CS1101S T03D

SESSION 2

I swear its not cos I'm lazy :P

Slides are not fancy anymore :(

Focus will be on you! :)

Main Takeaways

Substitution Model
Recursion

Substitution Model

Evaluating Combinations and Functions

Evaluating Combinations

23 + 42 * 2 - 38

Evaluating Functions

Application vs Normal Order Reduction

Similarities?

Evaluating Combination & Functions

23 + 42 * 2 - 38

Similarities?

Evaluating Combination & Functions

23 + 42 * 2 - 38

Functions: plus(x, y), times(x, y), minus(x, y)

Order Reductions

Applicative vs Normal

```
function f(x) { return square(x) + add_one(x) ; }
function square(x) { return x * x; }
function add_one(x) { return x + 1; }

f(add_one(3));
f(4);
square(4) + add_one(4);
(4 * 4) + (4 + 1);
16 + 5;
19;
```

Applicative or Normal?

```
function f(x) { return square(x) + add_one(x) ; }
function square(x) { return x * x; }
function add_one(x) { return x + 1; }

f(add_one(3));
f(4);
square(4) + add_one(4);
(4 * 4) + (4 + 1);
16 + 5;
19;
```

Applicative or Normal?

Applicative

Evaluate first, expand later

```
function f(x) { return square(x) + add_one(x) ; }
function square(x) { return x * x; }
function add_one(x) { return x + 1; }

f(add_one(3));
f(3 + 1);
square(3 + 1) + add_one(3 + 1);
((3 + 1) * (3 + 1)) + ((3 + 1) + 1);
(4 * 4) + (4 + 1);
16 + 5;
19;
```

Applicative or Normal?

```
function f(x) { return square(x) + add_one(x) ; }
function square(x) { return x * x; }
function add_one(x) { return x + 1; }

f(add_one(3));
f(3 + 1);
square(3 + 1) + add_one(3 + 1);
((3 + 1) * (3 + 1)) + ((3 + 1) + 1);
(4 * 4) + (4 + 1);
16 + 5;
19;
```

Applicative or Normal?

Normal

Expand first, evaluate later

```
function f(x) { return square(x) + add_one(x) ; }
function square(x) { return x * x; }
function add_one(x) { return x + 1; }

f(3);
square(3) + add_one(3);
(3 * 3) + (3 + 1);
9 + 4;
13;
```

Applicative or Normal?

```
function f(x) { return square(x) + add_one(x) ; }
function square(x) { return x * x; }
function add_one(x) { return x + 1; }

f(3);
square(3) + add_one(3);
(3 * 3) + (3 + 1);
9 + 4;
13;
```

Applicative or Normal?

Could be either...

Recursion

Recursion

Summarize how it works in 3 words

Recursion

Summarize how it works in 3 words

Repeat till cannot

Parts of Recursion

Parts of Recursion

Base Case

“Wishful Thinking”

Base Case

“Wishful Thinking”

“Wishful Thinking”

Pretend next recursion gives correct answer…
What to do with it?

```
function recursion(x) {
    return is_done(x)
            ? done()
            : process(recursion(next_x(x)));
}
```

General Structure of Recursion

Deferred Operations

Deferred Operations

Waiting for “Wishful Thinking”… Waiting…

```
function recursive(x) {
    return check_base_case(x)
            ? return_base_case(x)
            : deferred_operations(
                wishful_thinking(x)
            );
}

function check_base_case(x) { ... }
function return_base_case(x) { ... }
function deferred_operations(x) { ... }
function wishful_thinking(x) { ... }
```

General Structure of Recursive Process

Iterative Process

Summarize how it works in 4 words.

Iterative Process

Summarize how it works in 4 words.

