Procházení stromem

Základy algoritmizace – 4. cvičení

Zuzana Petříčková

20. března 2020

Mazání vrcholu v binárním vyhledávacím stromě

Princip:

1. najdu vrchol u, který chci smazat (a jeho otce o)
2. najdu náhradníka n (a jeho otce on)
3. převěsím ukazatele (alternativně: překopíruji data z n do u)
4. smažu vrchol u (pro alternativní řešení: smažu vrchol n)

Jaký vrchol bude náhradník?

- typicky to bude nejmenší větší prvek ve stromě (alternativně: největší menší)

Kde náhradníka najdu?

- bude to nejlevější vrchol v pravém podstromu vrcholu u

Mazání vrcholu v binárním vyhledávacím stromě

Varianty:

1. vrchol u nemá syny $\rightarrow$ prostě ho smažu
2. vrchol u má jednoho syna (levého nebo pravého) $\rightarrow$ u nahradím tímto synem
3. vrchol u má oba syny $\rightarrow$ u nahradím nejmenším větším prvkem ve stromě (bude to nejlevější vrchol v pravém podstromu u)
   1. najdu náhradníka n (a jeho otce on)
   2. převěsím ukazatele (alternativně: překopíruji data z n do u)
      - zvlášť je třeba ošetřit speciální případ, kdy n je pravým synem u. V tom případě novým synem vrcholu o bude místo u vrchol n (levého syna převezme od u, pravý syn mu zůstane původní)
      - Jinak: novým synem vrcholu o bude místo u vrchol n (n převezme levého a pravého syna od u) a novým synem vrcholu on bude místo n původní pravý syn n (levého syna vrchol n neměl)
   3. smažu vrchol u (pro alternativní řešení: smažu vrchol n)

Procházení binárním stromem do hloubky (BACKTRACKING) ... rekurzivně:

PREORDER

```
zpracuj( k ) ... vrchol k je kořen podstomu
   if (podstrom s kořenem k není prázdný) then
      zpracuj vrchol k
      zpracuj ( levý podstrom k ) ... rekurzivni volání
      zpracuj ( pravý podstrom k ) ... rekurzivni volání
```

INORDER

```
zpracuj1(k)
   if (podstrom s kořenem k není prázdný) then
      zpracuj1( levý podstrom k )
      zpracuj vrchol k
      zpracuj1( pravý podstrom k )
```

POSTORDER

```
zpracuj2( k )
   if (podstrom s kořenem k není prázdný) then
      zpracuj2 ( levý podstrom k )
      zpracuj2 ( pravý podstrom k )
      zpracuj vrchol k
```

Procházení binárním stromem do šířky (tj. po vrstvách) ... rekurzivně:

- nejprve chceme zpracovat (např. vypsat) všechny vrcholy na první hladině (vrstvě), pak všechny na druhé, atd.

Jak na to?

- vytvoříme si pomocnou funkci **zpracujR**, která zpracuje právě všechny vrcholy na hladině i
- pak ji pustíme postupně pro $i = 1, 2, ..., h$, kde h je počet hladin stromu

Jak implementovat zpracujR?

- jde o klasický rekurzivní průchod stromem do hloubky, ale hloubku si omezíme (stačí nám zastavit se na i-té hladině)
- zavedeme si počitadlo hladin, podobné, jako jsme měli ve funkci **vypisOdsazene**

Procházení binárním stromem do šířky (tj. po vrstvách) ... rekurzivně:

zpracuj (strom s kořenem k)
  spočítej hloubku stromu h
  for i = 1 **to** h **do**
    zpracujR (podstrom s kořenem k, i) ... zpracuj i-tou hladinu stromu

zpracujR (podstrom s kořenem k, i)
  if podstrom je prázdný **then**
    return
  if i je rovno 1 **then**
    zpracuj vrchol k
  else
    zpracujR (levý podstrom k, i-1)
    zpracujR (pravý podstrom k, i-1)

Jak spočítat počet hladin?

- rekurzivním průchodem stromem do hloubky
- nebo efektivněji: viz následující dva snímky

Procházení binárním stromem do šířky (tj. po vrstvách) ... rekurzivně (efektivnější varianta):

Musíme počítat počet hladin předem?

