8. CVIČENÍ Z DATOVÝCH STRUKTUR 1, ZS25/26

Ještě jednou hešování: lineární přidávání, počítací Bloomův filtr a využití rolling hash

1. *Lineární přidávání s většími skoky.* Uvažujme hešování s otevřenou adresací řízené obecnou lineární posloupností $h(x,i) = (f(x) + c \cdot i) \mod m$, kde c je konstanta nesoudělná s velikostí tabulky m. Srovnejte jeho chování s obyčejným lineárním přidáváním.

2. *Opravdové mazání pro lineární přidávání.* Na přednášce bylo řešení kolizí pomocí lineárního přidávání s tím, že pokud prvek mažeme, pouze ho označíme za smazaný („postavíme tam pomníček“). Zkuste domyslet detaily a alternativní řešení:

a) Pokud provedeme hodně operací mazání, bude hešovací tabulka obsahovat více označených (tj. smazaných) prvků než nesmazaných. Co provést v takovém případě? Jak zajistit konstantní amortizovanou časovou složitost (ve střední hodnotě) pro libovolnou sekvenci operací Insert a Delete? (Můžete předpokládat větu z přednášky o konstantní střední hodnotě počtu přihrádek do nejbližší volné.)

b) Alternativní způsob: mazaný prvek skutečně smažeme a poté vhodně přesuneme nějaké prvky. Vymyslete, jak to přesně udělat, abychom pak mohli vyhledat všechny nesmazané prvky (tedy abychom tabulku „nerozbili“).

3. *Počítací Bloomův filtr.* Uvažme Bloom filtr, který umožňuje mazání tím, že v každém políčku místo jednoho bitu máme b-bitové počítadlo pro malé b, např. $b = 4$. Toto počítadlo ale nesmí přetéct, takže když se dostane na hodnotu $2^b - 1$, *zamrzne* a už se nemění. Zanalyzujte pravděpodobnost, že jedno konkrétní počítadlo v jedné tabulce zamrzne po vložení n prvků. Můžete předpokládat, že velikost tabulky je $m \geq \ln 2 \cdot n$.

Poté odhadněte pravděpodobnost, že nějaké počítadlo zamrzne.

Nyní hurá na řetězcové algoritmy:

4. *Rabin-Karpův algoritmus na vyhledávání v textu.* Využijte „rolling hash“ $\mathcal{R}$ z přednášky k vytvoření jednoduchého algoritmu na nalezení všech výskytů daného řetězce J délky m („jehly“) v textu S délky n („seně“) a to v průměrném čase $O(n+m+k \cdot m)$, kde k je počet výskytů J v S. Jak velké potřebujeme prvočíslo p pro $\mathcal{R}$?

5. *Bonus: Perfektní hešování dle Fredmana, Komlóse a Szemerédiho (FKS).* Máme dánu n-prvkovou množinu S jako podmnožinu nějakého (obrovského) univerza $\mathcal{U}$, např. 64-bitové integery. Cílem je navrhnout pro S datovou strukturu velikosti $O(n)$, která zvládne pro zadaný dotaz x zjistit, jestli x náleží v S, v konstantním čase vždy. (V čem klasická hešovací tabulka velikosti $O(n)$ nesplňuje požadavky?)

a) Připomeňte si narozeninový paradox, tedy nejmenší počet lidí s rovnoměrně náhodnými narozeninami takový, aby s pravděpodobností alespoň 50% měli dva stejné narozeniny.
b) Hešování s úplně náhodnou (nebo c-univerzální) hešovací funkcí funguje podobně. Jak zhruba musíme mít velkou tabulku, aby nastala kolize s pravděpodobností méně jak 50%?
c) Perfektní hešování vybudujeme takto: Pořídíme si hešovací funkci $h : \mathcal{U} \to [m]$ pro $m = O(n)$, kterou vybereme náhodně z c-univerzální rodiny. Pro každou přihrádku $i \in [m]$ zavedeme tabulku druhé úrovně, která bude dost velká, aby tam nenastala kolize s pravděpodobností alespoň 50% (pokud by kolize nastala, změníme hešovací funkci).
d) Teď už stačí jen omezit celkovou velikost tabulek druhé úrovně ve střední hodnotě.