Algoritmy a datové struktury

Petr Švarný 2021

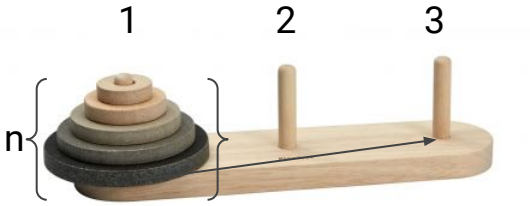
Zjednodušená podoba

$$T(n) = \begin{cases} \Theta(1), & \text{if } n = 1, \\ T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + \Theta(n), & \text{if } n > 1. \end{cases}$$

$$T(n) = 2T(n/2) + \Theta(n)$$

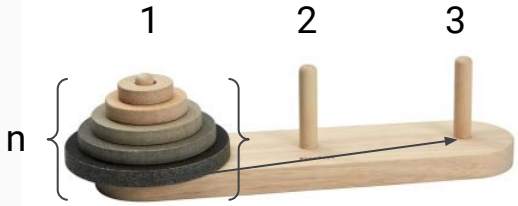
Ukázka převodu - Hanojské věže (lichý počet)

Ukázka převodu - Hanojské věže (lichý počet)




```
Hanoi(n, p1, p2, p3):
  if n>0:
    Hanoi(n-1, p1, p3, p2)
    presun disk z p1 na p3
    Hanoi(n-1, p2, p1, p3)
```

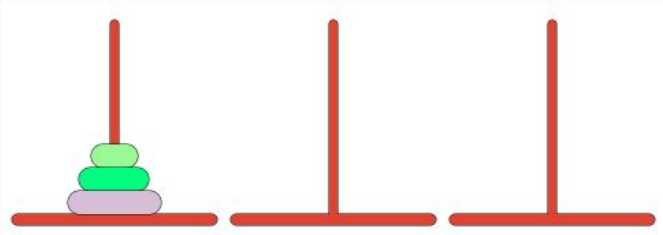
Ukázka převodu - Hanojské věže (lichý počet)




```
Hanoi(1, p1, p2, p3):
  if n>0:
    Hanoi(0, p1, p3, p2)
    presun disk z p1 na p3
    Hanoi(0, p2, p1, p3)
```

Ukázka převodu - Hanojské věže

```
Hanoi(3, p1, p2, p3):
  if n>0:
    Hanoi(2, p1, p3, p2)
    presun disk z p1 na p3
    Hanoi(2, p2, p1, p3)
```



```
Hanoi(2, p1, p3, p2):
  if n>0:
    Hanoi(1, p1, p2, p3)
    presun disk z p1 na p2
    Hanoi(1, p3, p1, p2)
```

```
Hanoi(2, p2, p1, p3):
  if n>0:
    Hanoi(1, p2, p3, p1)
    presun disk z p2 na p3
    Hanoi(1, p1, p2, p3)
```

Ukázka převodu - Hanojské věže

```
Hanoi(3, p1, p2, p3):
  if n>0:
    Hanoi(2, p1, p3, p2)
    presun disk z p1 na p3
    Hanoi(2, p2, p1, p3)
```

```
Hanoi(2, p1, p3, p2):
  if n>0:
    Hanoi(1, p1, p2, p3)
    presun disk z p1 na p2
    Hanoi(1, p3, p1, p2)
```

```
Hanoi(2, p2, p1, p3):
  if n>0:
    Hanoi(1, p2, p3, p1)
    presun disk z p2 na p3
    Hanoi(1, p1, p2, p3)
```

