Čítač s `DEC`. Uvažme n-bitový čítač, který podporuje operace `INC` a `DEC` (zvýšení a snížení o 1) a `TESTZERO` (zjisti, zda je číslo nulové). Modifikujeme ho tak, že každý bit může nabývat hodnot 0, +1 a -1. Ukažte, že v této reprezentaci je amortizovaná složitost obou operací $\mathcal{O}(1)$.

Kolize. Dostali jste hešovací funkci $h : [U] \to [m]$. Pokud o této funkci nic dalšího nevíte, kolik vyhodnocení funkce potřebujete, abyste našli k-tici prvků, které se všechny zobrazí do téže přihrádky?

`DELETE` vs `PSEUDODELETE`. U hešování s otevřenou adresací a lineárním prohledáváním jsme řekli, že mazání je složitější a usnadníme si práci tím, že místo mazání prvek označíme jako smazaný, a až bude takto označených prvků hodně (lineárně v n), tak tabulku přebudujeme. Vymyslete, jak se obejít bez této strategie, která funguje dobře jen amortizovaně, a místo toho navrhněte `DELETE`, který poběží ve stejném čase jako `INSERT` či `FIND` (tedy udělá průměrně nanejvýš cca. $\frac{1}{1-\alpha}$ operací).
Proč to nebude fungovat v jiných variantách, než je lineární prohledávání?

Narozeninový paradox. Kolik lidí musí být na party, aby pravděpodobnost, že dva lidé mají narozeniny ve stejný den, byla aspoň 1/2? (Odpověď je překvapivě nízké číslo a říká nám, že první kolize v hashovací tabulce nastane překvapivě brzy.)

Rick a zombies. K Rickovi se ze všech stran blíží zombies. Předpokládejme, že Rick má neomezenou zásobu nábojů, střílí dokonale přesně a na jeden zásobník umí zastřelit 6 zombíků. Vyměnit zásobník mu ale trvá 1s a během té se zombies posunou o 1m k němu. Přežije to? Spočítejte to co nejrychleji. Vstup je reprezentován jako (nesetříděná) posloupnost $d_1, \dots, d_n$, kde d_i představuje vzdálenost i-tého zombíka. Čísla d_i mohou být dlouhá, tzn. není dobrý nápad vytvořit pole délky $\max_i d_i$.

Bloomův filtr. Bloomův filtr je datová struktura pro přibližnou reprezentaci množiny. Skládá se z pole bitů $B[1, \dots, m]$ a hešovací funkce h, která prvkům univerza přiřazuje indexy v poli. `INSERT(`x`)` nastaví $B[h(x)] = 1$. `MEMBER(`x`)` otestuje, zda $B[h(x)] = 1$. Vložme nyní do filtru nějakou n-prvkovou množinu M. Pokud $x \in M$, `MEMBER(`x`)` vždy odpoví správně. Pokud se ale zeptáme na $x \notin M$, může se stát, že $h(x) = h(y)$ pro nějaké $y \in M$, a `MEMBER(`x`)` odpoví špatně. Spočítejte, s jakou pravděpodobností se to pro dané m a n stane.
Hint: využijte toho, že $1 + \alpha \le e^\alpha$, $\forall \alpha \in \mathbb{R}$.

Vylepšení Blooma. Spolehlivost Bloomova filtru můžeme zvýšit tak, že si pořídíme k filtrů s různými hešovacími funkcemi. `INSERT` bude vkládat do všech, `Member` se zeptá všech a odpoví ano pouze tehdy, když se na tom všechny filtry shodnou. Je-li pravděpodobnost chyby jednoho filtru p, pak kombinace k filtrů chybuje s pravděpodobností pouhých p^k. Vymyslete, jak nastavit m a k pro případ, kdy chceme ukládat 10^6 prvků s pravděpodobností chyby nejvýše 10^{-9}. Minimalizujte spotřebu paměti.

Mergesort na víc částí? Popište třídicí algoritmus, který bude vstup rozkládat na více než dvě části a ty pak rekurzivně třídit. Může být rychlejší než náš Mergesort?

Tranzitivní uzávěr. *Tranzitivní uzávěr* orientovaného grafu s vrcholy $\{1, \dots, n\}$ je nula-jedničková matice T tvaru $n \times n$, kde $T_{uv} = 1$ právě tehdy, když v grafu existuje cesta z vrcholu u do vrcholu v. Ukažte, že umíme-li násobit matice $n \times n$ v čase (n^ω), můžeme vypočítat tranzitivní uzávěr v čase $(n^\omega \log n)$.