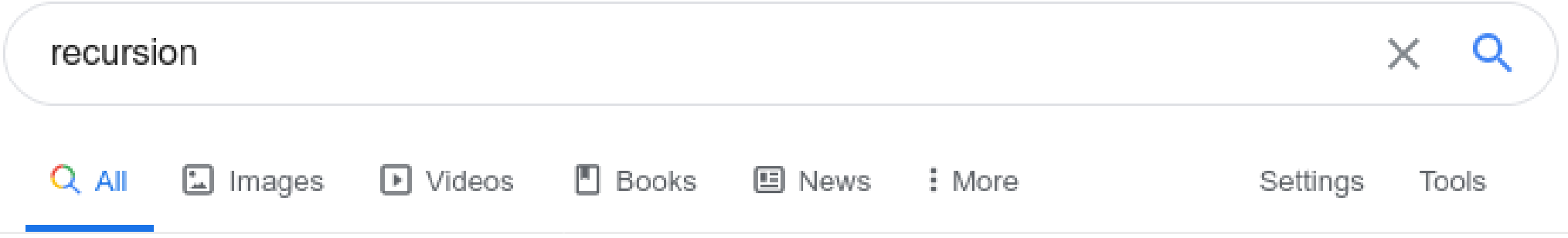
Rekurze

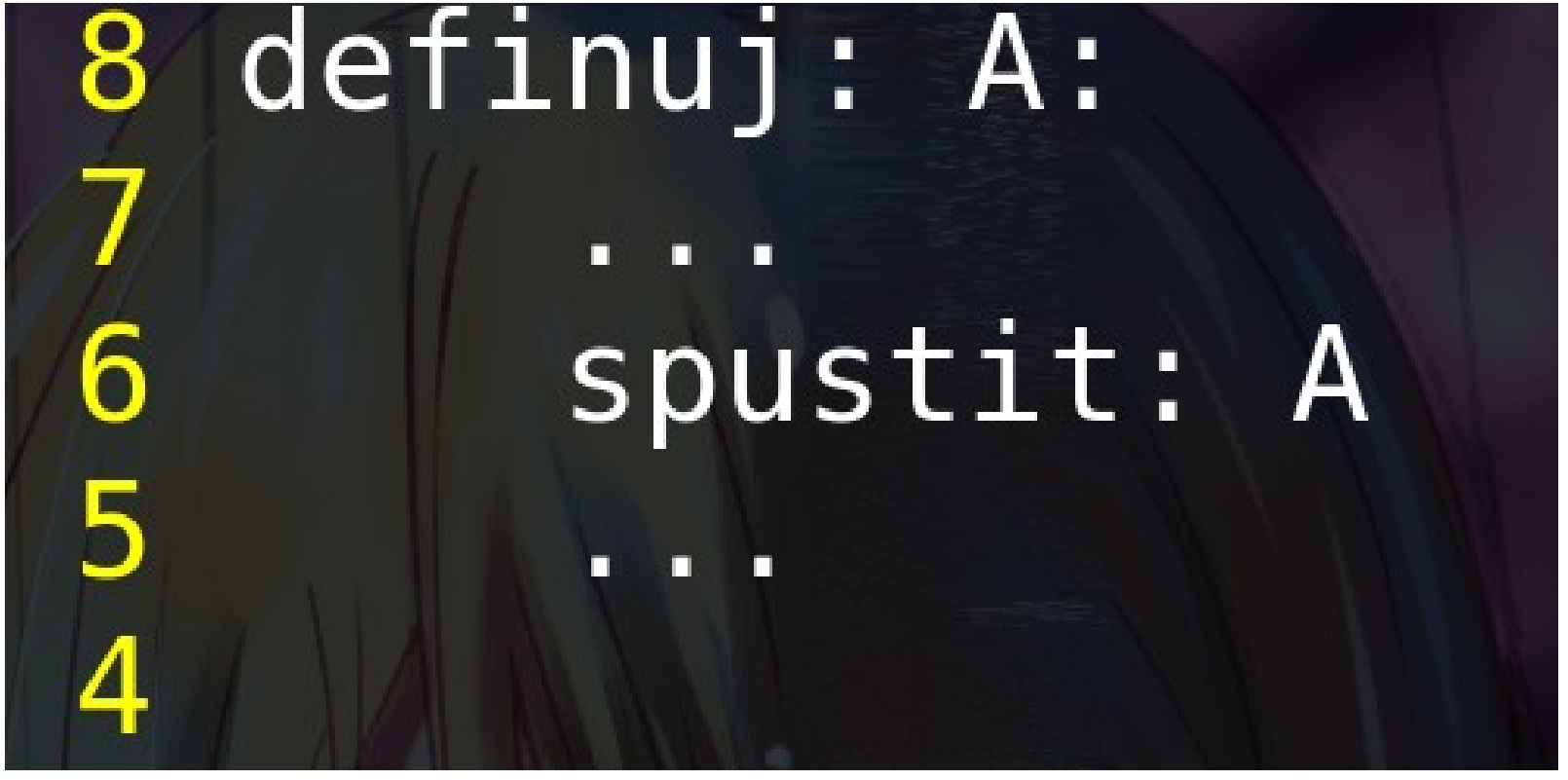


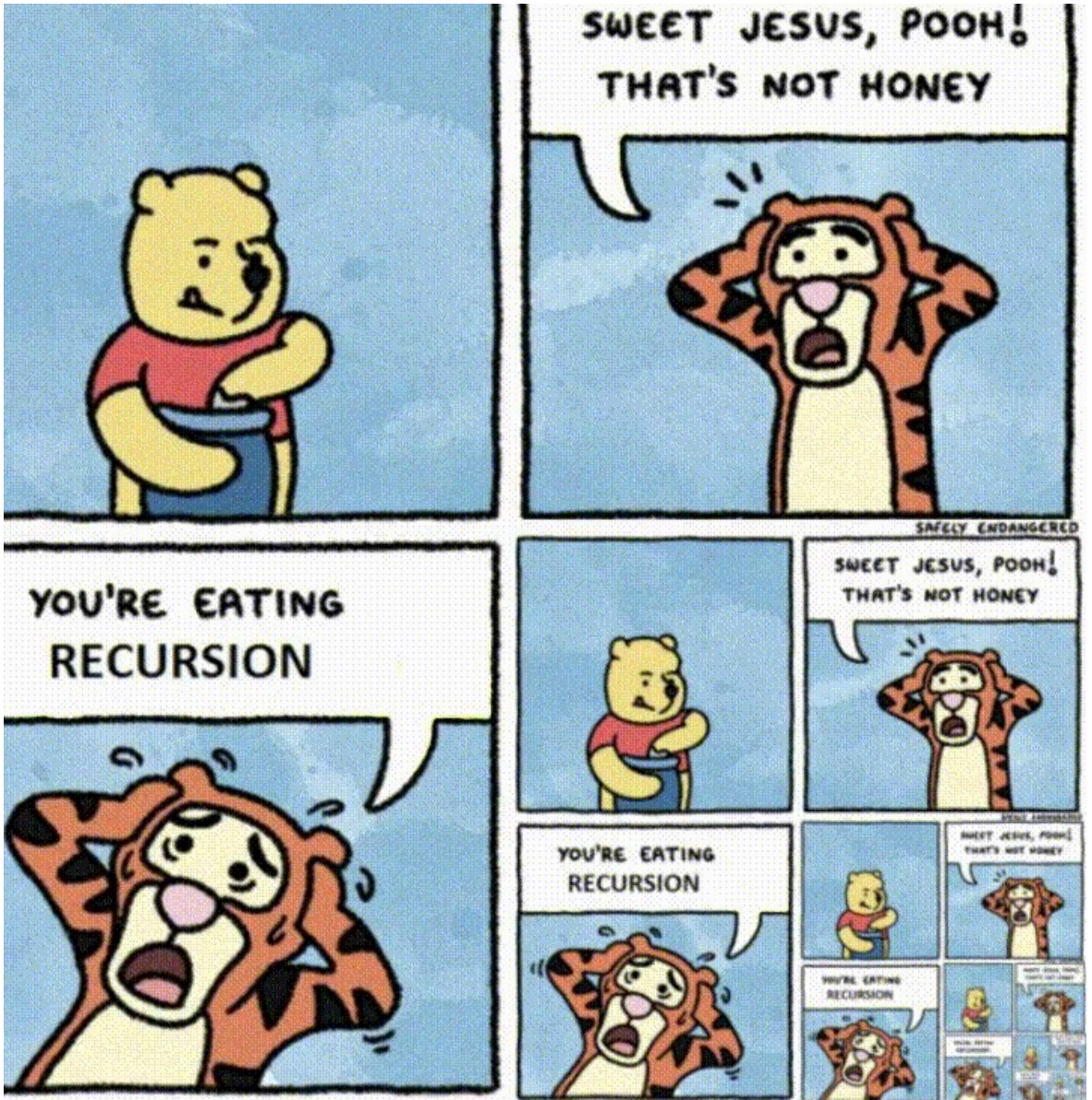