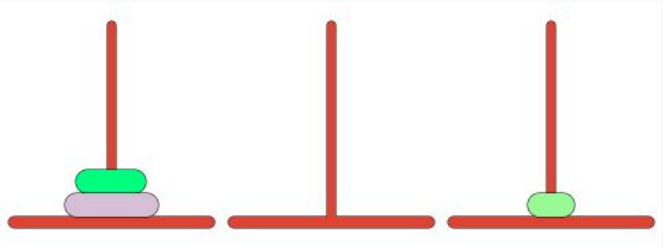
```
Hanoi(1, p1, p2, p3):
  if n>0:
    Hanoi(0, p1, p3, p2)
    presun disk z p1 na p3
    Hanoi(0, p2, p1, p3)
```

```
Hanoi(1, p3, p1, p2):
    presun disk z p3 na p2
```

```
Hanoi(1, p2, p3, p1):
    presun disk z p2 na p1
```

```
Hanoi(1, p1, p2, p3):
    presun disk z p1 na p3
```

1



Ukázka převodu - Hanojské věže

```
Hanoi(3, p1, p2, p3):
  if n>0:
    Hanoi(2, p1, p3, p2)
    presun disk z p1 na p3
    Hanoi(2, p2, p1, p3)
```

```
Hanoi(2, p1, p3, p2):
  if n>0:
    Hanoi(1, p1, p2, p3)
    presun disk z p1 na p2
    Hanoi(1, p3, p1, p2)
```

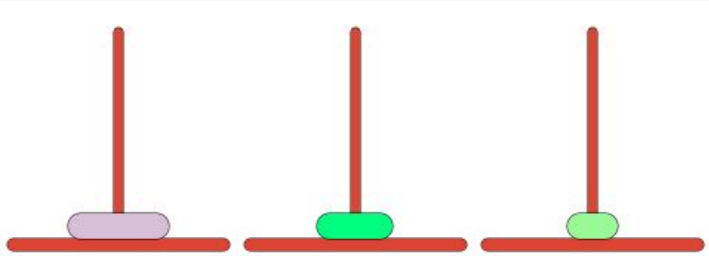
```
Hanoi(2, p2, p1, p3):
  if n>0:
    Hanoi(1, p2, p3, p1)
    presun disk z p2 na p3
    Hanoi(1, p1, p2, p3)
```

```
Hanoi(1, p1, p2, p3):
  if n>0:
    Hanoi(0, p1, p3, p2)
    presun disk z p1 na p3
    Hanoi(0, p2, p1, p3)
```

```
Hanoi(1, p3, p1, p2):
    presun disk z p3 na p2
    Hanoi(1, p2, p3, p1)
```

```
Hanoi(1, p2, p3, p1):
    presun disk z p2 na p1
    Hanoi(1, p1, p2, p3)
```

```
Hanoi(1, p1, p2, p3):
    presun disk z p1 na p3
```



Ukázka převodu - Hanojské věže

```
Hanoi(3, p1, p2, p3):
  if n>0:
    Hanoi(2, p1, p3, p2)
    presun disk z p1 na p3
    Hanoi(2, p2, p1, p3)
```

```
Hanoi(2, p1, p3, p2):
  if n>0:
    Hanoi(1, p1, p2, p3)
    presun disk z p1 na p2
    Hanoi(1, p3, p1, p2)
```

```
Hanoi(2, p2, p1, p3):
  if n>0:
    Hanoi(1, p2, p3, p1)
    presun disk z p2 na p3
    Hanoi(1, p1, p2, p3)
```

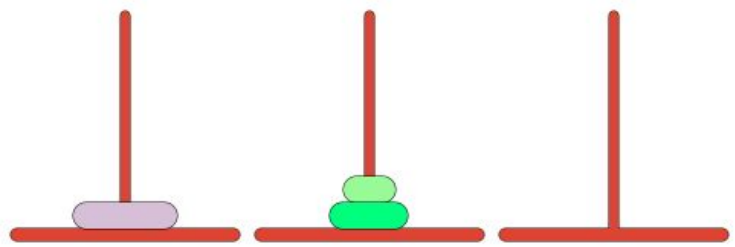
```
Hanoi(1, p1, p2, p3):
  if n>0:
    Hanoi(0, p1, p3, p2)
    presun disk z p1 na p3
    Hanoi(0, p2, p1, p3)
```

```
Hanoi(1, p3, p1, p2):
    presun disk z p3 na p2
    Hanoi(1, p2, p3, p1)
```

```
Hanoi(1, p2, p3, p1):
    presun disk z p2 na p1
    Hanoi(1, p1, p2, p3)
```

```
Hanoi(1, p1, p2, p3):
    presun disk z p1 na p3
```

