



# Algoritmizace

Marko Genyk-Berezovskyj, Daniel Průša

2010 - 2023

## Rekurzivní algoritmy

- Programovací technika rozděl a panuj
- Odvození časové složitosti

slido

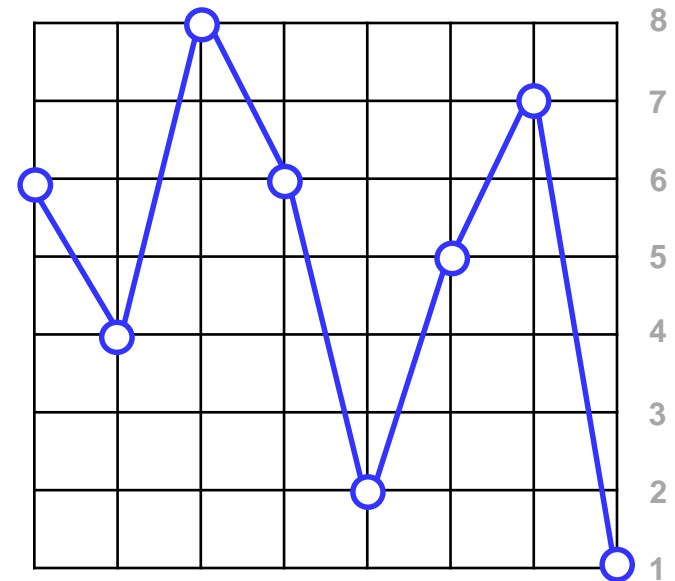
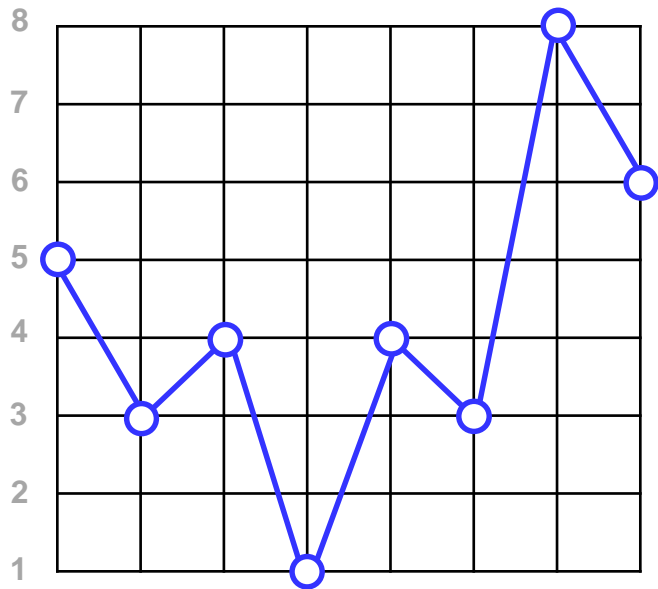


**Join at [slido.com](https://slido.com)  
#1526493**

① Start presenting to display the joining instructions on this slide.

# Nákup a prodej akcií

Predikce vývoje ceny  
v čase:



Úloha: Chceme naplánovat jeden nákup a jeden prodej tak, aby zisk byl maximální možný.

