Řešení samostatných úloh ze šestého cvičení předmětu ALG

Matouš Vrba

9. listopadu 2021

Příklad 4)

Hloubka vyváženého BVS s u uzly je $h = \log_2(n+1) - 1$. Pokud má vyvážený BVS $2^n - 1$ uzlů, je jeho hloubka $n - 1$. Příklad lze nyní rozdělit na dvě podúlohy:

Doba vyhledávání, pokud klíč neexistuje

Pokud vyhledávaný klíč ve stromu není, je potřeba prohledat celou hloubku stromu. Projití jednoho uzle trvá 1 µs a hloubka stromu je $n - 1$ uzlů, takže pokud klíč neexistuje, bude doba vyhledávání $n - 1$ mikrosekund.

Průměrná doba vyhledávání, pokud klíč existuje

Pokud je hledaný klíč v kořeni (který má hloubku 0), bude doba vyhledávání 1 µs. Pokud je hledaný klíč v první vrstvě (hloubka 1, obsahuje 2 klíče), bude doba vyhledávání 2 µs. Podobně pro další hloubky (viz tabulka).

hloubka	**počet klíčů 1**	**doba vyhledávání**
0	1	1
1	2	2
2	4	3
3	8	4
4	16	5
...	...	...
$n-1$	2^{n-1}	n

Průměrná doba vyhledávání v mikrosekundách je tedy

$$\frac{\sum_{i=0}^{n-1} 2^i(i+1)}{2^n - 1}.$$

Celková průměrná doba vyhledávání

Pokud 50% vyhledávaných klíčů není v BVS přítomných, je potom celková průměrná doba vyhledávání (v mikrosekundách)

$$\frac{1}{2}\left(\frac{\sum_{i=0}^{n-1} 2^i(i+1)}{2^n-1}+n-1\right)=\frac{1}{2}\left(n+\frac{n2^n-2^n+1}{2^n-1}-1\right).$$

Příklad 5)

Předpokládejme, že nemáme k dispozici seznam všech uzlů (pak by jejich součet šel zjistit snadno s lineární složitostí), takže pro prohledání všech uzlů musíme procházet stromem.

Lze použít například prohledávání do šířky, jehož časová složitost je $\Theta(n+v)$, kde n je počet uzlů a v počet hran mezi uzly. V případě stromu je počet hran mezi uzly vždy $v = n-1$. Výsledná složitost je tudíž $\Theta(n+v) = \Theta(n+n-1) = \Theta(n)$.

Příklad 6)

Obdobně jako v předchozím příkladu můžeme použít například prohledávání do šířky. Výsledná složitost je tudíž zas $\Theta(n)$.

Příklad 9)

Například

```
struct node
{
  node* left;
  node* right;
};

node* TreeMinimum(node* root)
{
  if (root->left != nullptr)
    return TreeMinimum(root->left);
  else
    return root;
}
```

Příklad 14)

Úlohu lze řešit podobně jako v příkladu na cvičení. Algoritmus prochází stromem v pořadí inorder a když narazí na první prvek, který je větší, než x, začne počítat uzly až do doby, než narazí na první prvek, který je větší, než y. Obecně je asymptotická složitost algoritmu $O(n)$, protože v nejhorším případě musíme projít všechny uzly ve stromu.

Příklad 15)

Algoritmus postupně odstraňuje celý pravý podstrom. Složitost operace R tedy bude záviset na hloubce tohoto podstromu, což je

1. $\Theta(\log(n))$ pro vyvážený strom,
2. $O(n)$ pro obecný strom strom.