Operace Find, Insert, Delete v BVS

1.
Složitost operace Insert ve vyváženém BVS s n uzly je vždy

1. $O(\log(n))$
2. O(log(hloubka BVS))
3. $\Theta(n)$
4. $O(1)$
5. $\Theta(\log(n))$
6. $\Theta(\log(\text{hloubka BVS}))$

2.
Složitost operace Delete v BVS s n uzly je vždy

1. $O(\log(n))$
2. O(log(hloubka BVS))
3. $\Theta(n)$
4. $O(n)$
5. $\Theta(\log(n))$
6. $\Theta(\log(\text{hloubka BVS}))$

2a.
Podobné otázky pro případ operací Find, Insert, Delete ve vyvážených nebo nevyvážených stromech jen opakují fakta z přednášky, resp. okamžitý jednoduchý náhled na celou věc.

3.
Binární vyhledávací strom:

a) musí splňovat podmínku haldy *(na tohle příjdeme později)*
b) udržuje v každém uzlu referenci na uzel s nejbližším větším klíčem
c) udržuje v každém uzlu referenci na uzel s nejbližším větším i s nejbližším menším klíčem
d) po každém vložení prvku do BVS musí proběhnout vyvážení stromu
e) po průchodu v pořadí inorder vydá seřazenou posloupnost klíčů

Podmínku haldy musí splňovat pouze halda, která ani není BVS. Reference na jiné uzly než jen bezprostřední potomky nejsou v BVS povinností a vyvažování BVS také není povinné. Zbývá tak jen možnost e), o níž se hovoří i při výkladu pořadí inorder jako takového.

4.
Čísla ze zadané posloupnosti postupně vkládejte do prázdného binárního vyhledávacího stromu (BVS), který nevyvažujte. Jak bude vypadat takto vytvořený BVS?
Poté postupně odstraňte první tři prvky. Jak bude vypadat výsledný BVS?

Posloupnost a)
14 24 5 13 1 3 22 10 19 11
Posloupnost b)
10 16 5 17 4 15 3 1 23 13 2 11
Posloupnost c)
17 4 15 2 5 9 1 12 3 19 16 18

5.
Mějme klíče 1, 2, 3, ..., n. Číslo n je liché. Nejprve vložíme do BVS všechny sudé klíče v rostoucím pořadí a pak všechny liché klíče také v rostoucím pořadí. Jaká bude hloubka výsledného stromu? Změnil by se nějak tvar stromu, kdybychom lichá čísla vkládali v náhodném pořadí?

Vyvažování a rotace v BVS, všude se předpokládá AVL strom

6.
Jednoduchá levá rotace v uzlu u má operační složitost

a) závislou na výšce levého podstromu uzlu u
b) mezi $O(1)$ a $\Theta(n)$
c) závislou na hloubce uzlu u

d) konstantní

7a.
LR rotace v uzlu u lze rozložit na
a) levou rotaci v pravém synovi uzlu u následovanou pravou rotací v uzlu u
b) pravou rotaci v pravém synovi uzlu u následovanou levou rotací v uzlu u
c) levou rotaci v levém synovi uzlu u následovanou pravou rotací v uzlu u
d) pravou rotaci v levém synovi uzlu u následovanou levou rotací v uzlu u

7b.
RL rotace v uzlu u lze rozložit na
a) levou rotaci v pravém synovi uzlu u následovanou pravou rotací v uzlu u
b) pravou rotaci v pravém synovi uzlu u následovanou levou rotací v uzlu u
c) levou rotaci v levém synovi uzlu u následovanou pravou rotací v uzlu u
d) pravou rotaci v levém synovi uzlu u následovanou levou rotací v uzlu u

8.
Kolik jednoduchých rotací se maximálně provede při jedné operaci Insert v AVL stromu s n uzly?
a) jedna
b) dvě
c) log(n)
d) n