- nemusíme, stačí nám, pokud funkce **zpracujR** vrátí informaci o tom, zda další hladina existuje:

```
zpracuj1 ( strom s kořenem k )
   pokracuj = true
   i = 1
   while pokracuj do
     pokracuj = zpracujR1 (podstrom s kořenem k, i) ... zpracuj i-tou hladinu stromu
     i zvětšíme o 1
  od
```

Procházení binárním stromem do šířky (tj. po vrstvách) ... rekurzivně (efektivnější varianta):

zpracujR1 (podstrom s kořenem k, i)
[funkce vrací true nebo false] ... true značí, že v podstromu je aspoň jeden vrchol na hladině i

```
if podstrom je prázdný then
  return false
if i je rovno 1 then
  zpracuj vrchol k
  return true
else
  pokracuj = false
  if zpracujR1 ( levý podstrom k, i-1 ) then
    pokracuj = true
  if zpracujR1 ( pravý podstrom k, i-1 ) then
    pokracuj = true
  return pokracuj
```

Výpis vrcholů stromu po jednotlivých vrstvách

- použijeme procházení stromem do šířky

Co kdybychom chtěli vypisovat vrcholy odsazeně, aby byla vidět struktura stromu?

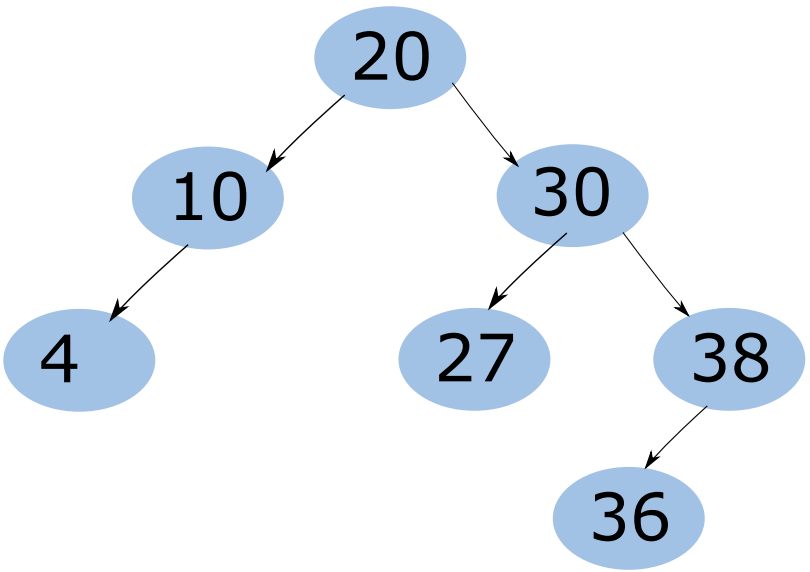
- předpokládáme, že klíč každého vrcholu se vypíše pomocí 3 znaků

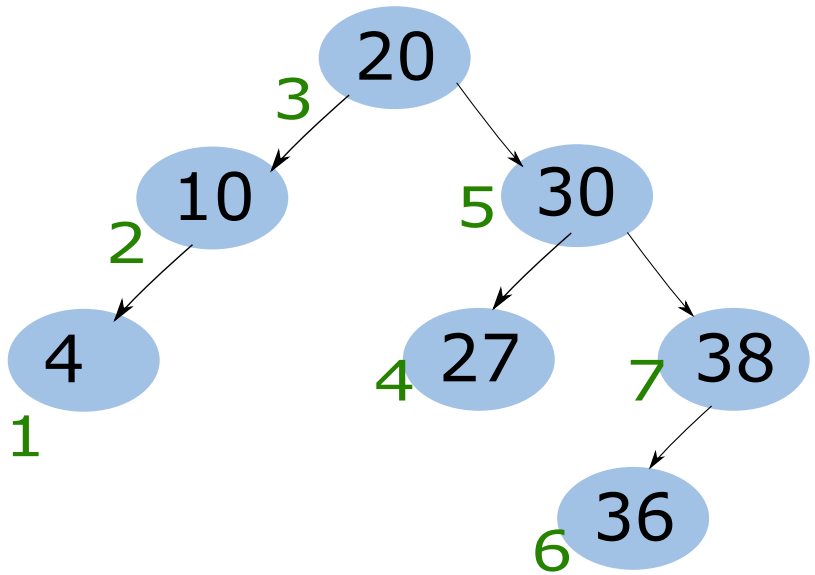
A) Kompaktní výpis

- každý vrchol bude vypsán ve svém sloupečku šířky 4
- pro každý vrchol si spočítáme jeho pořadí zleva
- pokud je vrchol i-tý zleva, odsadíme ho o $4 * i$ mezer (vzato od začátku řádku)
- pořadí vrcholů si můžeme předpočítat rekurzivním průchodem stromem do hloubky (jen se nám o index i ,,rozroste" struktura vrcholu), nebo si ho budeme počítat průběžně (za cenu časové složitosti navíc)