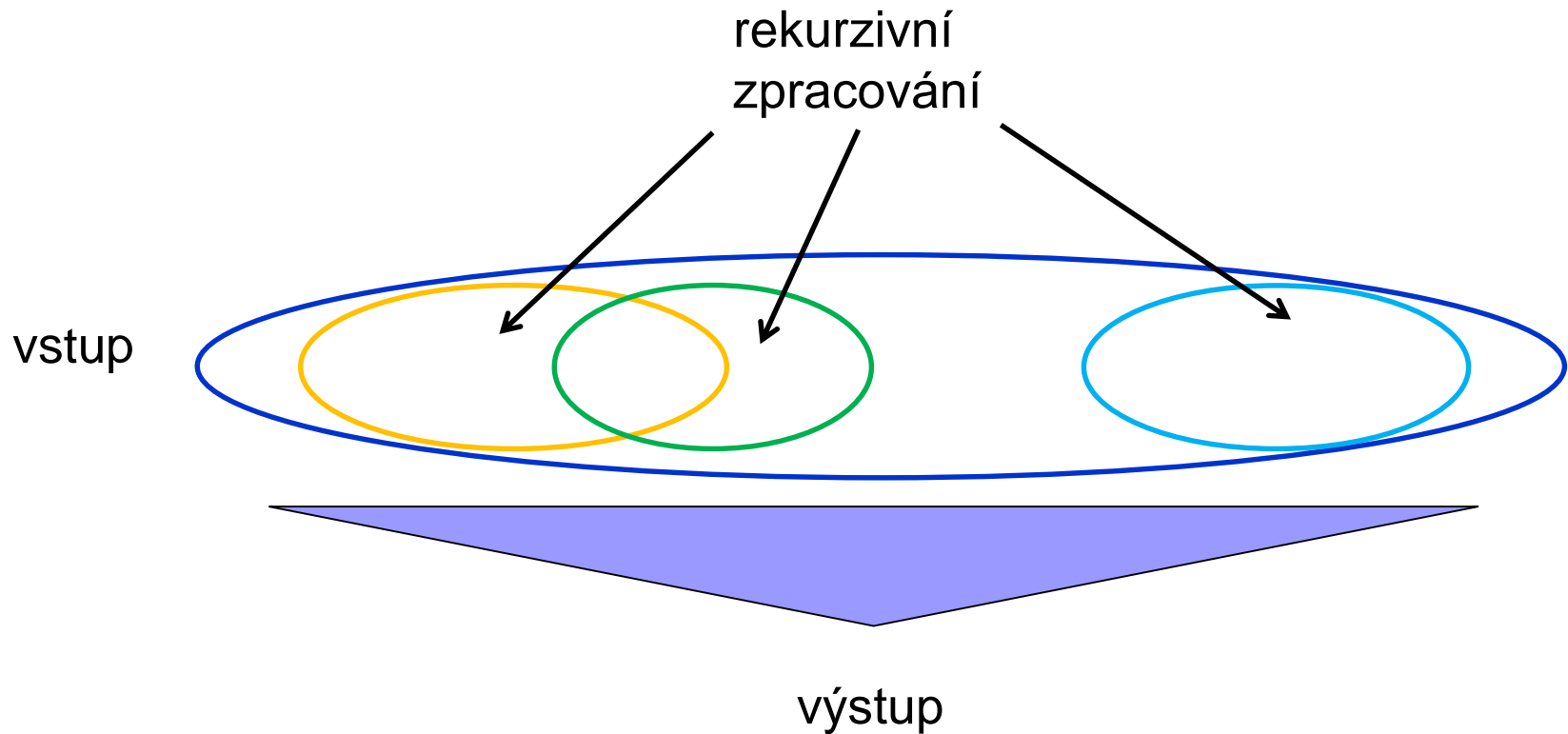
# Nákup a prodej akcií

Rekurzivní algoritmus:

```
MaxProfit(int low, int hi, int[] cena):  
  if low == hi then  
    return 0;  
  else  
    mid = (low + hi) / 2;  
    profit_1 = MaxProfit(low, mid, cena);  
    profit_2 = MaxProfit(mid + 1, hi, cena);  
    profit_3 = MAX(mid + 1, hi, cena) -  
              MIN(low, mid, cena);  
    return MAX3(profit_1, profit_2, profit_3);  
  endif
```

# Rozděl a panuj

- Divide and conquer, divide et impera
- Programovací technika



# Rozděľ a panuj

- Třídění
- Syntaktická analýza (top-down parsing)
- Násobení velkých čísel
- Násobení matic
- Diskrétní Fourierova transformace

# Nákup a prodej akcií

Rekurzivní algoritmus:

```
MaxProfit(int low, int hi, int[] cena):  
  if low == hi then  
    return 0;  
  else  
    mid = (low + hi) / 2;  
    profit_1 = MaxProfit(low, mid, cena);  
    profit_2 = MaxProfit(mid + 1, hi, cena);  
    profit_3 = MAX(mid + 1, hi, cena) -  
              MIN(low, mid, cena);  
    return MAX3(profit_1, profit_2, profit_3);  
  endif
```

Časovou složitost vyjádříme rekurentně:

$$T(1) = \theta(1)$$

$$T(n) = T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + \theta(n), \quad n > 1$$



slido



## Audience Q&A Session

① Start presenting to display the audience questions on this slide.

