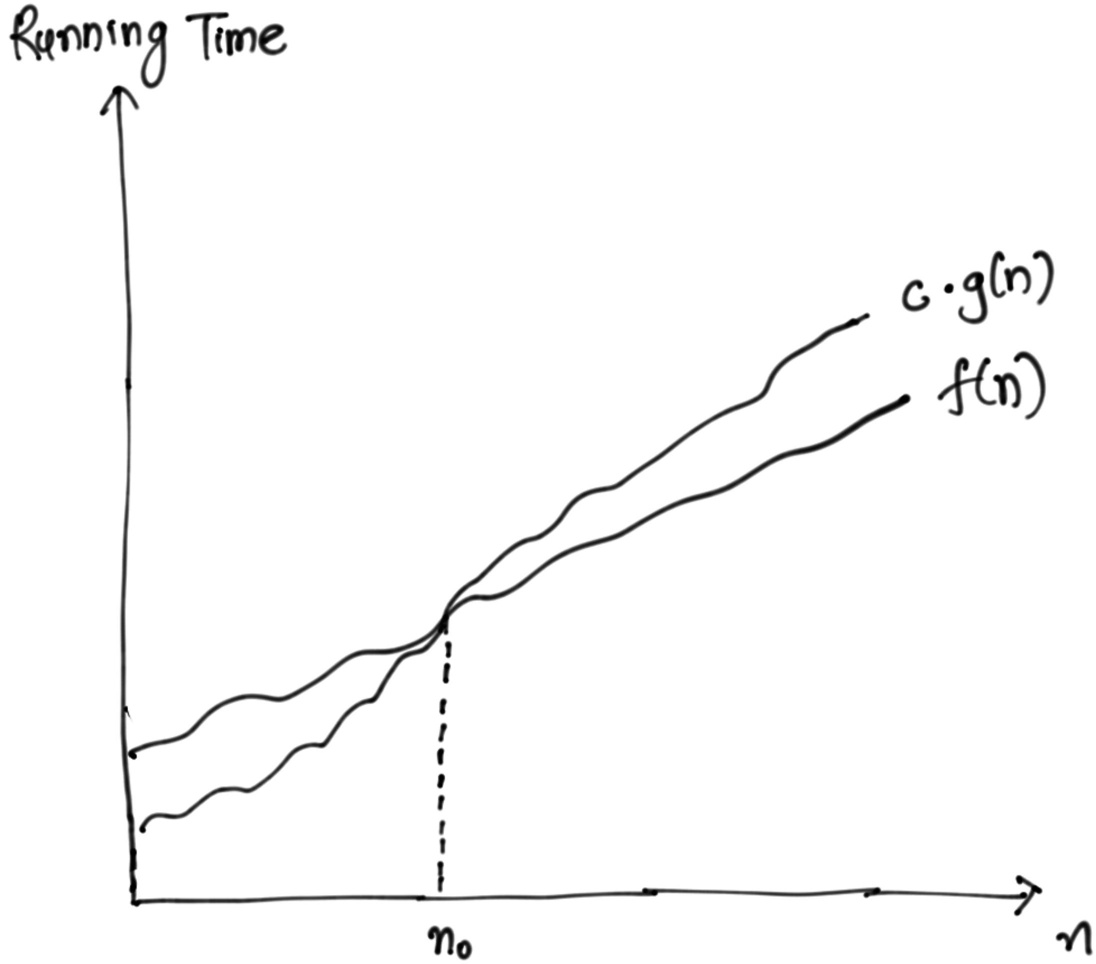
A: The highest order term is same

Def^n: Let $f(n)$ and $g(n)$ be two monotonically increasing function. Then $f(n) = O(g(n))$ if $\exists\ c \geq 0$ s.t for all $n \geq n_0$

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

Def^n: Let $f(n)$ and $g(n)$ be two monotonically increasing function. Then $f(n) = O(g(n))$ if $\exists\ c \geq 0$ s.t for all $n \geq n_0$