Rekurze

(Redirected from Rekurze)
Redirect page

↳ Rekurze

Petr Šťastný

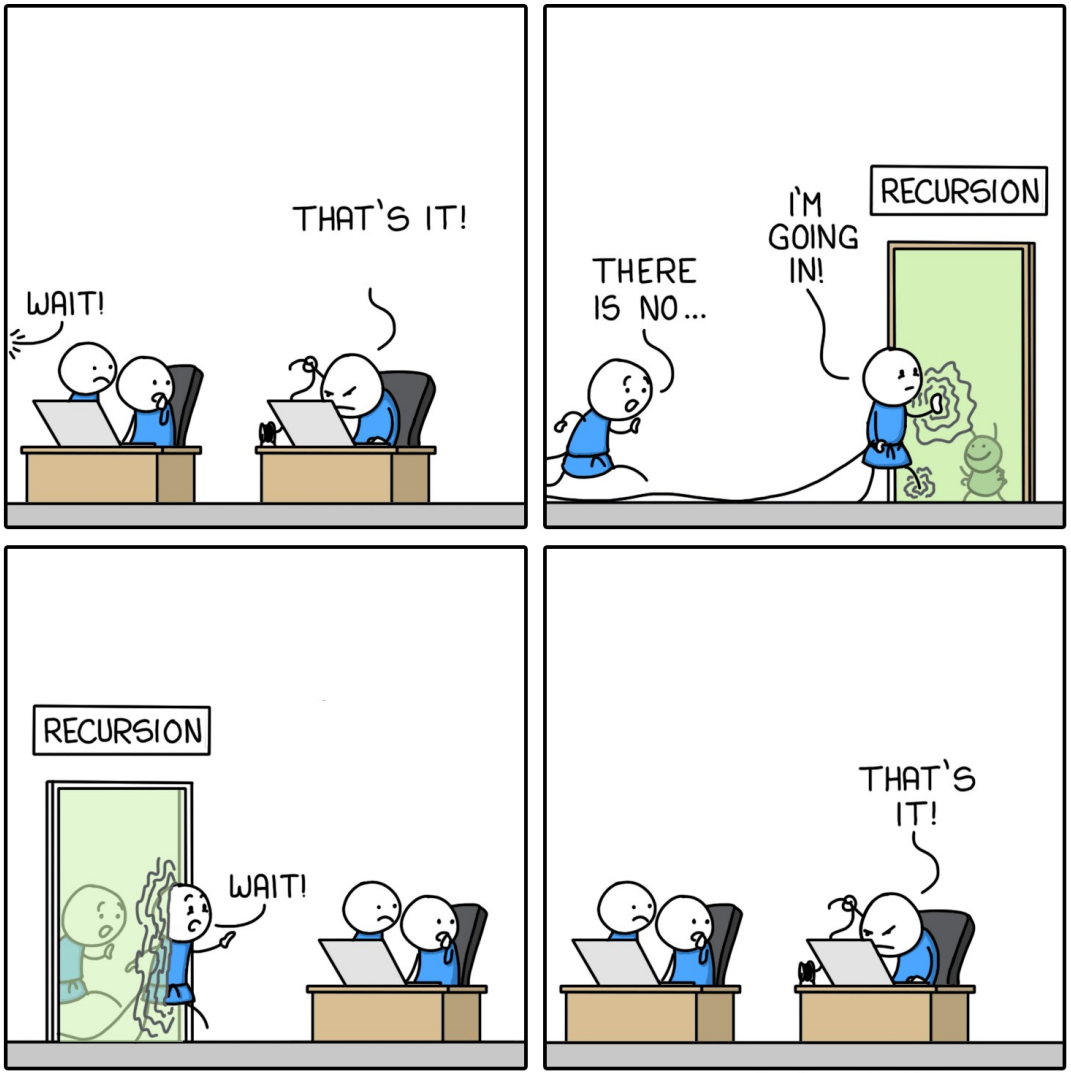

8 definuj: A:
7 ...
6 spustit: A
5 ...
4


SWEET JESUS, POOH!
THAT'S NOT HONEY
SAFELY ENDANGERED
YOU'RE EATING
RECURSION

Jak napsat rekurzi?