# Převod rekurence na přímé vyjádření

- Přímým vyjádřením složitosti myslíme vyjádření složitosti bez rekurence.
  - Např:  $T(n) \in \Theta(\log n)$
- Jaké jsou možnosti řešení?
  - **Substituční metoda**
    - „Uhádneme“ řešení a potom dokážeme, že je správné indukcí.
  - **Metoda rekurzivního stromu**
    - Spočítáme složitost celého rekurzivního stromu.
  - **Použití „kuchařky“** (Master theorem – mistrovská věta)
    - Pro některé speciální tvary rekurentních vztahů známe předem vypočítané řešení dle mistrovské věty.

# Substituční metoda

- Řešíme ve dvou krocích
  1. **Odhadneme přesný tvar řešení.**
    - Odhad lze stanovit například pomocí zjišťováním složitosti pro různá vstupní  $n$ .
  2. **Matematicky dokážeme, že je náš odhad správný.**
    - Obvykle se dokazuje pomocí matematické indukce.
- Metoda bývá zpravidla velmi účinná.
- Její nevýhodou je určování přesného tvaru řešení v kroku 1 pro které neexistuje obecný postup.

# Substituční metoda

$$T(1) = \Theta(1)$$

$$T(n) = T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + \Theta(n), \quad n > 1$$

Zkusme odhadnout, jakou má  $T(n)$  asymptotickou složitost.

- A.  $T(n) \in \Theta(n)$
- B.  $T(n) \in \Theta(n \log n)$
- C.  $T(n) \in \Theta(n^2)$
- D.  $T(n) \in \Theta(2^n)$

slido



# Prodej akcií - jaká je časová složitost?

ⓘ Start presenting to display the poll results on this slide.

# Substituční metoda

- $T(1) = d$

$$T(n) = T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + d \cdot n, \quad n > 1$$

- Nejprve mat. indukcí dokážeme  $\forall k \geq 1: T(2^k) \leq 2dk2^k$

- Ověříme platnost tvrzení pro  $k = 1$ :

$$T(2^1) = T(1) + T(1) + 2d = 4d$$

- A s použitím indukčního předpokladu  $T(2^k) \leq 2dk2^k$  odvodíme

$$\begin{aligned} T(2^{k+1}) &= T(2^k) + T(2^k) + d2^{k+1} \leq 2dk2^{k+1} + d2^{k+1} \\ &= (2k + 1)d2^{k+1} \leq 2(k + 1)d2^{k+1} = 2d(k + 1)2^{k+1} \end{aligned}$$

# Substituční metoda

- Nyní výsledek zobecníme pro každé  $n \geq 2$ .
- Pro takové  $n$  existuje mocnina dvojky  $n'$  splňující

$$n \leq n' < 2n$$

- $T(n)$  je neklesající funkce, platí tedy

$$T(n) \leq T(n') \leq 2dn' \log_2 n' < 4dn \log_2(2n) \in O(n \log n)$$

slido



## Audience Q&A Session

① Start presenting to display the audience questions on this slide.

