COL100: Introduction to Computer Science

7.1: More examples of efficiency analysis

Fibonacci numbers

$$F_1 = 1,$$
$$F_2 = 1,$$
$$F_n = F_{n-1} + F_{n-2} \text{ for all } n \geq 3.$$

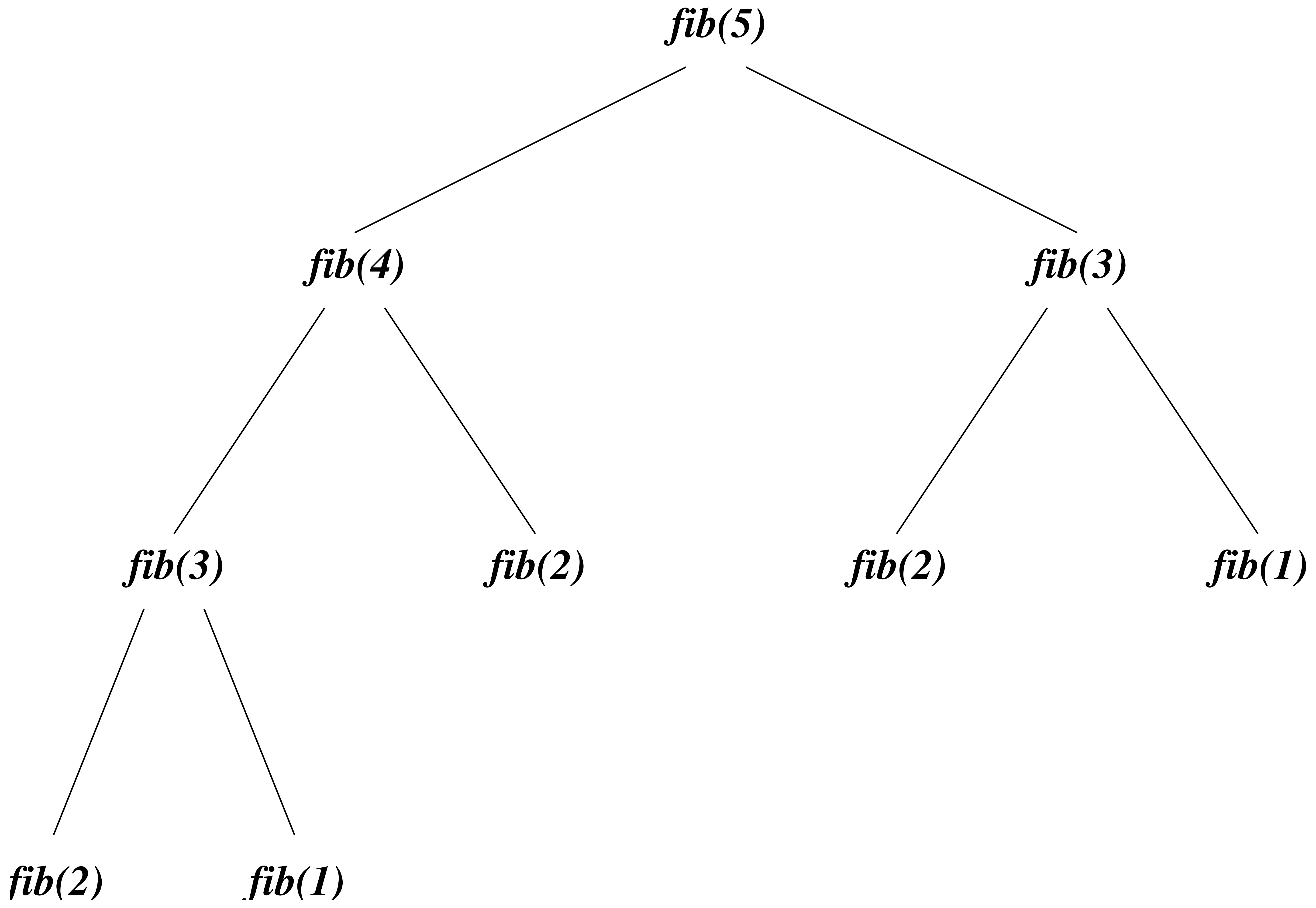
$F_1 = 1$
$F_2 = 1$
$F_3 = 2$
$F_4 = 3$
$F_5 = 5$
$F_6 = 8$
$F_7 = 13$
$F_8 = 21$
$\vdots$

Naive algorithm:

$$fib(n) = \begin{cases} 1 & \textit{if } n \leq 2, \\ fib(n-1) + fib(n-2) & \text{otherwise.} \end{cases}$$

Why does it turn out to be so slow?

$$fib(n) = \begin{cases} 1 & \text{if } n \leq 2, \\ fib(n-1) + fib(n-2) & \text{otherwise.} \end{cases}$$



$$fib(n) = \begin{cases} 1 & \textit{if } n \leq 2, \\ fib(n-1) + fib(n-2) & \text{otherwise.} \end{cases}$$

Show by induction that:

- Number of additions = $fib(n) - 1$
- $fib(n) = (\phi^n - \psi^n)/\sqrt{5}$ where ϕ, $\psi = (1 \pm \sqrt{5})/2$, and therefore, $fib(n) = O(\phi^n)$.