- Viz jak napsat rekurzi

EXIT CONDITION



Terminologie [editovat | editovat zdroj]

Rekurzivní chování může být různé v závislosti na tom, kolik podprogramů se jí účastní. Rozlišujeme dva základní typy dělení.

1. Volání může probíhat přímo nebo nepřímo:

- *Přímá rekurze* nastává, když podprogram volá přímo sám sebe.
- *Nepřímá rekurze* je situace, kdy vzájemné volání podprogramů vytvoří „kruh". Např. ve funkci *A* se volá funkce *B* a ve funkci *B* se volá opět funkce *A*.

2. Podprogram může být volán jednou nebo vícekrát:

- *Lineární rekurze* nastává, pokud podprogram při vykonávání svého úkolu volá sama sebe pouze jednou. Vytváří se takto lineární struktura postupně volaných podprogramů.
- *Stromová rekurze* nastává, pokud se funkce nebo procedura v rámci jednoho vykonání svého úkolu vyvolá vícekrát. Strukturu volání je možné znázornit jako zakořeněný strom. Pro dvě volání v jednom průchodu vzniká binární strom, pro tři ternární strom, atd. (Počet rekurzivních volání nemusí být konstantní, např. při rekurzivním procházení grafu voláme zpracování na všechny sousedy vrcholu, a těch je obecně různý počet.)

- Sečíst čísla od 1 do X

```
sumFromOneToX 1 = 1
sumFromOneToX n = n + sumFromOneToX(n-1)
```

```
fn sum_from_one_to_x(upper_bound: u32) -> u32 {
    if upper_bound == 0:
        return upper_bound
    else:
        return upper_bound + sum_from_one_to_x(upper_bound - 1)
}
```

```
fn sum_from_one_to_x(upper_bound: u32) -> u32 {
    if upper_bound == 0:
        return upper_bound
    else:
        return upper_bound + sum_from_one_to_x(upper_bound - 1)
}
```

```
f(5) = 5 + f(4)
       f(4) = 4 + f(3)
              f(3) = 3 + f(2)
                     f(2) = 2 + f(1)
                            f(1) = 1 + f(0)
                                   f(0) = 0
```