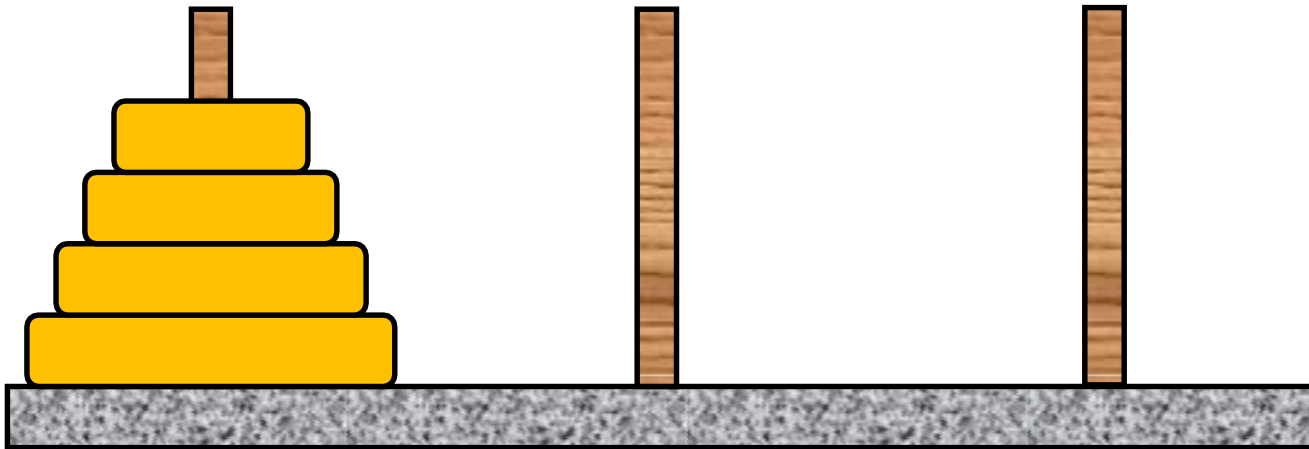


# Hanojská věž

Úkol: přemístit disky z tyče vlevo na tyč vpravo

Pravidla:

- disky přesouváme pouze jednotlivě, z tyče na tyč
- větší disk nesmí nikdy ležet na menším disku



# Hanojská věž

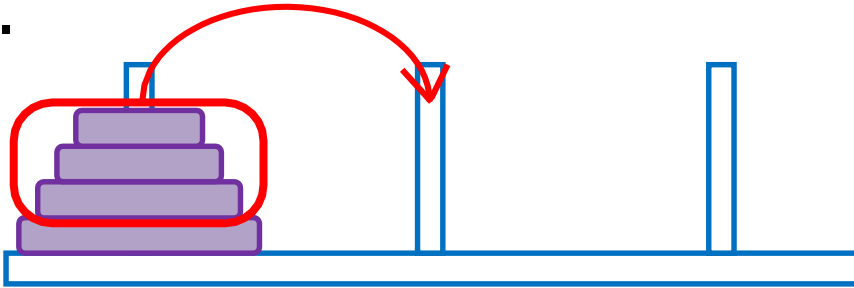


<https://www.youtube.com/watch?v=CBM1zE9vFnE>

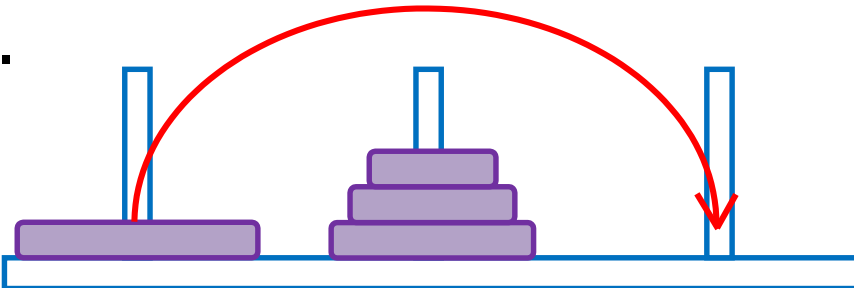
# Hanojská věž

Rekurzivní algoritmus:

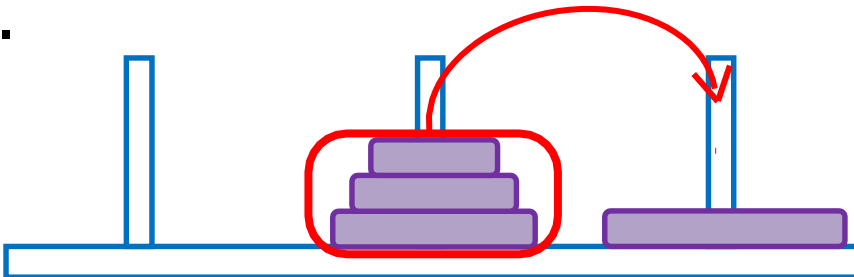
1.



2.