1 2 3



Ukázka převodu - Hanojské věže

```
Hanoi(3, p1, p2, p3):
  if n>0:
    Hanoi(2, p1, p3, p2)
    presun disk z p1 na p3
    Hanoi(2, p2, p1, p3)
```

```
Hanoi(2, p1, p3, p2):
  if n>0:
    Hanoi(1, p1, p2, p3)
    presun disk z p1 na p2
    Hanoi(1, p3, p1, p2)
```

```
Hanoi(2, p2, p1, p3):
  if n>0:
    Hanoi(1, p2, p3, p1)
    presun disk z p2 na p3
    Hanoi(1, p1, p2, p3)
```

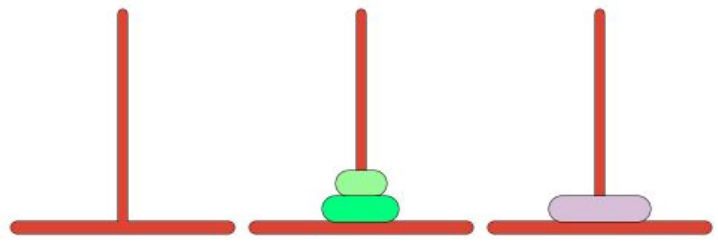
```
Hanoi(1, p1, p2, p3):
  if n>0:
    Hanoi(0, p1, p3, p2)
    presun disk z p1 na
p3
    Hanoi(0, p2, p1, p3)
```

```
Hanoi(1, p3, p1, p2):
    presun disk z p3 na
p2
    Hanoi(1, p2, p3, p1)
```

```
Hanoi(1, p2, p3, p1):
    presun disk z p2 na
p1
    Hanoi(1, p1, p2, p3)
```

```
Hanoi(1, p1, p2, p3):
    presun disk z p1 na
p3
```

1 2 3 4



Ukázka převodu - Hanojské věže

```
Hanoi(3, p1, p2, p3):
  if n>0:
    Hanoi(2, p1, p3, p2)
    presun disk z p1 na p3
    Hanoi(2, p2, p1, p3)
```

4

```
Hanoi(2, p1, p3, p2):
  if n>0:
    Hanoi(1, p1, p2, p3)
    presun disk z p1 na p2
    Hanoi(1, p3, p1, p2)
```

2

```
Hanoi(2, p2, p1, p3):
  if n>0:
    Hanoi(1, p2, p3, p1)
    presun disk z p2 na p3
    Hanoi(1, p1, p2, p3)
```

```
Hanoi(1, p1, p2, p3):
  if n>0:
    Hanoi(0, p1, p3, p2)
    presun disk z p1 na
p3
    Hanoi(0, p2, p1, p3)
```

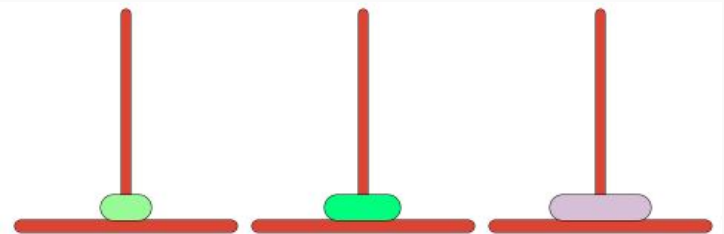
1

3

```
Hanoi(1, p3, p1, p2):
    presun disk z p3 na
p2
    Hanoi(1, p2, p3, p1):
    presun disk z p2 na
p1
```