So time complexity of $fib(n)$ is exponential in n, which is intractable for large n.

$$fib(n) = \begin{cases} 1 & \textit{if } n \leq 2, \\ fib(n-1) + fib(n-2) & \text{otherwise.} \end{cases}$$

Space complexity of *fib*(*n*):

$$S(n) = \begin{cases} 1 & \textit{if } n \leq 2, \\ 1 + \max(S(n-1), S(n-2)) & \text{otherwise.} \end{cases}$$

Show that this is just *O*(*n*).

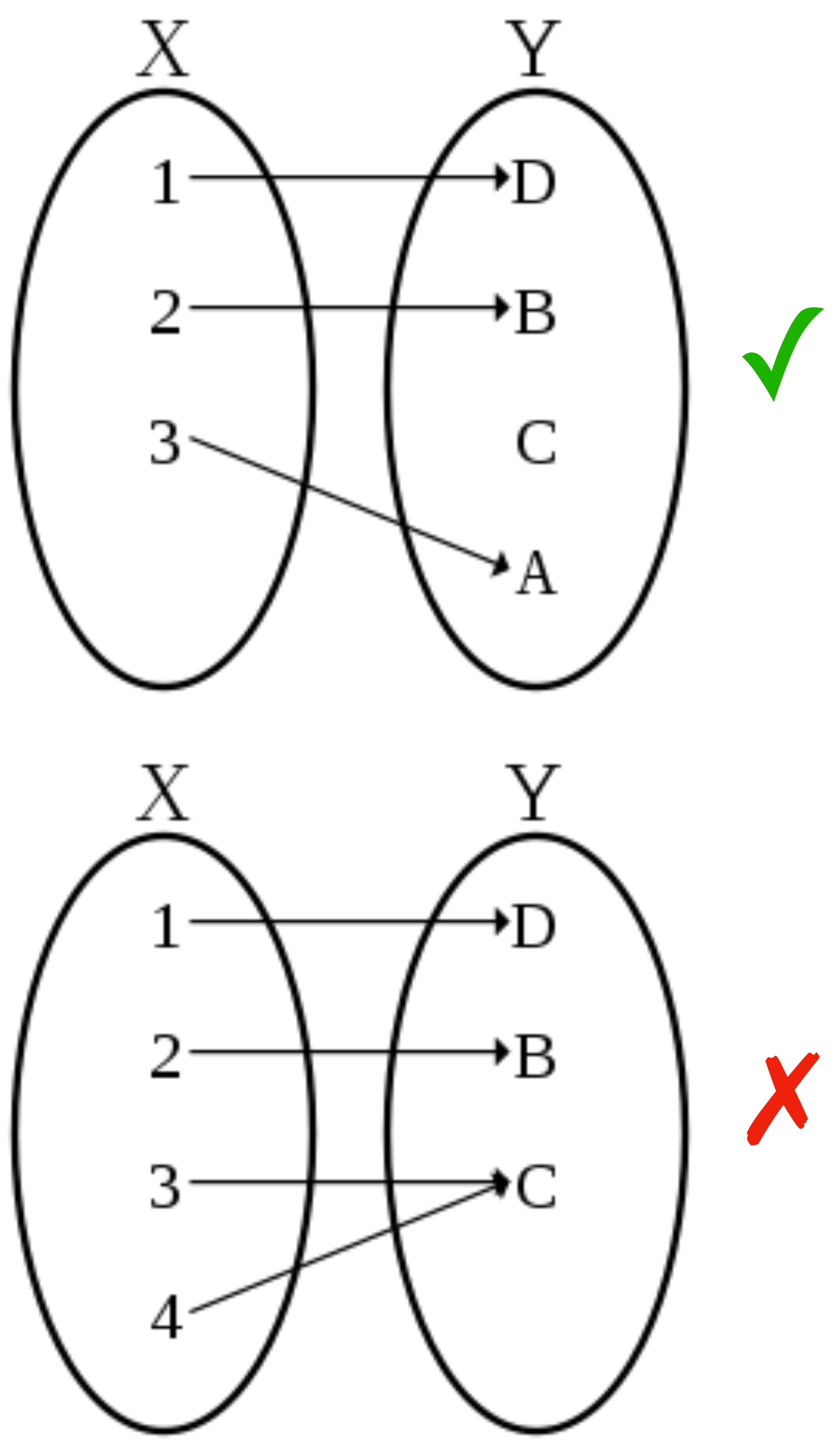
Pairwise comparisons and quadratic time

A function $f : X \to Y$ is said to be *injective* or *one-to-one* if it maps every input value to a different output value, i.e.

$$f(a) \neq f(b) \text{ for all } a, b \in X, a \neq b.$$

- $f(n) = 2n$ is injective because $m \neq n \Rightarrow 2m \neq 2n$.
- $f(n) = n^2$ is not injective (on $\mathbb{Z}$) because $1 \neq -1$ but $1^2 = (-1)^2$.

