Strom, průchody stromem, binární vyhledávací stromy

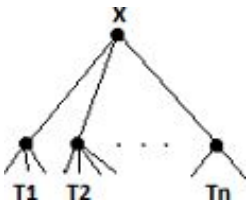
Z FAV wiki

Strom je ADT - matematicky jde o typ grafu, který neobsahuje cykly.

Definice

- **Vrchol (uzel)** - jeden prvek stromu. Má jednoho předchůdce a 0 - N následovníků.
- **Hrana (větev)** - spojení dvou vrcholů
- **List** - nrchol bez následovníků
- **Kořen** - vrchol, který nemá žádné předchůdce. Z kořene je pak možné se dostat do jakéhokoliv uzlu stromu. Každý strom má jen jeden kořen.
- **Podstrom** - // dopsat
- **Cesta stromem** - cesta od libovolného vrcholu ke kořeni stromu. Tato cesta je ve stromu unikátní.
- **Hloubka vrcholu** - délka cesty od kořene do daného vrcholu.
- **Výška stromu** - nejdelší možná cesta ve stromu.

// přidat obrázek s popisem částí stromu



Implementace: Objektově - každá prvek má ukazatele Parent a např. pole ukazatelů Childern, výodné, jednoduché Polem - složité pro obecné stromy, je nutné ukládat indexy kořenů podstromů, počty podstromů těchto kořenů a data vrcholů jako taková. Speciálně lze implementovat stromy s konstatntním počtem podstromů v každém vrcholu (BVS, Octree, ...), kde je implementace jednodušší

Adresářová struktura, sportovní výsledky, reprezentace stromů v přírodě :), ...

Průchod stromem

Do šířky - Projdeme vrcholy po patrech, tedy kořen, postupně kořeny jeho podstromů, kořeny všech těchto podstromů atd. Jelikož ve stromech obecně neznáme sousedy, musíme se často vracet. Měřením cesty vrcholů však můžeme prohledávání do šířky převést na prohledávání do hloubky, i když ne zrovna elegantní (vyhledáme všechny vrcholy o hloubkách 0, pak 1, pak 2, ...)

Do hloubky - Procházíme vrcholy do hloubky, tedy dojdeme-li do vrcholu, projdeme jeho podstromy a poté se teprve vracíme

Procházení stromu může být průchod do hloubky dvojího druhu, podle priority operací:

- **Preorder** - Nejprve vyzvedneme data vrcholu a poté projdeme postupně všechny podstromy

```
preorder(node) {
  print node.value
  if node.left ≠ null then preorder(node.left)
  if node.right ≠ null then preorder(node.right)
}
```

- **Postorder** - Nejprve projdeme všechny podstromy, a poté vyzvedneme data vrcholu

```
postorder(node) {
  if node.left ≠ null then postorder(node.left)
  if node.right ≠ null then postorder(node.right)
  print node.value
}
```

Procházení binárního stromu má navíc ještě jeden způsob:

- **Inorder** - Projdeme levý podstrom, poté vyzvedneme data vrcholu, a poté pravý podstrom. Při výpisu BVS touto metodou dojdeme k seřazenému poli.

Binární vyhledávací strom (BVS)

BVS je binární strom (strom, který má maximálně 2 podstromy v kažném vrholu) vytvořený tak, že kořen levého podstromu každého vrcholu má nižší hodnotu než tento vrchol a ten má nižší hodnotu než kořen pravého podstromu.

Pri přidáváni vrcholu začínáme u kořene, a rozhodujeme se, kterou hranou jít dále, tedy je li vkládaný prvek menší , nebo větší než tento kořen. Takto postupujeme i vybraným podstromem až najdeme místo, kam vrchol umístit. Umístěný vrchol je vždy listem stromu.

Mazání je složitější. Nejprve vrchol najdeme ve stromu a podle toho jaký je:

- Pokud je listem, je to triviální
- Pokud vrchol má pouze jeden podstrom, vrchol smažeme a na jeho místo umístíme kořen jeho (jediného) podstromu
- Pokud má dva, vydáme se doprava a poté jdeme stále doleva dokud nenarazíme na list, který smažeme, ale jeho hodnotu přiřadíme původně mazanému vrcholu.

Tento strom je nevyvážený, tedy může nastat situace, kdy bude jeden podstrom kořenu mít výrazně větší výšku než podstrom druhý. První vložený prvek bude vždy kořen stromu. Při načítání prvků do stromu je tedy vhodné alespoň pro tento prvek nalézt medián.

Přidání prvku do pole a mazání je maximálně O(n) (pro strom, který byl vytvořen ze seřazené posloupnosti - je to tedy pouze lineární spojový seznam), očekávaná složitost operací je však O(log2 n) jelikož je strom v každém vrcholu rozdělen na 2 skupiny hodnot.