5

```
Hanoi(1, p1, p2, p3):
    presun disk z p1 na
p3
```



Ukázka převodu - Hanojské věže

```
Hanoi(3, p1, p2, p3):
  if n>0:
    Hanoi(2, p1, p3, p2)
    presun disk z p1 na p3
    Hanoi(2, p2, p1, p3)
```

4

```
Hanoi(2, p1, p3, p2):
  if n>0:
    Hanoi(1, p1, p2, p3)
    presun disk z p1 na p2
    Hanoi(1, p3, p1, p2)
```

2

```
Hanoi(2, p2, p1, p3):
  if n>0:
    Hanoi(1, p2, p3, p1)
    presun disk z p2 na p3
    Hanoi(1, p1, p2, p3)
```

6

```
Hanoi(1, p1, p2, p3):
  if n>0:
    Hanoi(0, p1, p3, p2)
    presun disk z p1 na p3
    Hanoi(0, p2, p1, p3)
```

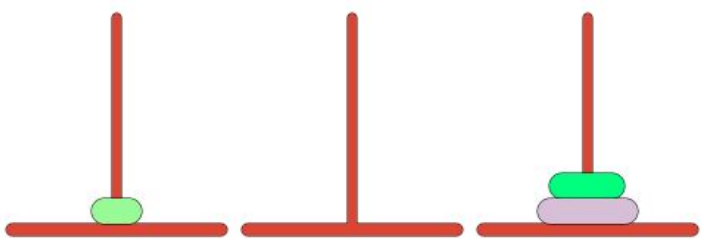
1

3

```
Hanoi(1, p3, p1, p2):
    presun disk z p3 na p2
    Hanoi(1, p2, p3, p1)
```

5

```
    presun disk z p2 na p1
    Hanoi(1, p1, p2, p3):
      presun disk z p1 na p3
```



Ukázka převodu - Hanojské věže

```
Hanoi(3, p1, p2, p3):
  if n>0:
    Hanoi(2, p1, p3, p2)
    presun disk z p1 na p3
    Hanoi(2, p2, p1, p3)
```

4

```
Hanoi(2, p1, p3, p2):
  if n>0:
    Hanoi(1, p1, p2, p3)
    presun disk z p1 na p2
    Hanoi(1, p3, p1, p2)
```

2

```
Hanoi(2, p2, p1, p3):
  if n>0:
    Hanoi(1, p2, p3, p1)
    presun disk z p2 na p3
    Hanoi(1, p1, p2, p3)
```

6

```
Hanoi(1, p1, p2, p3):
  if n>0:
    Hanoi(0, p1, p3, p2)
    presun disk z p1 na p3
    Hanoi(0, p2, p1, p3)
```

1 3

```
Hanoi(1, p3, p1, p2):
    presun disk z p3 na p2
    Hanoi(1, p2, p3, p1)
```

5

```
Hanoi(1, p1, p2, p3):
    presun disk z p1 na p3
```

7

Ukázka převodu - Hanojské věže

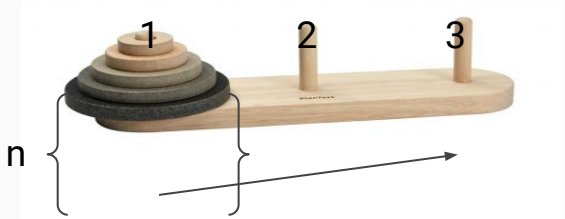
T(1) = 1

T(2) = T(1) + 1 + T(1) = 3

T(3) = T(2) + 1 + T(2) = 7

...

T(n) = 2T(n-1) + 1




```
Hanoi(n, p1, p2, p3):
  if n>0:
    Hanoi(n-1, p1, p3, p2)
    presun disk z p1 na p3
    Hanoi(n-1, p2, p1, p3)
```

Přímé vyjádření?