Do first, wait later

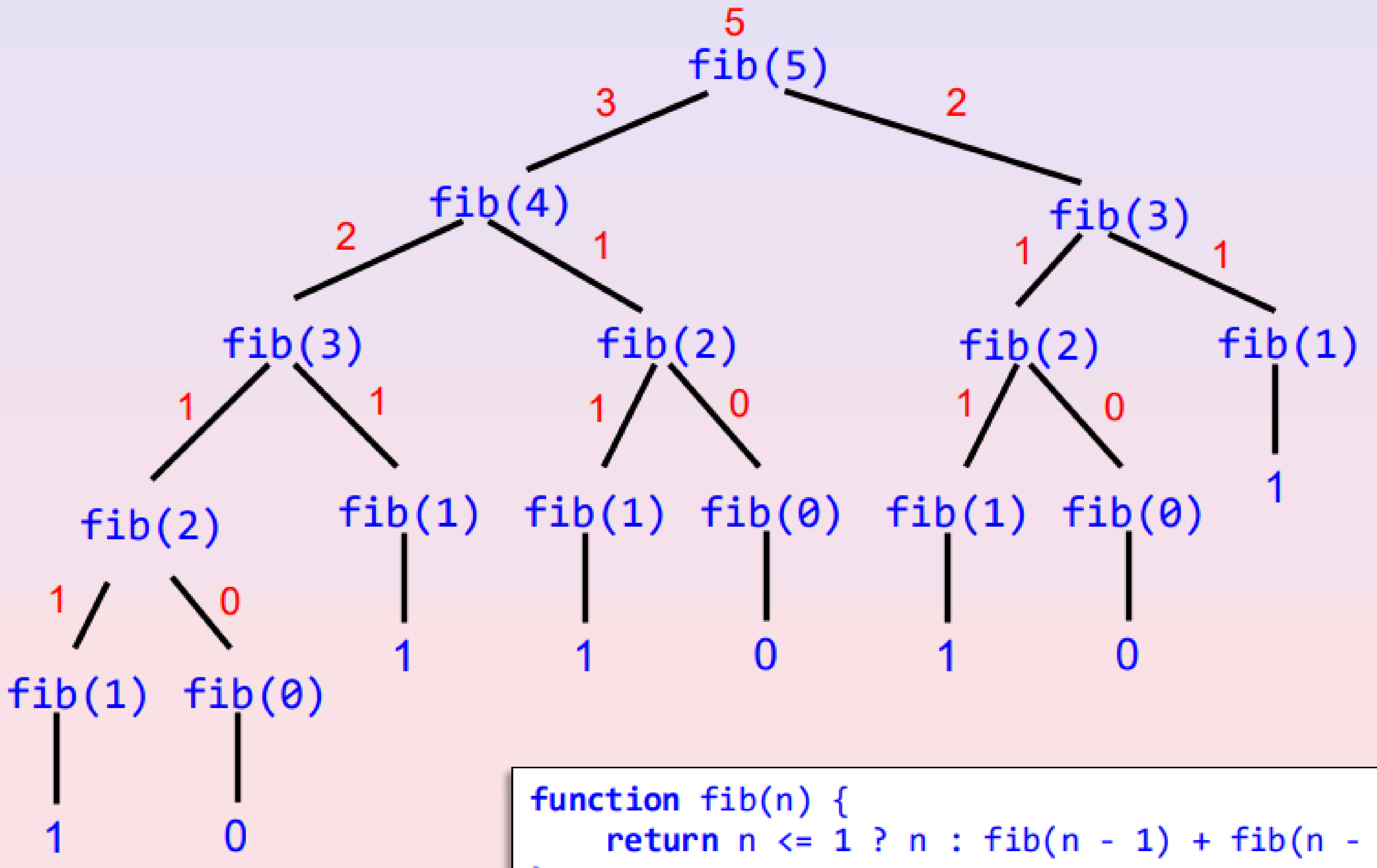
```
function iterative(x) {
    function helper(step, res) {
        return check_done(step, x)
               ? res
               : helper(next_step(step), process(step, res));
    }

    return helper(initial, start_value);
}

function check_done(step, x) { ... }
function next_step(step) { ... }
function process(step, res) { ... }
```

General Structure of Iterative Process

Tree Recursion

In programming, trees grow downwards.

Not enough grass touched…




```
function fib(n) {
    return n <= 1 ? n : fib(n - 1) + fib(n - 2);
}
```

Lecture Example: Fibonacci sequence

Time Complexity

How long does the program run?

Space Complexity

How much does the program have to keep track?

```
factorial(4)
➔ 4 * factorial(3)
➔ 4 * (3 * factorial(2))
➔ 4 * (3 * (2 * factorial(1)))
➔ 4 * (3 * (2 * 1))
➔ 4 * (3 * 2)
➔ 4 * 6
➔ 24
```

- **Observation:**
  - **Number of operations** *grows* ***linearly*** proportional to *n*

```
factorial(4)
➔ 4 * factorial(3)
➔ 4 * (3 * factorial(2))
➔ 4 * (3 * (2 * factorial(1)))
➔ 4 * (3 * (2 * 1))
➔ 4 * (3 * 2)
➔ 4 * 6
➔ 24
```

- **Observation:**
  - **Number of deferred operations** *grows* ***linearly*** proportional to *n*
    - Deferred operations need to be "remembered"