9.
LR rotace v uzlu u má operační složitost
a) závislou na hloubce uzlu u
b) Θ (log n)
c) konstantní
d) lineární

10a.
Levá rotace v uzlu u
a) v podstromu s kořenem u přemístí pravého syna u_p uzlu u do kořene. Přitom se uzel u stane levým synem uzlu u_p a levý podstrom uzlu u_p se stane pravým podstromem uzlu u
b) v podstromu s kořenem u přemístí levého syna u_l uzlu u do kořene. Přitom se uzel u stane pravým synem uzlu u_l a levý podstrom uzlu u_l se stane pravým podstromem uzlu u
c) v podstromu s kořenem u přemístí pravého syna u_p uzlu u do kořene. Přitom se uzel u stane levým synem uzlu u_p a pravý podstrom uzlu u_p se stane levým podstromem uzlu u
d) v podstromu s kořenem u přemístí levého syna u_l uzlu u do kořene. Přitom se uzel u stane pravým synem uzlu u_l a pravý podstrom uzlu u_l se stane levým podstromem uzlu u

10b.
Pravá rotace v uzlu u
a) v podstromu s kořenem u přemístí pravého syna u_p uzlu u do kořene. Přitom se uzel u stane levým synem uzlu u_p a levý podstrom uzlu u_p se stane pravým podstromem uzlu u
b) v podstromu s kořenem u přemístí levého syna u_l uzlu u do kořene. Přitom se uzel u stane pravým synem uzlu u_l a levý podstrom uzlu u_l se stane pravým podstromem uzlu u
c) v podstromu s kořenem u přemístí pravého syna u_p uzlu u do kořene. Přitom se uzel u stane levým synem uzlu u_p a pravý podstrom uzlu u_p se stane levým podstromem uzlu u
d) v podstromu s kořenem u přemístí levého syna u_l uzlu u do kořene. Přitom se uzel u stane pravým synem uzlu u_l a pravý podstrom uzlu u_l se stane levým podstromem uzlu u

11.
Vyvážení BVS některou z operací rotace
a) je nutno provést po každé operaci Insert
b) je nutno provést po každé operaci Delete
c) je nutno provést po každé operaci Insert i Delete
d) snižuje hloubku stromu
e) není pro udržování BVS nezbytné

Krátké implementační úlohy v BVS, u vyvažování/rotací se předpokládá AVL strom

12.
Uzel binárního binárního vyhledávacího stromu obsahuje tři složky: Klíč a ukazatele na pravého a levého potomka.
Navrhněte rekurzívní funkci (vracející `bool`), která porovná, zda má dvojice stromů stejnou strukturu. Dva BVS považujeme za strukturně stejné, pokud se dají nakreslit tak, že po položení na sebe pozorovateli splývají, bez ohledu na to, jaká obsahují data.

13.
Upravte sami (velmi snadno) předchozí funkci tak, aby zjistila, zda jsou dva BVS shodné, t.j. zda se shodují strukturou i svými daty.

14.
Napište rekurzivní verze operací: TreeMinimum, která vrátí referenci na uzel s nejmenší hodnotou v BVS.

15.
Napište proceduru *strom leváRotace(strom s).* Napřed si namalujte obrázek a rozmyslete si, kterých ukazatelů se změna dotkne.

16.
Je dána struktura popisující uzel binárního vyhledávacího stromu takto

```
struct node
{
      valType val;
      node * left, right;
      int count;
}
```

Nebo v Javě takto

```
class Node {
   valType val;    // na tom, co je valType, v tomto případě nesejde
   Node left;
   Node right;
   int count;
}
```

Navrhněte nerekurzívní proceduru, která do složky count v každém uzlu zapíše počet vnitřních uzlů v podstromu, jehož kořenem je tento uzel (včetně tohoto uzlu, pokud sám není listem).

17.
Při výpisu hodnot uzlů BVS v pořadí inorder získáme uspořádanou posloupnost hodnot. Napište nerekurzivní funkci, která v každém uzlu daného BVS zamění levý a pravý podstrom. (Po této úpravě bude strom „zrcadlovým obrazem“ původního a výpisem v pořadí inorder bychom získali opačně uspořádanou posloupnost.)

18.
Napište funkci, jejímž vstupem bude ukazatel (=reference) na uzel X v BVS a výstupem ukazatel (=reference) na uzel s nejbližší vyšší hodnotou ve stromu.

19.
Navrhněte algoritmus, který spojí dva BVS A a B. Spojení proběhne tak, že všechny uzly z B budou přesunuty do A, přičemž se nebudou vytvářet žádné nové uzly ani se nebudou žádné uzly mazat. Přesun proběhne jen manipulací s ukazateli. Předpokládejte, že v každém uzlu v A i v B je k dispozici ukazatel na rodičovský uzel.