T(n) = A) $n^2 - 1$ B) $2^n - 1$ C) n*2 + 1

Ukázka převodu - Hanojské věže

T(1) = 1

T(2) = 2T(1) + 1 = 3

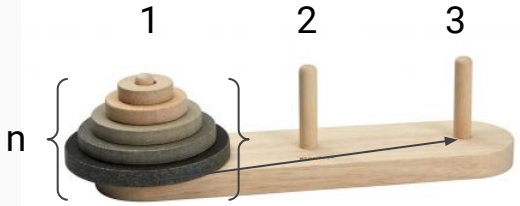
T(3) = 2T(2) + 1 = 7

T(4) = 2T(3) + 1 = 15

T(5) = 2T(4) + 1 = 31

Přímé vyjádření?

T(n) = A) $n^2 - 1$ C) n*2 + 1

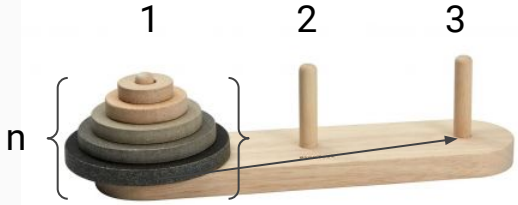



```
Hanoi(n, p1, p2, p3):
  if n>0:
    Hanoi(n-1, p1, p3, p2)
    presun disk z p1 na p3
    Hanoi(n-1, p2, p1, p3)
```

Ukázka převodu - Hanojské věže

Přímé vyjádření

$T(n) = \mathbf{2^n - 1}$




```
Hanoi(n, p1, p2, p3):
  if n>0:
    Hanoi(n-1, p1, p3, p2)
    presun disk z p1 na p3
    Hanoi(n-1, p2, p1, p3)
```

Co tedy platí?

A) Hanoi $\in O(n^2)$
B) Hanoi $\in O(n)$
C) Hanoi $\in O(2^n)$

Převod rekurence na přímé vyjádření

Přímé vyjádření, tj. bez rekurence/opětovného výskytu

- **Substitucí**
  Odhadneme řešení a induktivně dokážeme
- **Rekurzivním stromem**
  Spočítáme složitost rekurzivního stromu
- **Použitím “kuchařky”**
  Mistrovská věta (Master theorem), některé tvary mají známá řešení

Substituce

1. **Odhad**
   (např. testováním pro různá n)
2. **Ověření indukcí**
   (či jinou rigorózní metodou)

Substituce, příklad

$T(n) = 2T(n-1) + 1$

Odhad: $T(n) = O(n^2)$, tedy $T(n) \leq cn^2$

Nechť platí odhad pro menší: $T(n-1) \leq c(n-1)^2$

Indukcí a dosazením do rekurentního vztahu:

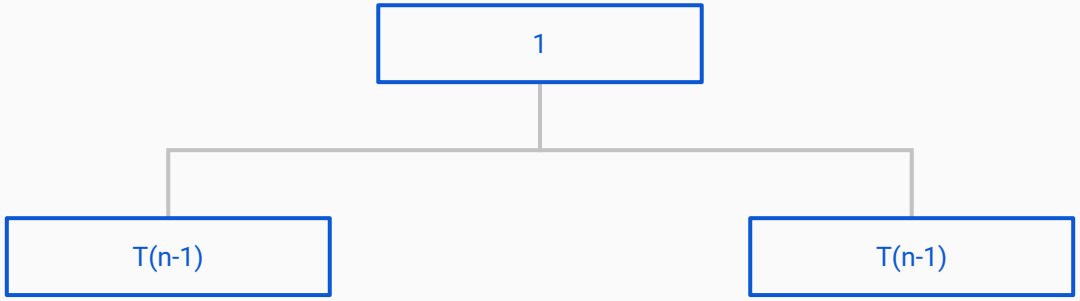
$T(n) \leq 2c(n-1)^2 + 1 = 2cn^2 - 4cn + 2c + 1 = cn^2 - 2cn + c + 1 \leq cn^2$

$T(n) \leq cn^2$

Rekurzivní strom

Iterativně rozložíme problém do rekurzivních stromů.