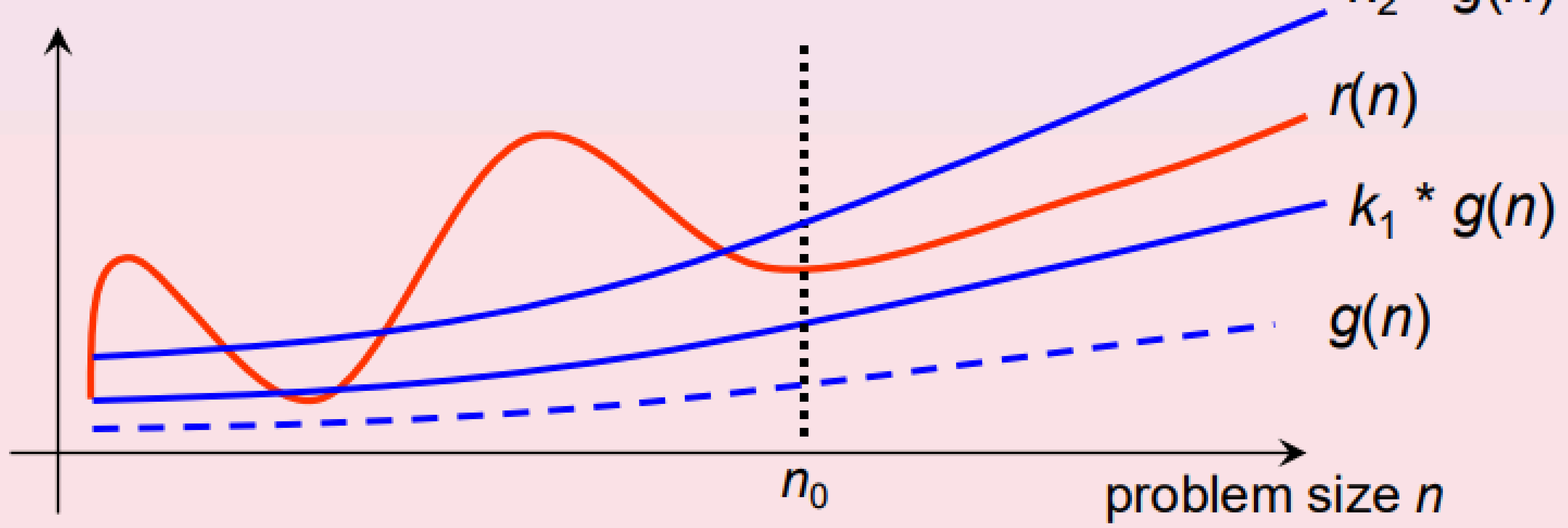
Space Complexity

Big Theta Notation

For both time and space complexity

After a threshold n_0, what is the mathematical function that bounds both UPPER and LOWER extreme cases.

- The function r has order of growth $\Theta(g(n))$ if there are positive constants k_1 and k_2 and a number n_0 such that $k_1 \cdot g(n) \leq r(n) \leq k_2 \cdot g(n)$ for all $n > n_0$