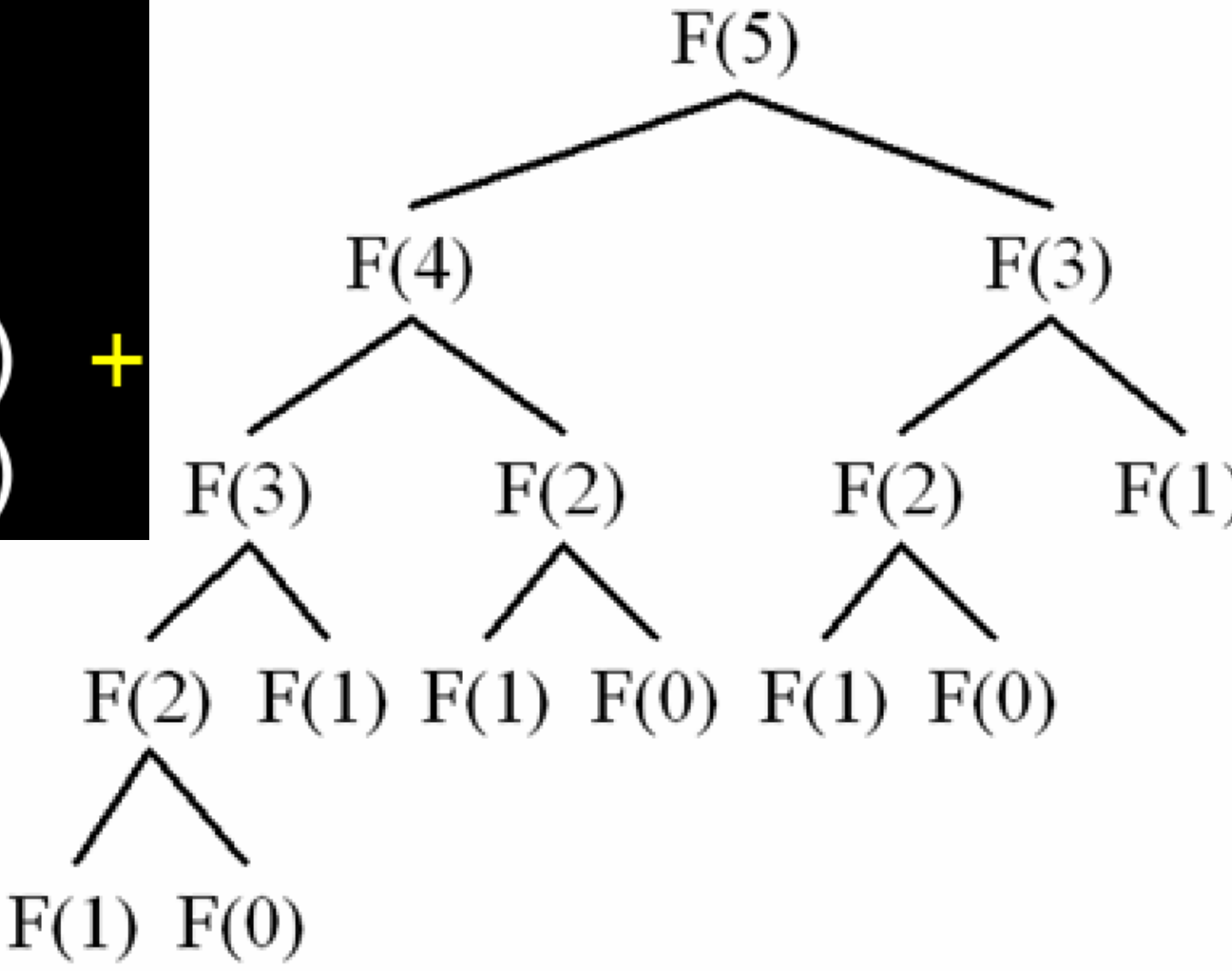
```
factorial(4) = 4 * factorial(3)
...
factorial(4) = 4 * 3 * 2 * 1
```

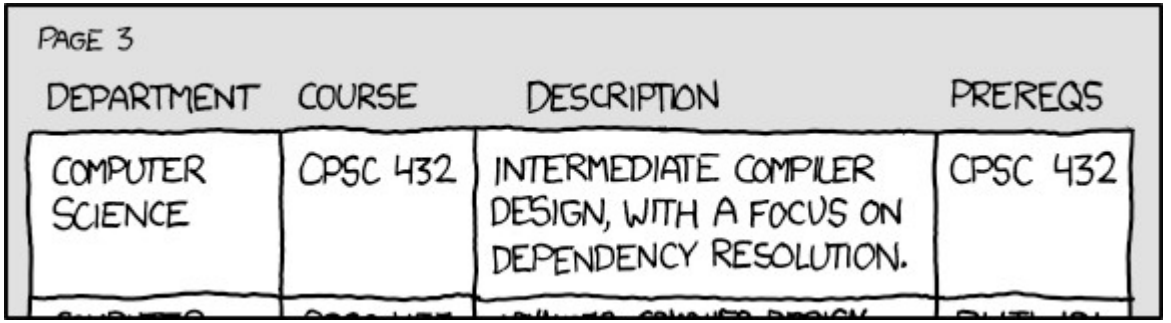
```
fn factorial(n: u32) -> u32 {
    if n == 0:
        return 1;
    else:
        return n * factorial(n-1);
}
```

- Fibonacci
- 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ...

```
fn fibonacci(n: u32) {
    if n == 0 {
        return 0;
    }
    if n == 1 {
        return 1;
    }
    return fibonacci(n-2) + fibonacci(n-1);
}
```

```
fib 0 = 0
fib 1 = 1
fib n =
  fib(n-1) +
  fib(n-2)
```





- WINE — WINE Is Not an Emulator[20] (Originally, Windows Emulator[21])

