Algoritmy 2 - randomizované struktury

Petr Osička

KATEDRA INFORMATIKY
UNIVERZITA PALACKÉHO V OLOMOUCI

Nejdříve o binárních vyhledávacích stromech (BVS).

Uvažujeme množinu U vrcholů s různými klíči, $|U| = n$.

Permutace U je posloupnost

$$u_1, \ldots, u_n$$

prvků U taková, že každý prvek z U se v ní nachází právě jednou. Takový posloupností existuje $n!$.

Sestavení BVS metodou náhodných klíčů

1. Navzorkujeme permutaci vrcholů (s uniformní pravděpodobností, každá permutace má pravděpodobnost výběru rovnu $\frac{1}{n!}$).
2. Vezmeme prázdný BVS a vložíme do něj operací `insert` vrcholy z U v pořadí daném permutací z předchozího bodu.

Podmínka kořenové uniformity BVS

(= podmínka, kterou nějaká metoda vzorkování BVS buď splňuje nebo nesplňuje)

1. pro každý vrchol $x \in U$ platí, že pravděpodobnost, že vrchol $x \in U$ je kořenem navzorkovaného stromu, je rovna $\frac{1}{|U|}$.
2. Je-li x kořenem stromu, potom kořenová uniformita platí i pro
   - levý podstrom a množinu $L = \{y \in U \mid y \text{ má menší klíč než } x\}$
   - pravý podstrom a množinu $R = \{y \in U \mid y \text{ má větší klíč než } x\}$

Pozn: jde o induktivní definici. *Koncem rekurze* je situace, kdy $|U| = 1$ nebo $|U| = 0$, kdy je podmínka triviálně splněna.

Věta

Vzorkování metodou sestavení z náhodných klíčů je kořenově uniformní.

Důkaz (kostra). Přep. $|U| = n > 1$.
Uvážíme-li posloupnost $u_1, \ldots, u_n$, pak se kořenem stane vrchol u_1, kořenem levého podstromu se stane první prvek v posloupnosti s menším klíčem než u_1, kořenem pravého podstromu pak první prvek v posloupnosti s větším klíčem než u_1.

Protože permutace vzorkujeme uniformně, je pravděpodobnost, že $x \in U$ se stane kořenem (tj. stane se u_1), rovna

$$\frac{(n-1)!}{n!} = \frac{1}{|U|}.$$

Připomeňme, že pro kořen x, máme množiny

$$L = \{y \in U \mid y \text{ má menší klíč než } x\}$$
$$R = \{y \in U \mid y \text{ má větší klíč než } x\}$$

(*pokračování důkazu*)

Dále vidíme, že levý podstrom závisí pouze na podpermutaci permutace $u_2, \ldots, u_n$, ve které uvažujeme pouze prvky z L. Přitom každá taková konkrétní podpermutace má stejnou pravděpodobnost výskytu (bereme-li pst přes navzorkované permutace prvků z $U - \{x\}$).

Tady ovšem platí, že $y \in L$ je kořenem levého podstromu, pokud je v této podpermutaci na prvním místě. To je ovšem s pravděpodobností

$$\frac{(|L| - 1)!}{|L|!} = \frac{1}{|L|}$$

Podobně bychom dokázali i pravděpodobnost pro pravý podstrom a R.

Průměrná složitost úspěšného vyhledávání

Předpoklad: pravděpodobnost, že vyhledáváme vrchol x je rovna $1/n$, fixní BVS t s n uzly.

Označíme $h_t(x)$ hloubku vrcholu x ve stromu t.

Průměrná složitost (= potřebný počet porovnání) je potom

$$\mathrm{avg}(t) = \frac{1}{n} \sum_{x \in t} h_t(x) + 1$$

Značení $x \in t$ je *přes všechny uzly* x *ve stromu* t.

Průměrná složitost neúspěšného vyhledávání

Předpoklad: pravděpodobnost, že vyhledávání skončí ve vrcholu x je $\frac{1}{m}$, kde m je počet vrcholů s méně než 2 potomky, fixní BVS t s n vrcholy.

$$\overline{\mathrm{avg}}(t) = \frac{1}{m} \sum_{\substack{x \in t \\ x \text{ nemá 2 potomky}}} h_t(x) + 1$$

Opravdové průměrné složitosti

Uvažujeme vzorkování stromů s n vrcholy. Pro každý strom t tedy navíc máme pravděpodobnost $P(t)$, s jakou byl navzorkován.

$$\mathrm{avg}(n) = \sum_{t:|t|=n} P(t) \cdot \mathrm{avg}(t)$$

$$\overline{\mathrm{avg}}(n) = \sum_{t:|t|=n} P(t) \cdot \overline{\mathrm{avg}}(t)$$

Značení $t : |t| = n$ — suma je přes stromy t s n vrcholy.