Big Theta Notation

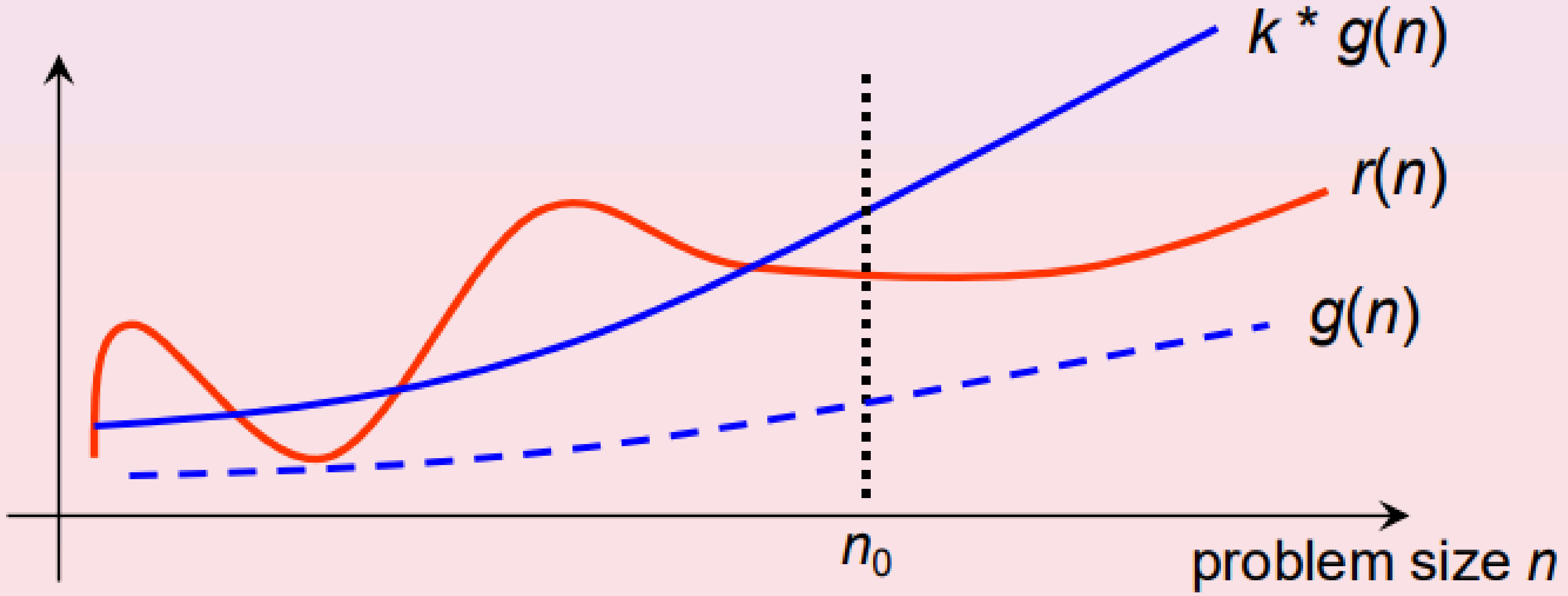
Big-O Notation

For both time and space complexity

After a threshold n_0, what is the mathematical function that bounds the UPPER extreme cases.

Definition:

- The function r has order of growth $O(g(n))$ if there is a positive constant k and a number n_0 such that $r(n) \leq k \cdot g(n)$ for all $n > n_0$



Big-O Notation

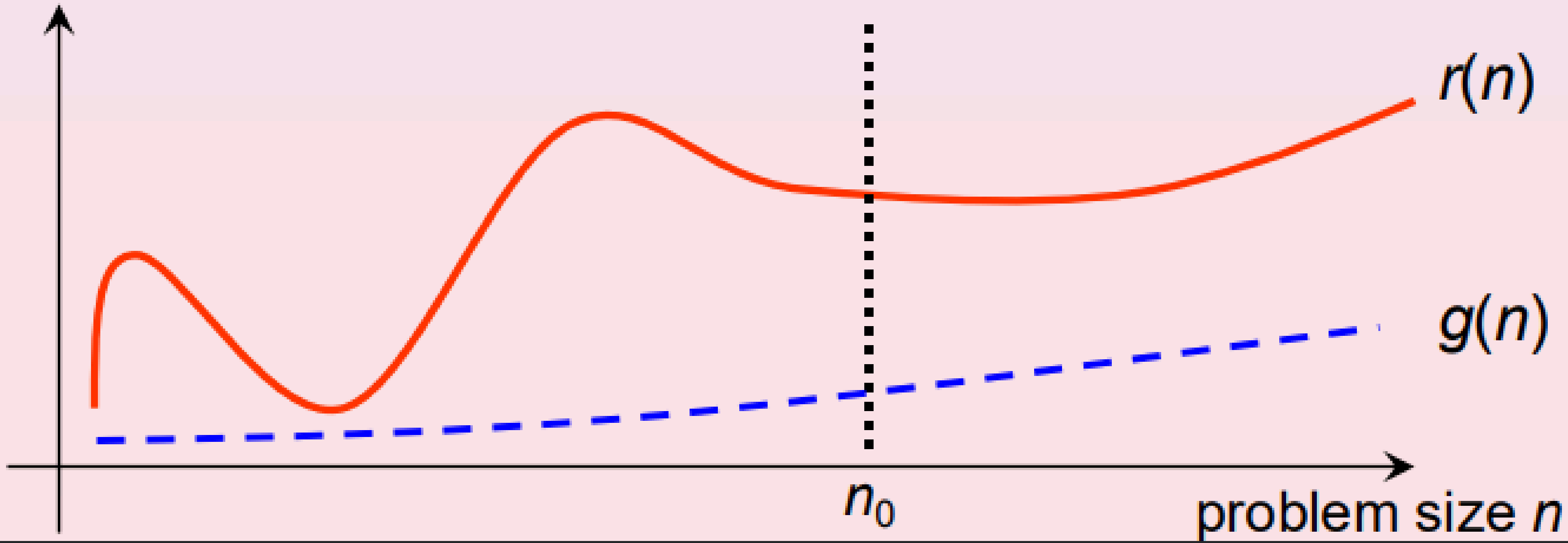
Big-Omega Notation

For both time and space complexity

After a threshold n_0, what is the mathematical function that bounds the LOWER extreme cases.