[–] [deleted] **1 point** 9 years ago

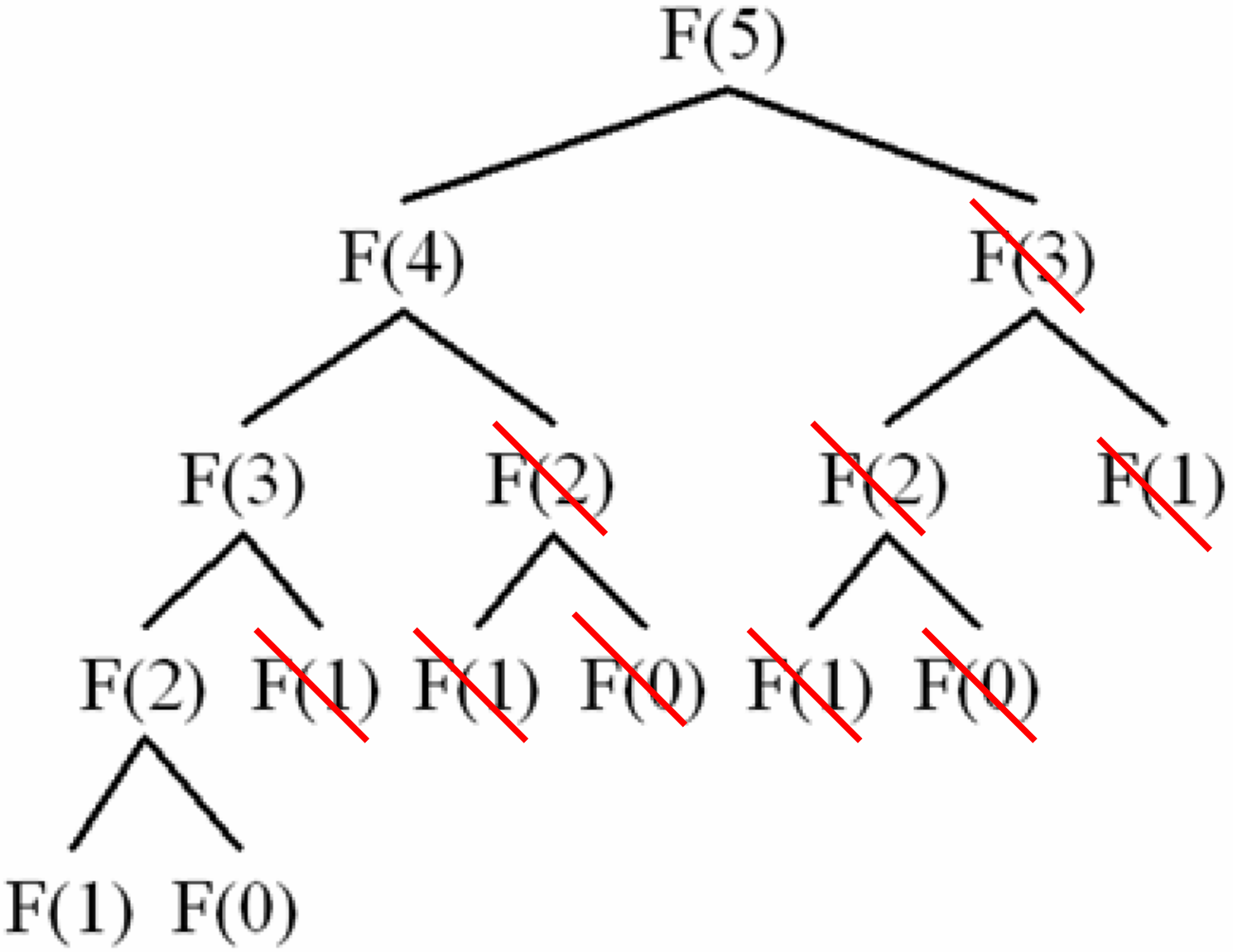
EMACS: **E**MACS **M**akes **A**ny **C**omputer **S**low

GNU is a recursive acronym for "GNU's Not Unix!",

- TikZ – TikZ ist kein Zeichenprogramm
- PHP — PHP: Hypertext Preprocessor
- JACK — JACK Audio Connection Kit
- cURL — Curl URL Request Library

“Není to takhle pomalé?”

- Ano!
- Memoization (česky: Tabelace)
- Pamatuju si to co jsem už spočítal


F(5)
F(4)
F(3)
F(3)
F(2)
F(2)
F(1)
F(2)
F(1)
F(1)
F(0)
F(1)
F(0)
F(1)
F(0)