1. Popište algoritmus, který ze setříděného pole vytvoří dokonale vyvážený BVS v lineárním čase.

Vezmeme prostřední prvek pole jako kořen a rekurzivně sestavíme levý a pravý podstrom z prvků od něj nalevo/napravo.

2. Popište, jak v BVS najít nejbližší větší hodnotu y pro zadané x.

```
v <- Find(x)
pokud v má pravý podstrom:
    v <- pravý syn v
    dokud v má levý podstrom:
        v <- levý syn v
    return v
pokud v nemá pravý podstrom:
    pokud v je kořen: return v
    v' <- otec v
    pokud v je levý syn v': return v'
    pokud v je pravý syn v': v <- v'
```

3. Ukažte, jak z jednoduchých rotací na AVL stromě složit dvojitou.

Pro situaci z Průvodce stačí rotovat hranu xy a pak vzniklou hranu yz.

4. Ukažte, že v AVL stromě nikdy opakované mazání a přidávání téhož prvku nezpůsobí víc jak konstantní počet rotací celkem.

Pro každý vrchol v na cestě od kořene do místa přidávaného prvku:
Pokud má v znaménko 0, po operaci zjevně nebude potřebovat vyvážit.
Pokud má v znaménko $-$ a prvek patří do jeho levého podstromu, může dojít při prvním vložení k vyvážení v, čímž získá znaménko 0. Po následném mazání si udrží znaménko 0 nebo získá znaménko + (viz další případ). Obdobně pro znaménko + a pravý podstrom.
Pokud má v znaménko $-$ a prvek patří do jeho pravého podstromu, tak si buď udrží znaménko $-$ nebo získá znaménko 0 (netřeba vyvažovat). Obdobně pro znaménko + a levý podstrom.

5. Navrhněte algoritmus, který v (modifikovaném) AVL stromě pro $a, b \in X$ najde počet vrcholů v intervalu $[a, b]$ v čase $\mathcal{O}(\log n)$.

AVL strom pozměníme tak, aby si v každém vrcholu ukládal počet prvků v podstromě. Najdeme cestu od kořene k vrcholu s hodnotou a a b. Od vrcholu r, ve kterém se tyto cesty rozchází, projdeme do a a posčítáme počty vrcholů v pravých podstromech. Přidáme počet prvků na cestě, které do intervalu také náleží. Tím získáme počet prvků v intervalu $[a, r)$. Obdobně získáme počet prvků v intervalu $(r, b]$. Součtem pak získáme celkový počet vrcholů.

6. Uvažme volnější podmínku hloubkového vyvážení stromu, která dovoluje rozdíl hloubek až H. Jak by vypadala varianta AVL stromu, která toto využívá, a jakou by měly časovou složitost její operace?

Úpravou argumentu o logaritmické hloubce AVL stromu (počítáním minimálního počtu vrcholů pro danou hloubku, viz Průvodce) dostaneme rekurenci $A_1 = 1, \ldots, A_H = H, A_h = 1 + A_{h-1} + A_{h-H-1}$. Ta nám dává stále exponenciální růst minimálního počtu vrcholů a tedy i logaritmickou hloubku, akorát s jiným základem.

7. Mějme AVL strom s řetězci délky nanejvýš L řazenými lexikograficky. Jaká je asymptotická složitost jeho operací? Porovnejte jej s trií (písmenkovým stromem).

V AVL stromě strávíme v každém vrcholu $\mathcal{O}(L)$ čas porovnáváním, tedy operace budou trvat $\mathcal{O}(L \cdot \log n)$. Trie přitom zvládá operace v $\mathcal{O}(L)$. Zajímavé začíná porovnání být pokud budeme místo asymptotické složitosti řešit počet přístupů do paměti.