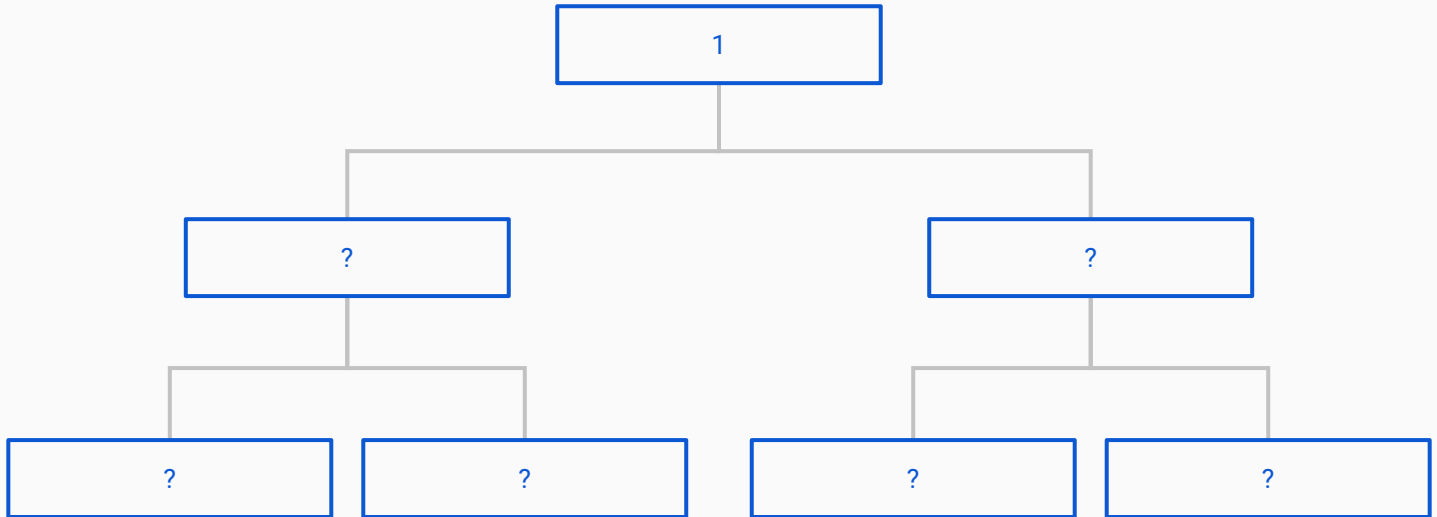
T(n) = 2T(n-1) + 1



Rekurzivní strom

Iterativně rozložíme problém do rekurzivních stromů.