Výpis vrcholů stromu po jednotlivých vrstvách - kompaktní výpis

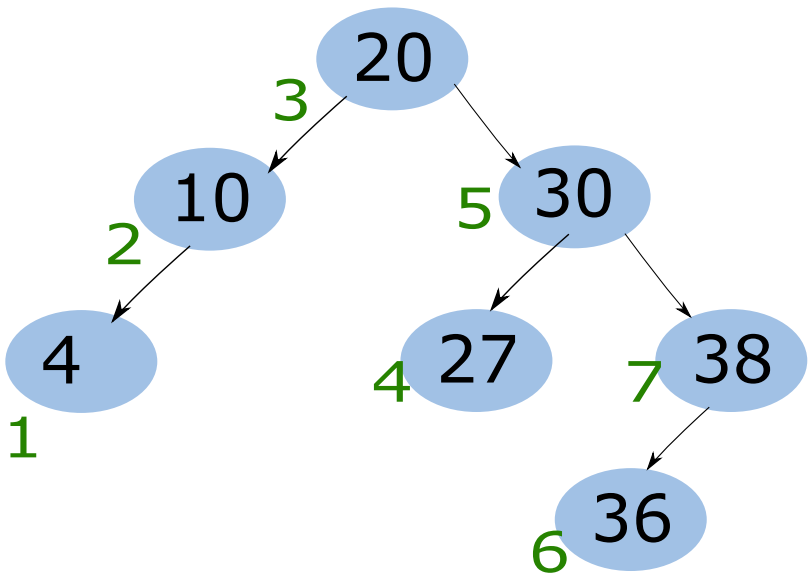
- pro každý vrchol si spočítáme jeho pořadí zleva
  - pořadí vrcholů si můžeme předpočítat rekurzivním průchodem stromem do hloubky (INORDER), jen se nám o index i „rozroste" struktura vrcholu)

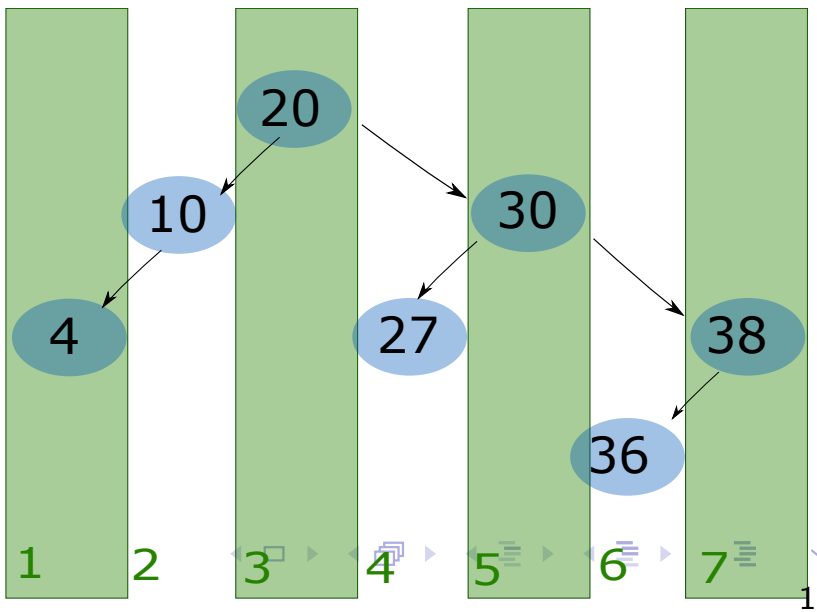




Výpis vrcholů stromu po jednotlivých vrstvách - kompaktní výpis

- Máme n sloupečků o šířce 4,
- i-tý vrchol zleva vytiskneme v i-tém sloupečku, tj. odsadíme ho o $4 * i$ mezer (vzato od začátku řádku, při vypisování si tedy musíme pamatovat, který sloupeček na daném řádku jsme minule zaplnili)





Výpis vrcholů stromu po jednotlivých vrstvách

B) Široký výpis

- pro strom o hloubce h budeme vypisovat strom tak, jako by byl ,,dokonalý" (plně zaplněný)
- vykreslovaný strom bude symetrický za cenu prázdných sloupečků pro chybějící vrcholy
- ve stromě o hloubce h může být až $2^h - 1$ vrcholů, vlevo od kořene jich bude $2^{h-1} - 1$
  $\rightarrow$ kořen odsadíme o $s = 4 * (2^{h-1} - 1)$ mezer
- jeho syny odsadíme zhruba o $t_L = s - s/2$ a $t_R = s + s/2$ mezer (vzato od začátku řádku)
- atd. (stejný postup rekurzivně) pro další úrovně stromu