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

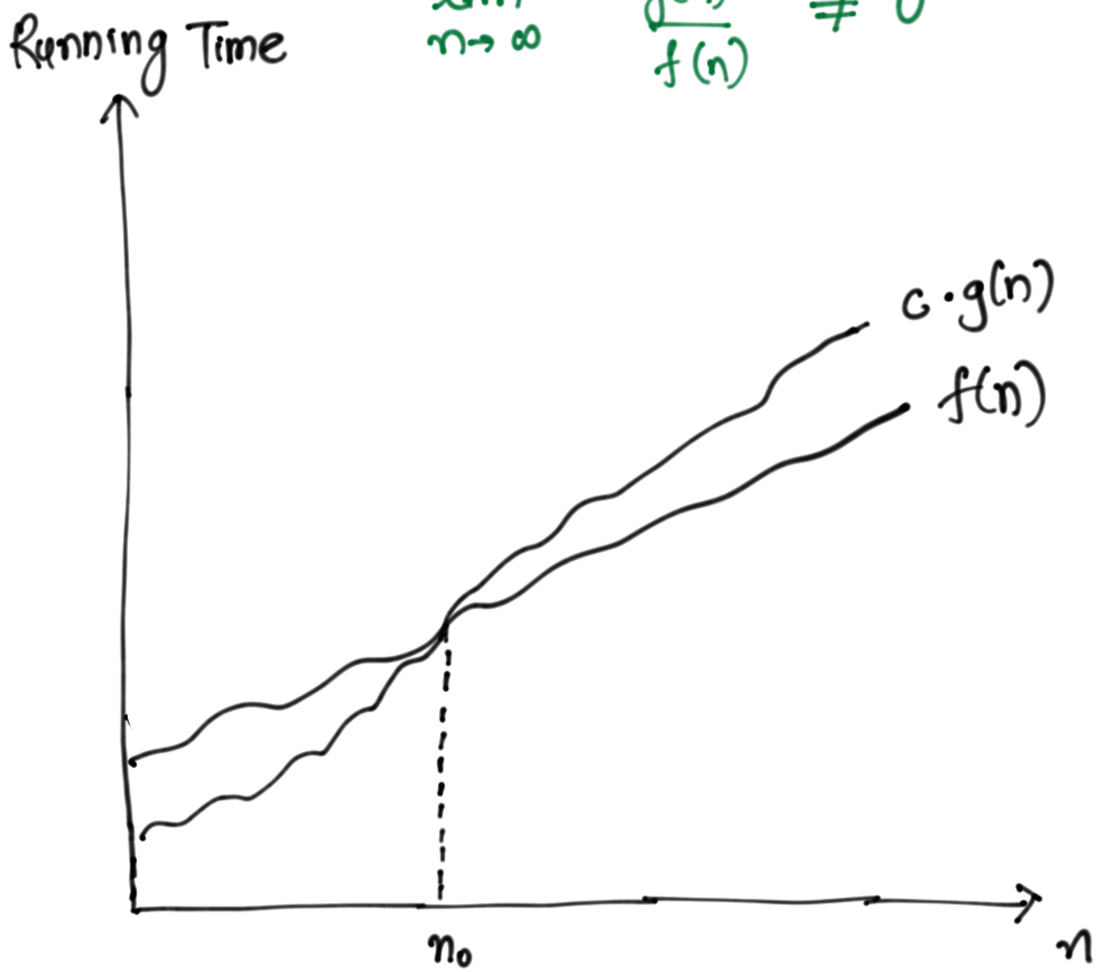


Def^n: Let $f(n)$ and $g(n)$ be two monotonically increasing function. Then $f(n) = O(g(n))$ if $\exists\ c \geq 0$ s.t for all $n \geq n_0$