Věta

Pokud vzorkujeme metodou, která je kořenově uniformní, pak platí $\mathrm{avg}(n) \approx 2\ln(n)$ a $\overline{\mathrm{avg}}(n) \approx 2\ln(n)$.

Randomizovaný strom

Zdroj náhodnosti u stromu sestaveného z náhodných klíčů je ve vzorkování permutace. Přesuneme jej dovnitř algoritmu pro přidávání/mazání prvku do/z BVS.

```
random-insert
⟵ r: Node
⟵ x: Node
⟶ Node
```

```
1  if (r == nil) then return x
2  if (random(0, r.count) == 0) then return root-insert(r,x)
3  if (x.id < r.id)
4     set-left-child(r, random-insert(r.left, x))
5  else
6     set-right-child(r, random-insert(r.right, x))
7  return r
```

Procedura `random(a,b)` vrací náhodné číslo z množiny $\{a, a+1, \ldots, b\}$ (s uniformní pravděpodobností).

Mazání uzlu `x`

- Pokud má `x` 2 potomky, s pravděpodobností

$$\frac{\texttt{x.left.count}}{\texttt{x.count-1}}$$

  jen nahradíme pořádkovým předchůdcem. Jinak je nahradíme pořádkovým následníkem
- Zbytek algoritmu je nezměněn.

Věta

Pokud se na fixní posloupnost randomizovaných operací `insert` a `delete` díváme jako na vzorkování, je toto vzorkování kořenově uniformní.

Skip-list

Analogie seznamu (pro teď uvažujme jednosměrné seznamy)

Každý uzel může být zapojen do 1 až t seznamů a obsahuje tedy odpovídající počet pointerů na další uzel (ty jsou organizovány např. do pole). Číslo t chápeme jako konstantu, která parametrizuje skip-list (podobně jako u B stromů).

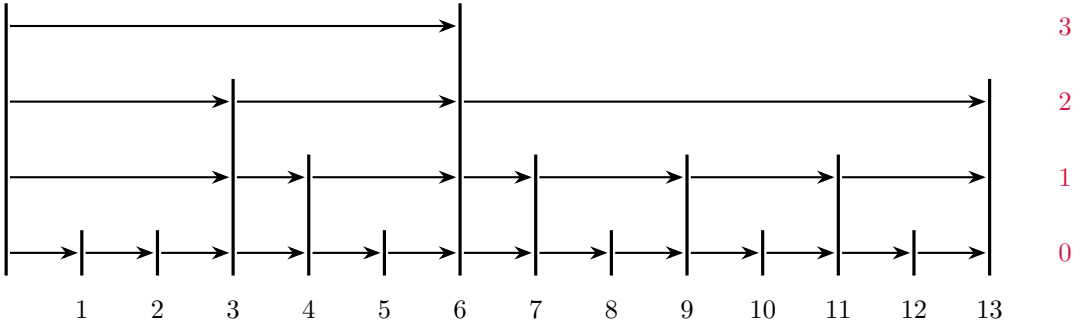
V každém seznamu jsou uzly uspořádány vzestupně podle klíčů

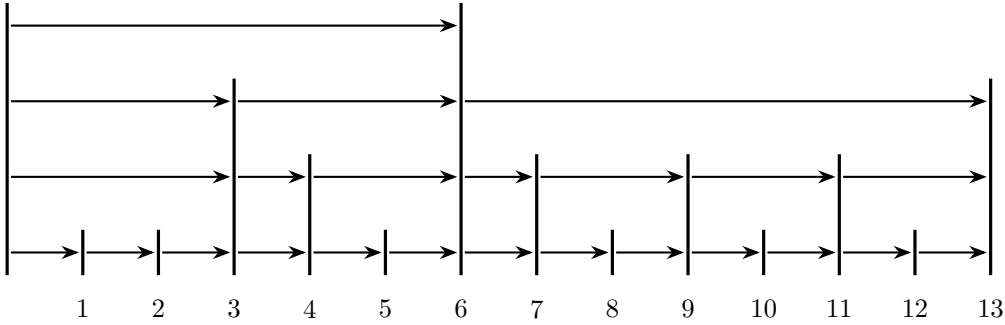
I samotné seznamy jsou uspořádány do hierarchie

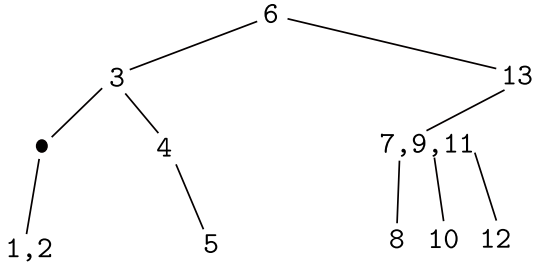
- seznam na úrovni 0 obsahuje všechny uzly
- seznam na úrovni $i > 0$ je podseznamem seznamu na úrovni $i - 1$.