3.



odkud, přes, kam

Hanoj(n, t1, t2, t3):

**if** n > 0 **then**

Hanoj(n-1, t1, t3, t2);

přesuň disk z t1 na t3;

Hanoj(n-1, t2, t1, t3);

**endif**

Hanoj(4, 1, 2, 3);

# Hanojská věž

Časová složitost je úměrná počtu tahů (přesunů), které provedeme.

```
Hanoj(n, t1, t2, t3):  
if n > 0 then  
    Hanoj(n-1, t1, t3, t2);  
    přesuň disk z t1 na t3;  
    Hanoj(n-1, t2, t1, t3);  
endif
```

$$T(1) = 1,$$

$$T(n) = 2T(n - 1) + 1, \quad n > 1.$$

# Substituční metoda

- $T(1) = 1,$   
 $T(n) = 2T(n - 1) + 1, \quad n > 1.$

Zkusme odhadnout asymptotickou složitost funkce  $T(n)$ .

- A.  $T(n) \in \Theta(n)$
- B.  $T(n) \in \Theta(n \log n)$
- C.  $T(n) \in \Theta(n^2)$
- D.  $T(n) \in \Theta(2^n)$

slido



# Hanojská věž - jaká je časová složitost?

ⓘ Start presenting to display the poll results on this slide.

# Substituční metoda

- $T(1) = 1,$   
 $T(n) = 2T(n - 1) + 1, \quad n > 1.$

Dokážeme  $T(n) = 2^n - 1$

- Pro  $n = 1$ :

$$2^1 - 1 = 1 = T(1)$$

- Pro  $n > 1$ :

$$\begin{aligned} T(n + 1) &= 2T(n) + 1 = 2(2^n - 1) + 1 = 2^{n+1} - 2 + 1 \\ &= 2^{n+1} - 1 \end{aligned}$$

slido



## Audience Q&A Session

① Start presenting to display the audience questions on this slide.

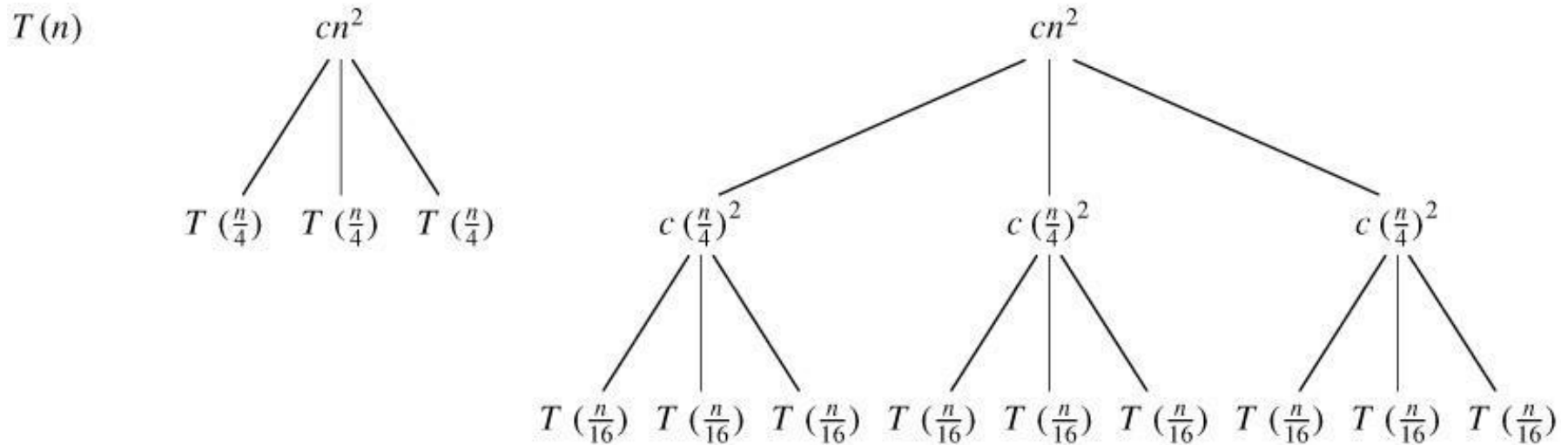


# Metoda rekurzivního stromu

- Příklad:

