1. Ukažte, proč jsou nutné podmínky na parametry a, b u (a, b)-stromu – vytvořte příklady, kde některá z operací nepůjde provést nebo bude pomalá.

   Mít a menší než 2 by znamenalo povolit vrcholy bez klíče nebo s pouze jedním synem, což by znamenalo, že strom přestane mít všechny listy na nejnižší úrovni. Navíc by mohly vznikat dlouhé cesty.
   Malé b by rozbilo slučování/štěpení vrcholů (v nově sloučeném bude moc klíčů/ve vrcholech zbylých po rozštěpení nebude dost klíčů).

2. Popište asymptotickou složitost operací na (a, b)-stromě v závislosti na počtu uložených prvků n a parametrech a, b.

   Viz https://mj.ucw.cz/vyuka/dsnotes/03-abtree.pdf:
   - `Find`: $\Theta(\log b \cdot \log_a n) = \Theta\left(\log n \cdot \frac{\log b}{\log a}\right)$
   - `Insert`: $\Theta(b \cdot \log_a n) = \Theta\left(b \cdot \frac{\log n}{\log a}\right)$
   - `Delete`: $\Theta(b \cdot \log_a n) = \Theta\left(b \cdot \frac{\log n}{\log a}\right)$

3. Najděte optimální parametry pro (a, b)-strom 64bitových čísel ukládaný na disk s 4096bytovými bloky (tedy každé čtení/zápis načte/zapíše 4096 bytů, chceme minimalizovat jejich počet).

   Budeme uvažovat 8bytové pointery, tedy chceme maximální velikost vrcholu ≤ 4096. $b = 256$ dává ≤ 256 synů, tedy ≤ 255 klíčů, což se vejde do bloku ($256 \cdot 8 + 255 \cdot 8 = 4088 \leq 4096$). Pak dává smysl $a = 128$ (z minulého cvičení, $(a, 2a)$-stromy jsou optimální).

4. Upravte `Insert` na $(2, 4)$-stromě tak, aby mu stačil pouze jeden průchod stromem od kořene dolů.

   Můžeme strom udržovat jako $(2, 3)$-strom (tedy 1 nebo 2 klíče na vrchol) s tím, že přetečení na 3 klíče nevyřešíme ihned při `Insert`u zpětným průchodem ke kořeni, ale až při dalším průchodu dolů přes vrchol.
   Toto nám garantuje, že při štěpení vrcholu můžeme jeho otci vždy přidat další klíč (měl nanejvýš dva, bude mít nanejvýš tři).

5. Mějme softwarový systém, který používá hashovací tabulku s přihrádkami s fixní hashovací funkcí h pro řetězce bytů pevné délky l definovanou polynomem ($h(x) = \sum_{i=1}^{l} a^i x_i \bmod m$). Představte si, že jste útočník, který chce systém zahltit. Najděte pro to vhodnou posloupnost prvků. Jaká bude asymptotická složitost `Insertu` každého prvku v takovém případě, jaká pak následného `Findu`?

Chceme najít co nejvíce prvků, které se zahashují do téže přihrádky. Pro přihrádku p je hledání takových prvků x stejné, jako hledání řešení lineární rovnice $\sum_{i=1}^{l} a^i x_i \equiv p (\mathrm{mod}\, m)$.
Můžeme použít nějaký obecný algoritmus na řešení takových rovnic, nebo třeba pokud existují b, i t.ž. $a^i b \equiv p$, dosadit za x_i násobky b.
Pro k prvků, co se zahashují do stejné přihrádky, bude každý `Insert` stát $\mathcal{O}(1)$ (přidání do spojového seznamu) a každý `Find` stát $\mathcal{O}(k)$ (musíme projít spojový seznam).

6. Rozšiřte nějaký vyhledávací strom přiřazující klíčům čísla tak, aby uměl operaci `Add`(x, a, b), která ke každému číslu s klíčem v intervalu $[a, b]$ přičte hodnotu x, v čase $O(\log n)$. Zachovejte složitost existujících operací.

Vezměme třeba AVL strom. Do každého vrcholu v přidáme hodnotu Δ_v, která bude „virtuálně“ přičtena k celému podstromu. `Add` pak spočívá v nalezení všech podstromů obsahujících hodnoty z intervalu $[a, b]$ a přičtení x k jejich Δ_v. `Find` při průchodu posčítá všechny Δ z předchůdců hledaného vrcholu a přičte tento součet k jeho hodnotě. `Insert` tento součet s spočítá taky a uloží hodnotu o s menší.