Na začátku seznamů máme speciální prvek — sentinel.

Příklad




3
2
1
0
1
2
3
4
5
6
7
8
9
10
11
12
13

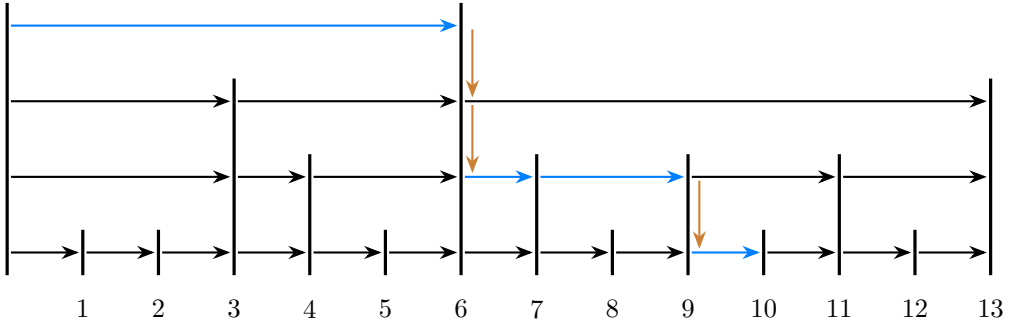

6
3
13
4
7,9,11
1,2
5
8
10
12

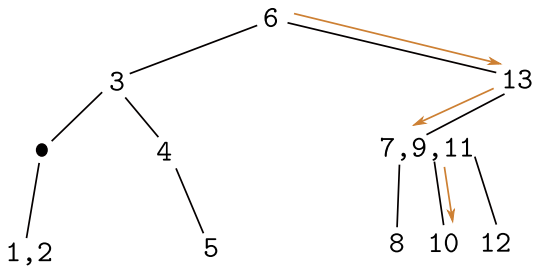
Vyhledávání

Idea hledání klíče k

- Prohledáváme nejvyšší seznam dokud nenalezneme uzel s jehož soused je `nil`, nebo neobsahuje klíč větší než k.
- Posuneme se na seznam, který je o úroveň níže a pokračujeme obdobně.
- Při návštěvách uzlů kontrolujeme shodu jejich klíče s k.
- Vyhledávání končí neúspěchem, pokud klíč nenajdeme na levelu 0.

Složitost je dána součtem délek úseků seznamů, z jednotlivých levelů, které projdeme.


3
2
1
0
1
2
3
4
5
6
7
8
9
10
11
12
13


6
3
13
4
7,9,11
1,2
5
8
10
12

Vkládání

Vkládáme uzel x

- Označme s délku seznamu na urovni i, který se nachází mezi dvěma prvky seznamu na úrovni $i+1$.
- s chápeme jako jeden z parametrů skip-listu, podobně jako t. Musíme si ovšem uvědomit, že s a t nejsou nezávislé na počtu prvků, které chceme do skip listu vložit.
- Pro vkládaný uzel vybereme maximální úroveň, do které bude zapojen. Uděláme to s použitím náhody. Předpokládáme, že máme proceduru

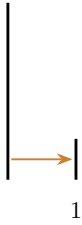
$$\texttt{rand(s,t)} = \begin{cases} j & \text{s pravděpodobností } 1/s^j, \text{ pokud } t > j > 0 \\ 0 & \text{jinak, tj. s pravděpodobností } 1 - \sum_{k=1}^{t-1} 1/s^k \end{cases}$$

  Nastavíme `x.sz` $\leftarrow$ `rand(s,t)+1`
- Provádíme proceduru `search` s modifikacemi: jsme-li na úrovni $l <$ `x.sz` a ve vyhledávání klesáme o úroveň níže, nebo jsme na úrovni 0 a došlo k neúspěšnému vyhledávání, zapojíme x do seznamu na úrovni l za aktuálně testovaný uzel.

Příklad, $t = 3$, $s = 2$

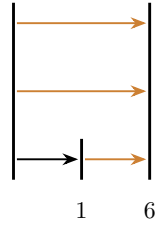
```
insert(1)

rand -> 0
```



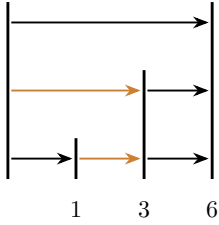

```
insert(6)

rand -> 2
```



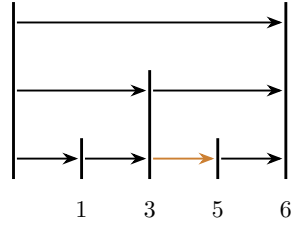

```
insert(3)

rand -> 1
```




```
insert(5)

rand -> 0
```



Složitost

Věta

Vyhledávání a vkládání do skip-listu s parametrem s vyžaduje průměrně $O(s \log_s n)/2$ porovnání.

Ale nezapomenout na vztah mezi s, n, t !