$$T(n) = 3 \cdot T\left(\frac{n}{4}\right) + cn^2$$

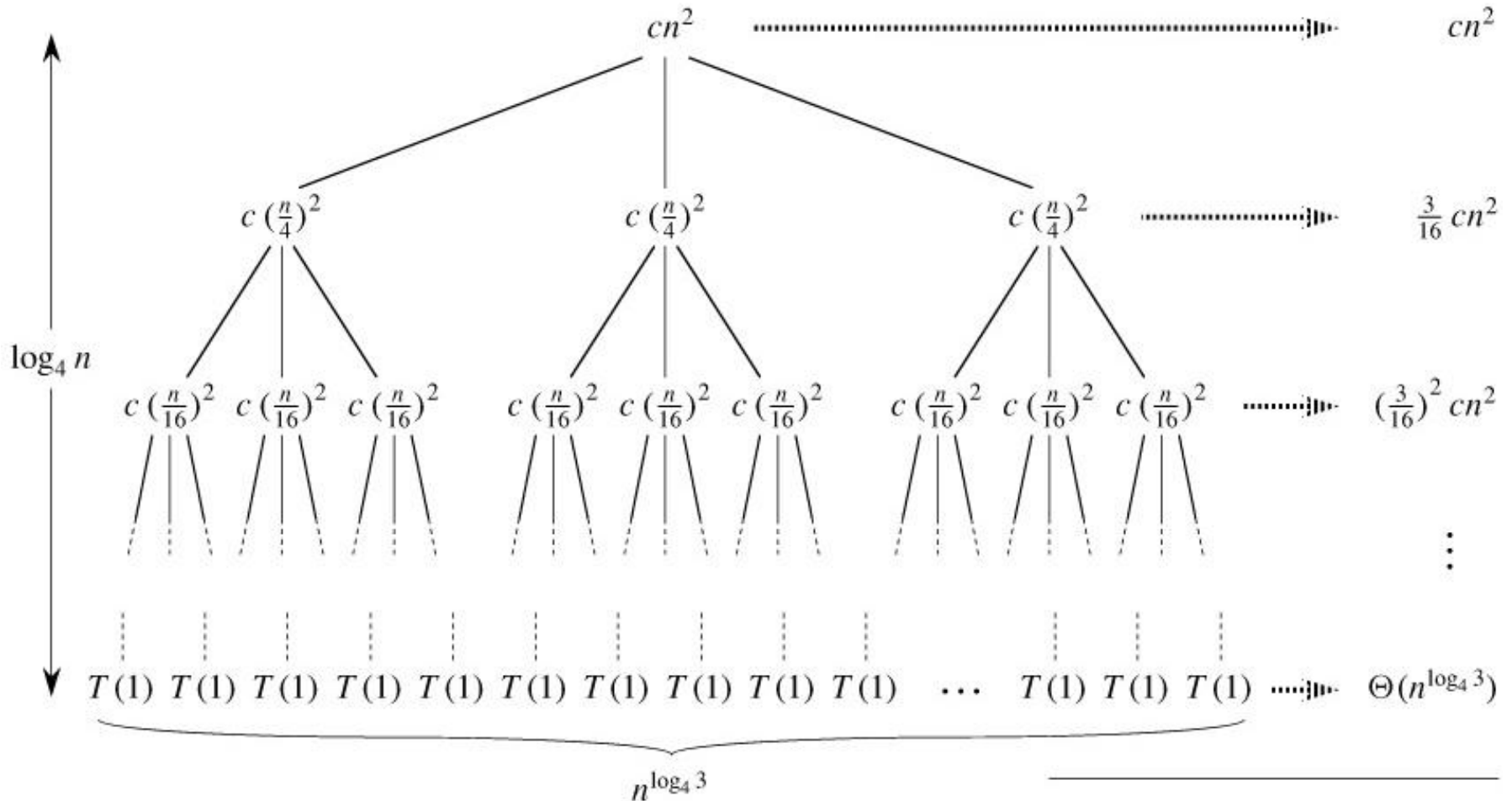
- Iterativně rozkládáme do rekurzivních stromů:



- Pro každý strom platí, že součet všech uzlů dá složitost  $T(n)$  podle původního rekurentního vztahu.
- Rekurzivní stromy jsou pouze grafická vizualizace rozvoje rekurentního vztahu.