$$\underbrace{f(n) \leq c \cdot g(n)}$$

$$\Downarrow$$

$$\lim_{n \to \infty} \frac{g(n)}{f(n)} \neq 0$$



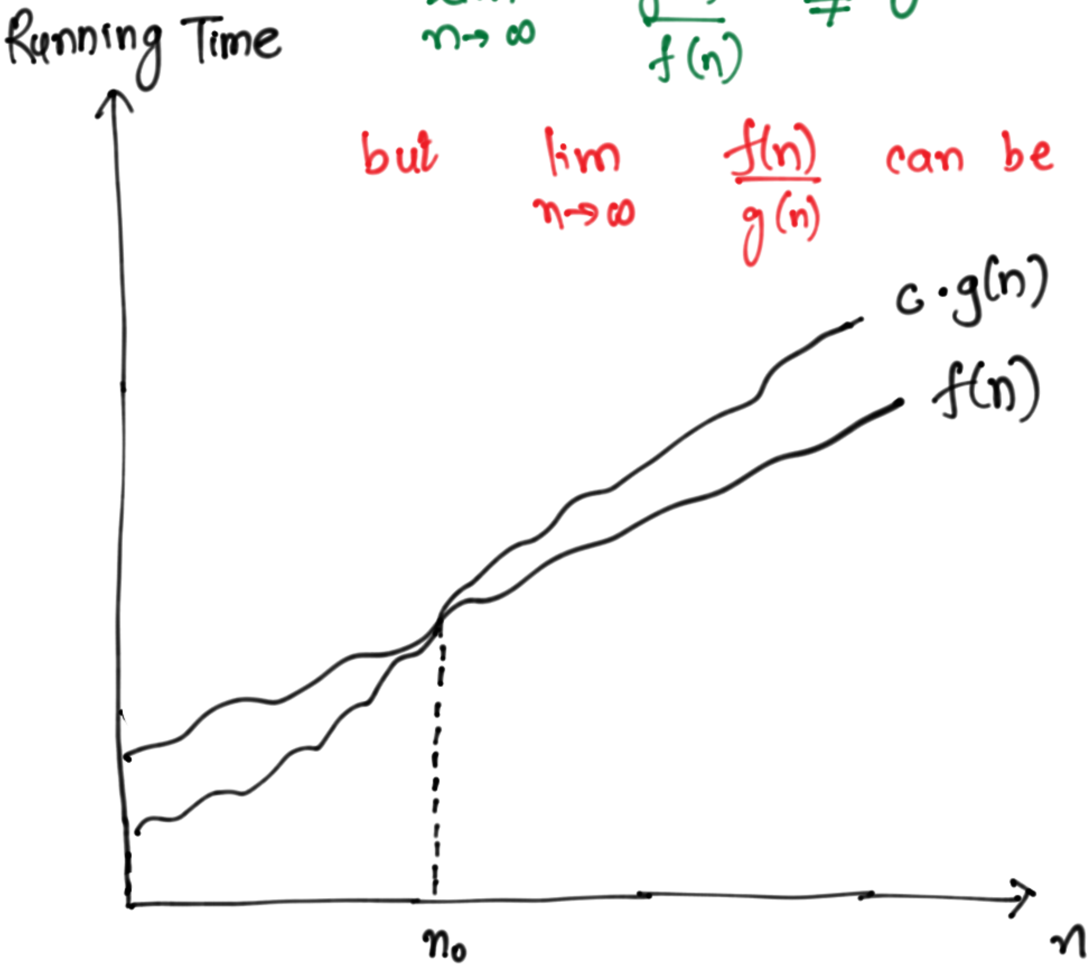
Def^n: Let $f(n)$ and $g(n)$ be two monotonically increasing function. Then $f(n) = O(g(n))$ if $\exists\ c \geq 0$ s.t for all $n \geq n_0$

$$\underbrace{f(n) \leq c \cdot g(n)}$$

$\Downarrow$

$$\lim_{n \to \infty} \frac{g(n)}{f(n)} \neq 0$$

but $\lim_{n \to \infty} \frac{f(n)}{g(n)}$ can be 0.



Some examples

(1) $20n^2 = O(n^2)$

$f(n) = 20n^2 \qquad g(n) = n^2$

Find c & n_0

Some examples

(1) Is $20n^2 = O(n^2)$

$f(n) = 20n^2$ $\quad g(n) = n^2$

Find c & n_0

$c = 20$ & $n_0 = 1$

(2) Is $20n^2 + 20n + 20 = O(n^3)$

$f(n) = 20n^2 + 20n + 20$

$g(n) = n^3$

Find c & n_0

Some examples

(1) Is $20n^2 = O(n^2)$

$f(n) = 20n^2$ $\quad g(n) = n^2$

Find c & n_0

$c = 20$ & $n_0 = 1$

(2) Is $20n^2 + 20n + 20 = O(n^3)$

$f(n) = 20n^2 + 20n + 20$

$g(n) = n^3$

Find c & n_0

$c = 60$ & $n_0 = 1$

(3) Is $20 = O(1)$

$f(n) = 20$

$g(n) = 1$

Find c & n_0

(5) Is $n \log n = O(n)$

$f(n) = n \log n$ & $g(n) = n$.

(5) Is $n \log n = O(n)$

$f(n) = n \log n$ & $g(n) = n$.

$f(n) \leq c\, g(n) \quad \forall\, n \geq n_0$

$\Rightarrow \dfrac{f(n)}{g(n)} \leq c \quad \forall\, n \geq n_0$

$\Rightarrow \dfrac{n \log n}{n} \leq c \quad \forall\, n \geq n_0$

$\Rightarrow \log n \leq c \quad \forall\, n \geq n_0$

$\downarrow$

A contradiction.

$$f(n) = 100n$$
$$g(n) = n \log n.$$

Which one is better?

$f(n) = 100n$

$g(n) = n \log n.$

Which one is better?

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} = \lim_{n \to \infty} \frac{100\cancel{n}}{\cancel{n} \log n} = 0$$

$\Rightarrow$ $f(n) \lll g(n)$ for higher values of n.