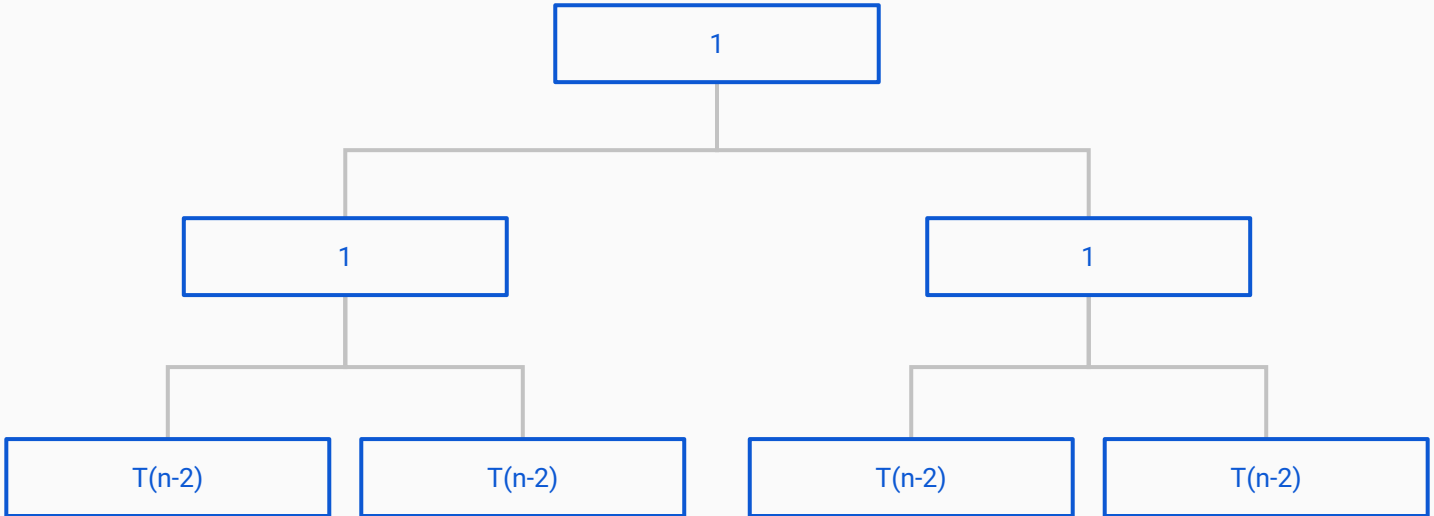
T(n) = 2T(n-1) + 1



Rekurzivní strom

Iterativně rozložíme problém do rekurzivních stromů.