# Metoda rekurzivního stromu

- Výsledný strom má následující tvar:



- Vyjádříme součty v patrech a patra sečteme:

$$O(n^2)$$

# Metoda rekurzivního stromu

- Výsledný rekurzivní strom má  $3^{\log_4 n}$  listů, z čehož odvodíme

$$\begin{aligned} 3^{\log_4 n} &= \left(4^{\log_4 3}\right)^{\log_4 n} = 4^{(\log_4 3) \cdot (\log_4 n)} \\ &= \left(4^{\log_4 n}\right)^{\log_4 3} = n^{\log_4 3} \end{aligned}$$

# Metoda rekurzivního stromu

- Součet pater spočítáme následovně:

$$\begin{aligned} T(n) &= cn^2 + \frac{3}{16}cn^2 + \left(\frac{3}{16}\right)^2 cn^2 + \dots + \left(\frac{3}{16}\right)^{\log_4 n - 1} cn^2 + \Theta(n^{\log_4 3}) \\ &= \sum_{i=0}^{\log_4 n - 1} \left(\frac{3}{16}\right)^i cn^2 + \Theta(n^{\log_4 3}) \\ &< \sum_{i=0}^{\infty} \left(\frac{3}{16}\right)^i cn^2 + \Theta(n^{\log_4 3}) \\ &= \frac{1}{1 - (3/16)} cn^2 + \Theta(n^{\log_4 3}) \\ &= \frac{16}{13} cn^2 + \Theta(n^{\log_4 3}) \\ &= O(n^2). \end{aligned}$$

podle vzorce  $\sum_{i=0}^{\infty} q^i = \frac{1}{1-q}$   
kde  $0 < q < 1$



# Audience Q&A Session

① Start presenting to display the audience questions on this slide.

# Mistrovská věta

- Master theorem nebo také kuchařková věta
- Řeší složitost rekurentní funkce tvaru

$$T(n) = a T(n/b) + f(n)$$

kde  $a \geq 1$  a  $b > 1$  jsou konstanty

a  $f(n)$  je asymptoticky kladná funkce.

- Zaokrouhlení u členu  $T(n/b)$  na  $T(\lfloor n/b \rfloor)$  nebo  $T(\lceil n/b \rceil)$  neovlivní v tomto případě výslednou složitost.
- Lze aplikovat na Mergesort, binární vyhledávání, hledání mediánu, Strassenův algoritmus pro násobení matic, ...

# Mistrovská věta

Nechť jsou  $a \geq 1$  a  $b > 1$  konstanty, necht' je  $f(n)$  asymptoticky kladná funkce a necht'  $T(n)$  je definováno pro nezáporná celá čísla rekurencí  $T(n) = a T(n/b) + f(n)$ , kde  $n/b$  má význam buď  $\lfloor n/b \rfloor$  nebo  $\lceil n/b \rceil$ . Potom lze  $T(n)$  asymptoticky vyjádřit následovně:

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 velká  $n$ , potom

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

# Použití „kuchařky“ – příklad 1

- Příklad 1 (Strassenův algoritmus):

$$T(n) = 7 \cdot T(n/2) + \Theta(n^2)$$

- Z toho dostáváme, že  $a = 7$ ,  $b = 2$ ,  $f(n) \in O(n^{\log_2(7)-0.5})$ .  
Jedná se tedy o případ číslo 1.
- Dostáváme složitost:

$$T(n) \in \Theta(n^{\log_2(7)}) \subseteq \Theta(n^{2.8074})$$



# Použití „kuchařky“ – příklad 2

- Příklad 2 (Nákup a prodej akcií):

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

- Z toho dostáváme, že  $a = 2$ ,  $b = 2$ ,

$$f(n) \in \Theta(n) = \Theta(n^{\log_2(2)}).$$

Jedná se tedy o případ číslo 2.

- Dostáváme složitost:

$$T(n) \in \Theta(n \cdot \log(n))$$

# Použití „kuchařky“ – příklad 3

- Příklad 3 (fiktivní):

$$T(n) = 3 \cdot T(n/4) + n \log(n)$$

- Z toho dostáváme, že  $a = 3$ ,  $b = 4$ ,

$$f(n) = n \log(n) \text{ a víme, že } n^{\log_4(3)} = O(n^{0.793}) .$$

Platí tedy, že  $f(n) \in \Omega(n^{\log_4(3)+0.2})$ .