Theory Vs Practice

$$f(n) = 100n$$
$$g(n) = n\log n.$$

Which one is better?

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{100\cancel{n}}{\cancel{n}\log n} = 0$$

$\Rightarrow$ $f(n) \lll g(n)$ for higher values of n.

But how much higher is this higher value?

$$f(n) < g(n)$$
$$100\cancel{n} < \cancel{n}\log n$$

$$n > 2^{100}$$

Theory Vs Practice

$$f(n) = 100n$$
$$g(n) = n \log n.$$

Which one is better?

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{100\cancel{n}}{\cancel{n}\log n} = 0$$

$\Rightarrow$ $f(n) \lll g(n)$ for higher values of n.

But how much higher is this higher value?

$$f(n) < g(n)$$
$$100\cancel{n} < \cancel{n}\log n$$

$$n > 2^{100}$$

Almost surely, we will never face such an input.

Assume that the running time of our algo is

$$f(n) = 3n^2 + 2n + 2\log n + 5$$

Simplify the notation for running time.

$f(n) = 3n^2 + 2n + 2\log n + 5$

Simplification : $O()$ notation

<u>Def</u>^{n}: $f(n) = O(g(n))$ if there exists a a constant c s.t for all $n > n_0$ $f(n) \leq c \cdot g(n)$

(2) Simplify the notation for running time.

$f(n) = 3n^2 + 2n + 2\log n + 5$

Simplification : $O()$ notation

<u>Def</u>^{n}: $f(n) = O(g(n))$ if there exists a a constant c s.t for all $n > n_0$
$f(n) \le c \cdot g(n)$

Apply this def^n:
Is $f(n) = O(n^3)$?

(2) Simplify the notation for running time.

$f(n) = 3n^2 + 2n + 2\log n + 5$

Simplification : $O()$ notation

<u>Def</u>^{n}: $f(n) = O(g(n))$ if there exists a a constant c s.t for all $n > n_0$ $f(n) \le c \cdot g(n)$

Apply this def^n:

Is $f(n) = O(n^3)$? ✓ Yes

Is $f(n) = O(n^2)$?

(2) Simplify the notation for running time.

$f(n) = 3n^2 + 2n + 2\log n + 5$

Simplification : O() notation

Defⁿ: $f(n) = O(g(n))$ if there exists a a constant c s.t for all $n > n_0$

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

Apply this defⁿ:

Is $f(n) = O(n^3)$? ✓ Yes

Is $f(n) = O(n^2)$? ✓ Yes

Is $f(n) = O(n)$?

(2) Simplify the notation for running time.

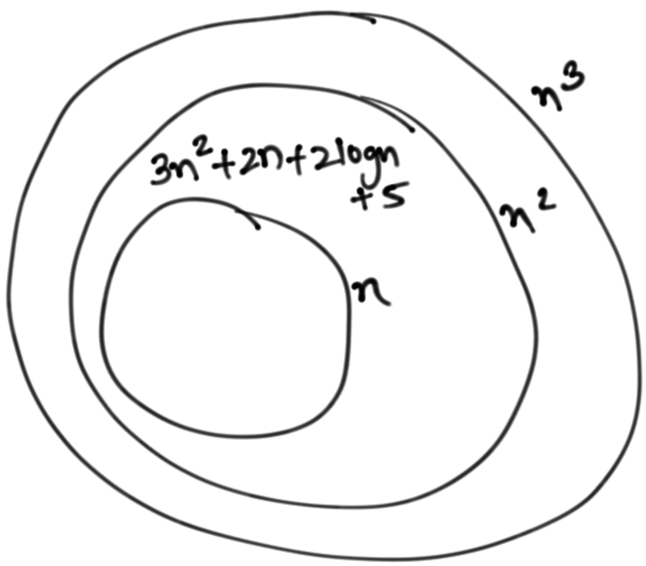
$f(n) = 3n^2 + 2n + 2\log n + 5$

Simplification : $O()$ notation

Def^n: $f(n) = O(g(n))$ if there exists a
a constant c s.t for all $n > n_0$
$f(n) \le c \cdot g(n)$

Apply this def^n:

Is $f(n) = O(n^3)$? ✓ Yes
Is $f(n) = O(n^2)$? ✓ Yes
Is $f(n) = O(n)$? ✗ No



Advantages of order notation

Advantages of order notation

Simplify the expression for running time

Disadvantages of order notation

Advantages of order notation

Simplify the expression for running time

Disadvantages of order notation

Ignores constant factors

($100n$ is worse than $n\log n$)