

Datové struktury – Binární strom

Michal Blažek

Fakulta informačních technologií Vysokého učení technického v Brně
Božetěchova 1/2. 612 66 Brno - Královo Pole
xblaze38@fit.vutbr.cz

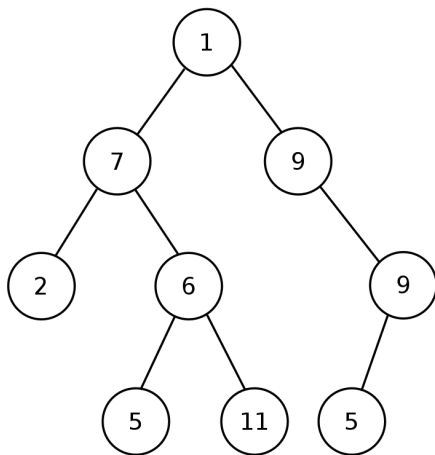


22. dubna 2024

- Binární strom je orientovaný graf s jedním vrcholem a z něj existuje cesta do všech ostatních vrcholů grafu.
- Binární strom se skládá z **uzlů**, které jsou propojeny pomocí **hran**.
- Každý uzel může mít maximálně 2 další potomky a 1 rodičovský uzel.
- Uzel, který nemá žádného rodiče, se nazývá **kořen stromu**.
- Kořenový uzel existuje v každém binárním stromě a to přesně jeden.

Rekurzivní definice binárního stromu viz (1):

Binární strom je buď prázdný, nebo sestává z jednoho uzlu zvaného kořen a dvou binárních podstromů – levého a pravého. (Oba podstromy mají vlastnosti binárního stromu.)



Obrázek: Příklad binárního stromu s kořenem, který má hodnotu 1

- Každý uzel má přiřazen klíč a podle něj jsou uzly uspořádány.
- **Levý** podstrom obsahuje vždy pouze uzly s klíčem, který je **menší** než je klíč tohoto uzlu.
- **Pravý** podstrom potom obsahuje pouze uzly s klíčem, který je **větší** než je klíč tohoto uzlu.

Poznámka:

Čísla na obrázku 1 nezobrazují klíče jednotlivých uzlů, ale pouze ilustrační hodnoty.

Výhody

- + Ve vyváženém binárním stromě je přístup k uzlu rychlý.
- + Lze využít i k řazení prvků.
- + Využití při tvorbě asociativní paměti.

Nevýhody

- Strom může při nešťastném vkládání a odstraňování prvků degradovat na seznam, pokud se nevyvažuje.
- Vrcholy naukládají informaci o rodiči, takže je potřeba si ukládat cestu k danému uzlu.

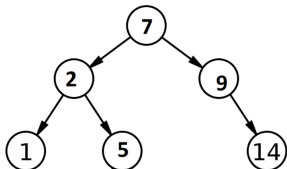
Základní operace:

- Průchod stromem: $O(N)$
- Smazání stromu: $O(N)$
- Přidání uzlu: $O(\log(N))$
- Odstranění uzlu: $O(\log(N))$
- Nalezení uzlu: $O(\log(N))$

Existují 3 průchody binárním stromem:

- **Preorder** - nejprve zpracovává kořen, pak levý podstrom a nakonec pravý podstrom.
- **Inorder** - nejprve zpracovává levý podstrom, pak kořen a nakonec pravý podstrom.
- **Postorder** - nejprve zpracovává levý podstrom, pak pravý podstrom a až nakonec kořen.

Toto zpracovávání se provádí rekurzivně, přičemž se vždy začíná v kořenu binárního stromu.



- Například průchod stromem na obrázku pomocí **preorderu** by vypadal následovně: 7,2,1,5,9,14.
- **Inorder** by nám vrátil seřazené prvky: 1,2,5,7,9,14.
- A **postorder** by vypadal následovně: 1,5,2,14,9,7

void printPreorder(N* *node*):

```
if node is not null then  
    write(node->data)  
    printPreorder(node->left)  
    printPreorder(node->right)  
end if
```

void printInorder(N* *node*):

```
if node is not null then  
    printInorder(node->left)  
    write(node->data)  
    printInorder(node->right)  
end if
```

void printPostorder(N* *node*):

```
if node is not null then  
    printPostorder(node->left)  
    printPostorder(node->right)  
    write(node->data)  
end if
```

Tyto algoritmy ukazují možnou rekurzivní implementaci jednotlivých funkcí pro průchod stromem.

Pro maximální efektivitu binárního stromu je zapotřebí strom vyvážit po každém vložení nebo odstranění uzlu.

Existují 2 druhy vyvážení (lze nalézt v (1)):

- Výšková vyvážení

Binární strom je výškově vyvážený, když pro každý jeho uzel platí, že výška levého podstromu se rovná výšce pravého podstromu a nebo se liší právě o 1.

- Váhová vyvážení

Binární strom je váhově vyvážený, když pro každý jeho uzel platí, že počty uzlů jeho levého a pravého podstromu se rovnají a nebo se liší právě o 1.

Datová struktura	Průměrná složitost	Maximální složitost	Seřazené prvky
Binární strom	$O(\log(N))$	$O(\log(N))$	Ano
Nevyvážený binární strom	$O(\log(N))$	$O(N)$	Ano
Vázaný seznam	$O(N)$	$O(N)$	Ne
Pole (nutná znalost indexu)	$O(1)$	$O(1)$	Ne
Hashovací tabulka	$O(1)$	$O(N)$	Ne

Tabulka: Porovnání složitostí při hledání prvku a dalších vlastností některých datových struktur



Honzík, J. M.; Honzík, J. M.: Vybrané kapitoly z programovacích technik. Brno: VUT v Brně, vyd. 3. vydání, 1991, ISBN 80-214-0345-4.

Děkuji za pozornost!