Definition:

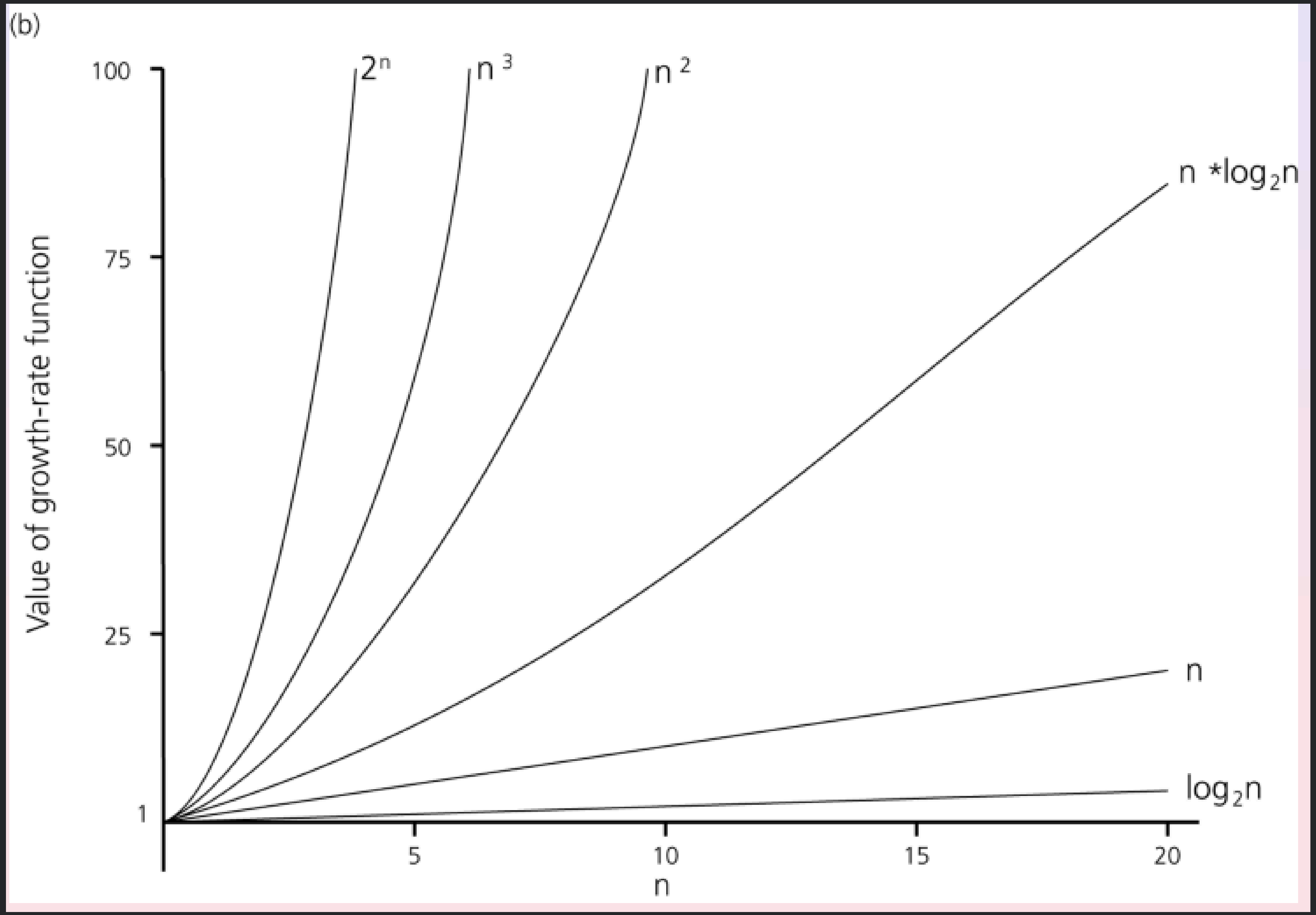
- The function r has order of growth $\Omega(g(n))$ if there is a positive constant k and a number n_0 such that $k \cdot g(n) \leq r(n)$ for all $n > n_0$



