1. *Perfektní vyváženost: Navrhněte algoritmus, který ze setříděného pole vyrobí v lineárním čase dokonale vyvážený BVS. (Počet vrcholů v levém podstromu se liší od počtu vrcholů v pravém podstromu maximálně o 1)*

   Nechť P je vstupní setříděné pole. Použijem rekurzivní funkci Postav(a, b), která postaví dokonale vyvážený BVS z prvků $P[a], P[a+1], \ldots, P[b]$. Pokud je $b-a$ menší než 2 tak přímo postavíme strom. Jinak vybereme prostřední prvek, tj. na indexu $s = \lfloor \frac{a+b}{2} \rfloor$, jako kořen a jeho levý podstrom, resp. pravý podstrom, sestrojíme pomocí Postav(a, $s-1$), resp. Postav($s+1$, b). Na začátku stačí zavolat Postav(0, $n-1$).

2. *BVS-join: Navrhněte algoritmus, který dostane dva BVS T_1, T_2 a sloučí jejich obsah do jediného BVS. Algoritmus by měl pracovat v čase $O(|T_1| + |T_2|)$.*

   Nejpřímočařejší řešení je pomocí inorder průchodu vypsat oba stromy do setříděných polí P_1 a P_2, následně je slít do jediného setříděného pole P. A nakonec použít předchozí příklad k postavení stromu. Vše stihneme v lineárním čase.

3. *BVS-split: Navrhněte operaci* BvsSplit*, která dostane BVS T a hodnotu s, a rozdělí strom na dva BVS T_1 a T_2 takové, že hodnoty v T_1 jsou menší než s a hodnoty v T_2 jsou větší než s.*

   Jedna možnost po vzoru předchozího příkladu by nejjednodušší řešení bylo vypsat strom v lineárním čase do pole, rozdělit na prvky menší než s a větší než s, a následně z obou půlek vytvořit dokonale vyvážený strom. Vše opět v lineárním čase.

   Pokud bychom nevyžadovali aby výsledné stromy byly dokonale vyvážené, postačilo by vložit do T hodnotu s a pak s vyrotovat do kořene. Tím dostaneme T_1 a T_2 v čase lineárním s hloubkou T.

   Otázka na rozmyšlení - co kdybychom místo toho pracovali s AVL stromy. Šlo by udělat split v logaritmickém čase? Jako první krok se hodí vymyslet jak rychle umíte sloučit dva AVL stromy T_1 a T_2 takové, že všechny hodnoty v T_1 jsou menší než hodnoty T_2 (v závislosti na hloubce T_1 a T_2).

4. *Vylepšování BVS: Uvažujme obecný BVS uchovávající pouze hodnoty (bez klíčů) s operacemi Insert, Delete a Find v čase $O(\log n)$. Se zachováním asymptotické složitosti všech operací (včetně rotací v $O(1)$) naučte BVS následující operace v co nejlepší časové složitosti:*

   - *Počet prvků, Min a Max celého stromu.*
   - *k-tý prvek stromu.*
   - *Pro prvek předaný odkazem odpovědět kolik menších prvků ve stromě existuje.*
   - *Kolik prvků ve stromě je z intervalu $[a, b]$.*

   .

   - Počet prvků si můžeme jednoduše uchovat mimo strom, aktualizovat při každému operaci a odpověď na dotaz zvládneme v konstantním čase. Minimum (maximum je symetrické) zvládneme obdobně - udržujeme si globální minimum. Při Insert operaci ho případně aktualizujeme, při Deletu potenciálně musíme najít nové minimum. Pokud Delete odstranil minimální prvek jednoduše najdeme nový minimální v čase O(hloubka stromu), čímž nezhoršíme složitost Delete.

- Pro k-tý nejmenší prvek si budeme u každého vrcholu udržovat velikost jeho podstromu $|T(v)|$. Potom hledáme jednoduše rekurzivně - v každém vrcholu umíme v konstantním čase zjistit kolik prvků leží v jeho levém podstromě (a tedy je menších) a podle toho pokračujeme rekurzivně doleva, doprava nebo jsme právě nalezli k-tý prvek. Rozmyslete si, že udržování velikostí podstromů zvládneme bez zhoršení časových složitostí ostatních operací. Přidáním nebo smazáním prvku vždy změníme velikosti jen u vrcholů na jedné cestě - a jednoduše umíme informaci upravit. Složitost tedy O(hloubka stromu).
- Pro prvek x předaný odkazem zjistíme jeho pořadí s využitím informací o velikostech podstromů. Začneme v x a víme že všechny prvky v levém podstromu jsou menší. Postupně se vynořujeme od x ke kořenu a pokaždé když do vrcholu y přijdeme zprava, započítáme prvky v levém podstromu a navíc ještě y. Tím spočítáme všechny prvky menší než x a tedy i pořadí x. Složitost tedy O(hloubka stromu).
- Díky předchozímu odstavci umíme zjistit počet prvků menších než a (rozmyslete si detaily pokud a není ve stromě). Symetricky umíme spočítat počet prvků větších než b. Pak už jen tyto hodnoty odečteme od celkového počtu prvků. Opět složitost O(hloubka stromu). Rozmyslete si jak by fungoval „přímočarý“ přístup - tedy počítat přímo prvky v intervalu, bez odečítání od celkového počtu. To budeme potřebovat v následujícím příkladu.

5. *Vylepšování BVS 2: Uvažme obecný BVS uchovávající dvojice (klíč, hodnota) setříděné podle klíče. Se zachováním asymptotické složitosti všech operací naučte BVS následující operace v co nejlepší časové složitosti:*

   - *Min a Max hodnot v rámci celého stromu.*
   - *Min, Max a průměr v daném intervalu klíčů* $[a, b]$.
   - *Přičtení* δ *ke všem hodnotám v intervalu klíčů* $[a, b]$.

   - Ukážeme jak vyřešit Min, Max je symetrické. U každého vrcholu si budu uchovávat minimální hodnotu v jeho podstromu. Díky tomu umíme zodpovědět dotaz v $O(1)$ - vypíšeme minimum v podstromu kořene. Uchovávání funguje podobně jako velikosti podstromů v předchozím případě - u Insert a Delete potřebuji aktualizovat jen jednu cestu do kořene, což stihnu v čase O(hloubka stromu). Detaily si rozmyslete.
   - Pro jednoduchost předpokládejme, že klíče a a b jsou ve stromě a jsou listy (rozmyslete si co se změní v opačném případě). Nechť c je nejhlubší vrchol, který leží na cestě z kořene do a i na cestě z kořene do b. Potom kdykoliv na cestě z c do a zahneme ve vrcholu d doleva, celý pravý podstrom d i d samotné musí ležet v intervalu $[a, b]$. Stejně tak kdykoliv na cestě z c do b zahneme ve vrcholu d doprava, musí celý levý podstrom d i d samotné ležet v intervalu $[a, b]$. Tímto jsme interval $[a, b]$ rozložili na malou množinu disjunktních podstromů a jelikož pro každý z nich máme uložené minimum, dokážeme najít minimum přes celý interval. Složitost opět O(hloubka stromu).
   - Tohle si ještě schováme na příště :)