Design an algorithm to check if $f : \{a, a+1, \ldots, b\} \to \mathbb{Z}$ is injective.



f is trivially injective if $a = b$.

f is injective on domain $\{a, \ldots, b\}$ if (i) it is injective on $\{a, \ldots, b-1\}$, and (ii) $f(b) \neq f(i)$ for all $i \in \{a, \ldots, b-1\}$.

$$isInj(f, a, b) = \begin{cases} \text{true} & \text{if } a = b, \\ isInj(f, a, b-1) \wedge isDiff(f(b), f, a, b-1) & \text{otherwise,} \end{cases}$$

$$isDiff : \mathbb{Z} \times (\mathbb{Z} \to \mathbb{Z}) \times \mathbb{Z} \times \mathbb{Z} \to \mathbb{B}$$

$isDiff(y, f, a, b)$ is true if and only if $y \neq f(i)$ for all $i \in \{a, \ldots, b\}$.

$$isDiff(y, f, a, b) = \begin{cases} \text{true} & \text{if } b < a, \\ \text{false} & \text{if } y = f(b), \\ isDiff(y, f, a, b-1) & \text{otherwise.} \end{cases}$$

What is the time complexity of *isInj*(f, a, b)?

Intuitively, *isInj* checks that

$$
\begin{aligned}
&\ f(a+1) \neq f(a) \\
&\wedge\ f(a+2) \neq f(a) \wedge f(a+2) \neq f(a+1) \\
&\wedge\ f(a+3) \neq f(a) \wedge f(a+3) \neq f(a+1) \wedge f(a+3) \neq f(a+2) \\
&\vdots \\
&\wedge\ f(b) \neq f(a) \wedge f(b) \neq f(a+1) \wedge f(b) \neq f(a+2) \wedge \cdots \wedge f(b) \neq f(b-1)
\end{aligned}
$$

so it should make about $1 + 2 + \cdots + (n-1) = n(n-1)/2 = O(n^2)$ comparisons, where $n = b - a + 1$.

Formally,

$$T(1) = 0,$$

$$T(n) = T(n-1) + T_D(n-1)$$

where $T_D(n)$ is the time complexity of *isDiff*(y, f, a, b) with $n = b - a + 1$.

Problem: Time required for *isDiff* also depends on y and f:

- If a collision $y = f(i)$ is found early, e.g. f = const, $T_D(n) \ll n$.
- If a collision $y = f(i)$ is never found, $T_D(n) = n$.

Formally, the *worst-case* time complexity of *isInj* is

$$T(1) = 0,$$

$$\begin{aligned} T(n) &= T(n-1) + T_D(n-1) \\ &= T(n-1) + (n-1) \end{aligned}$$

since $T_D(n) = n$ is the worst-case time complexity of *isDiff*.

Then one can prove that $T(n) = n(n-1)/2 = O(n^2)$.

A fallacy

Let us prove by induction that $T(n) = O(n)$.

Base case: $T(1) = 0 = O(1)$.

Induction hypothesis: $T(n-1) = O(n-1) = O(n)$.

Induction step:

$$\begin{aligned} T(n) &= T(n-1) + T_D(n-1) \\ &= O(n) + n - 1 \\ &= O(n) \end{aligned}$$

What's wrong with this argument?

$T(n) = O(n)$ means there are <u>constants</u> C, n_0 <u>independent of n</u> such that $T(n) \leq Cn$ for all $n \geq n_0$.

Induction hypothesis: $T(n-1) \leq C(n-1)$ if $n - 1 \geq n_0$.

Induction step: We need to show that $T(n) \leq Cn$ if $n \geq n_0$.

If $n \geq n_0 + 1$,

$$\begin{aligned} T(n) &= T(n-1) + T_D(n-1) \\ &\leq C(n-1) + (n-1) \\ &= (C+1)n - C - 1. \end{aligned}$$

(If $n = n_0$, more problems…)

$$O(n) + O(n) = O(n),$$

but

$$\underbrace{O(n) + O(n) + \cdots + O(n)}_{(n \text{ times})} = O(n^2)!$$

Afterwards

- The course webpage (http://www.cse.iitd.ac.in/~narain/courses/col100/) has an updated copy of the notes without missing figures.
- Complete the proofs for the time and space complexity of *fib*(n).
- A function $f : X \to Y$ is *surjective*, or *onto*, if every output value is mapped to by some input value. Design an algorithm to check if a function $f : \{a, \ldots , b\} \to \{c, \ldots, d\}$ is surjective, and find its time complexity in terms of $m = b - a + 1$ and $n = d - c + 1$.