Big Omega Notation

Some Common $g(n)$

- 1
- $\log n$
- n
- $n \log n$
- n^2
- n^3
- 2^n



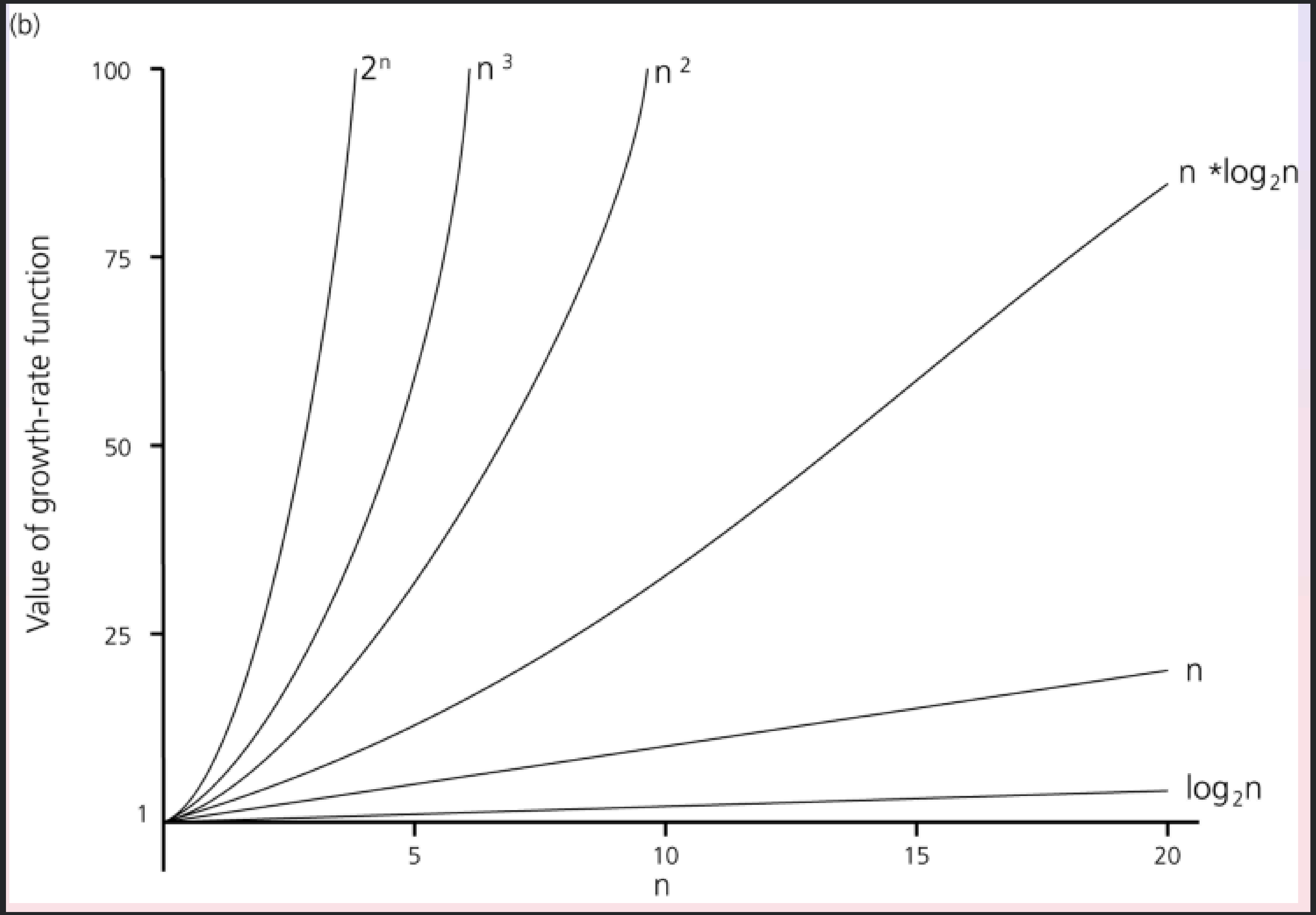
Comparing the growth rates

Do the constants matter?

O(10000)

Omega(50n)

O(10^{1000} x 2^n)



Studio Sheet

studio-s3.pdf

In-Class Studio Sheet

studio-s3-in-class.pdf

gimme a sec, imma send on tele :)