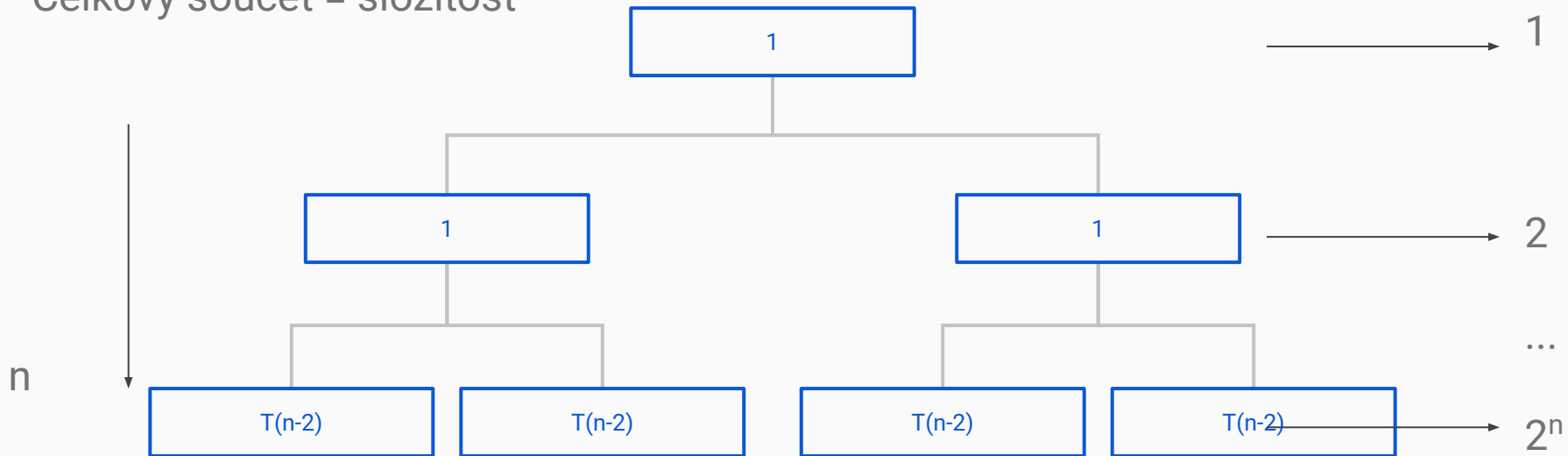
T(n) = 2T(n-1) + 1



Rekurzivní strom

T(n) = 2T(n-1) + 1

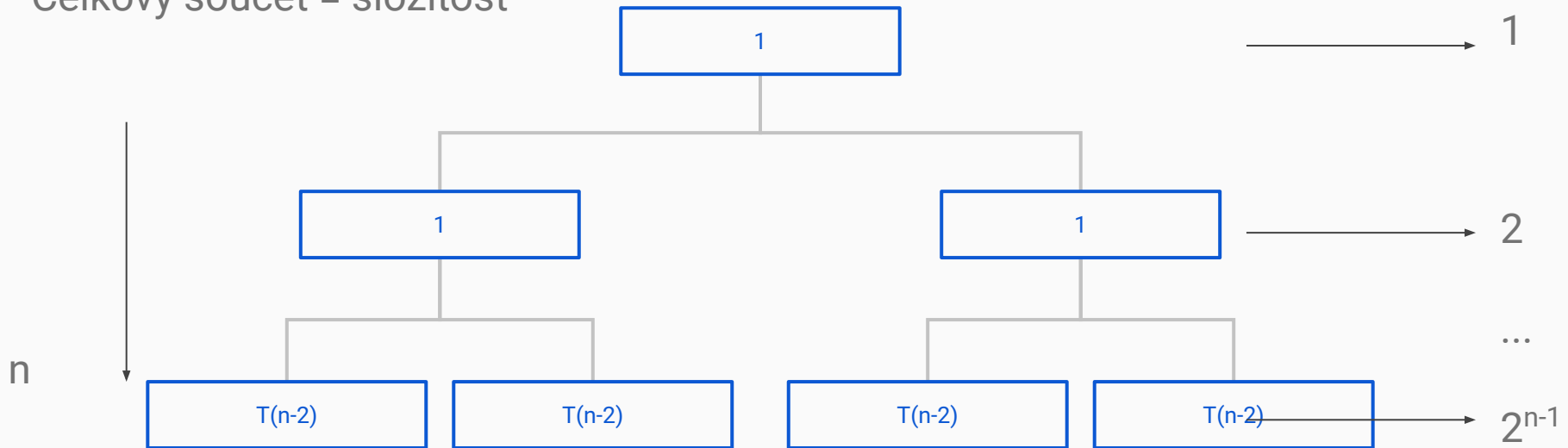
- Součty složitosti v patře
- Celkový součet = složitost



Rekurzivní strom

$T(n) = 2T(n-1) + 1$ … $T(n) = 2^0 + 2^1 + 2^2 + \dots + 2^{n-1} = 2^n - 1$

- Součty složitosti v patře
- Celkový součet = složitost



Kuchařka dle Master theorem (mistrovské věty)

Rekurentní složitost tvaru:

T(n) = aT(n/b) + f(n)

a >= 1, b > 1, f(n) asymptoticky kladná funkce

Kuchařka dle Master theorem (mistrovské věty)

1. Pokud $f(n) \in O(n^{\log_b(a)-\varepsilon})$ pro nějakou konstantu $\varepsilon > 0$, potom

   $T(n) \in \Theta(n^{\log_b(a)})$.

2. Pokud $f(n) \in \Theta(n^{\log_b(a)})$, potom

   $T(n) \in \Theta(n^{\log_b(a)} \log(n))$.

3. Pokud $f(n) \in \Omega(n\log_b(a)+\varepsilon)$ pro nějakou konstantu $\varepsilon > 0$ a pokud a $f(n/b) \leq c\, f(n)$ pro nějakou konstantu $c < 1$ a všechna dostatečně velké n, potom

   $T(n) \in \Theta(f(n))$.

Mistrovská věta, příklad

T(n) = 2T(n/2) + n

a = 2, b = 2, f(n) = n $\in O(n^{\log_2(2)})$

Tedy druhý případ neboť $\log_2(2) = 1$ a cokoliv menšího je tedy horší než n.

Tedy:

$T(n) \in \Theta(n^{\log_b(a)}\log(n)) = \Theta(n^{\log_2(2)}\log(n)) = \Theta(n \log(n))$

Bonus: Pozor na Python

Python je lehký na úvod, ale stále běží na počítači a může tedy mít některé závislosti (např. stále rozlišuje mezi daty samotnými a odkazováním na ně).

Pro zábavu se můžete podívat na https://github.com/satwikkansal/wtfpython