Pokud by se mělo jednat o případ 3 musí ještě platit pro  $c < 1$  a všechna dostatečně velká  $n$ , že  $a f(n/b) \leq c f(n)$  tedy  $a f(n/b) = 3(n/4)\log(n/4) \leq (3/4)n \log(n) = c f(n)$  pro  $c = 3/4$ .

- Dostáváme složitost:

$$T(n) \in \Theta(n \log(n))$$

slido

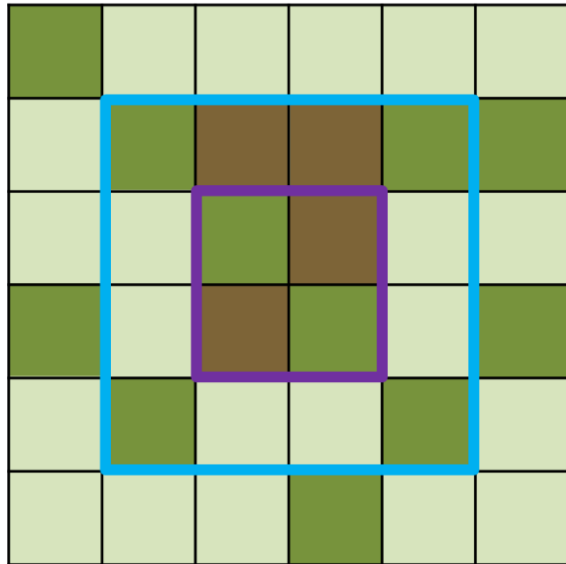


## Audience Q&A Session

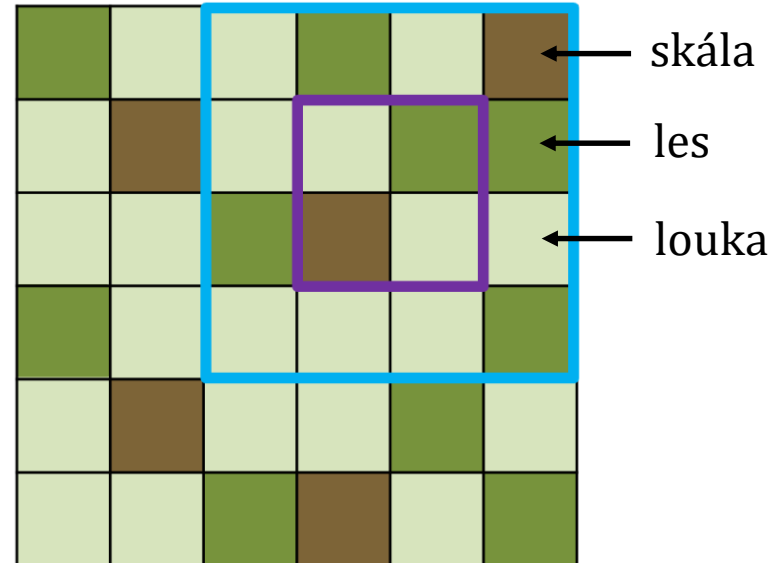
① Start presenting to display the audience questions on this slide.

# Domácí úloha

a)  $N = 6, K = 4, S = 2$



b)  $N = 6, K = 4, S = 1$



Plocha  $N \times N$ .

Hledáme v ní plochu  $K \times K$ , která obsahuje maximální počet lesů a jejíž centrální část má alespoň  $S$  skal.

# Prefixové součty

A ... pole čísel indexované od 1 do N

P ... pole prefixových součtů (v A) indexované od 0 do N

$P[i] = A[1] + \dots + A[i]$  - prefixový součet pro úsek délky i

$P[0] = 0$

Příklad:

$A = \{1, -2, 4, 5, -1, -5, 2, 7\}$

$P = \{0, 1, -1, 3, 8, 7, 2, 4, 11\}$

Každý součet  $A[i] + A[i+1] + \dots + A[j-1] + A[j]$  lze určit v konstantním čase, je roven  $P[j] - P[i-1]$ .

Viz programátorská kuchařka:

<https://ksp.mff.cuni.cz/kucharky/zakladni-algoritmy/>

slido



**Jak dlouho jste řešili první domácí úkol?**

ⓘ Start presenting to display the poll results on this slide.

slido



# Audience Q&A Session

① Start presenting to display the audience questions on this slide.