PÁTEČNÍ CVIČENÍ Z DATOVÝCH STRUKTUR 1

Hešování: lineární přidávání a ještě univerzalita/nezávislost

Včetně stručných řešení:

1. *Lineární přidávání s většími skoky.* Uvažujme hešování 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.

Řešení. Pro libovolné c nesoudělné s m se chová stejně jako běžné lineární přidávání se skokem 1 — také se tvoří srůstající řetězce, jen pro permutaci indexů danou přičítáním c

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. 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?

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“).

Řešení.

a) Pokud tabulka obsahuje (např.) polovinu prvků, které jsou již označené za smazané, tak vše přehešujeme. To má pořád amortizovanou konstantní složitost (v průměru a za předpokladu, že srůstající řetízky jsou v průměru konstantně dlouhé; průměr zde znamená střední hodnotu přes náhodnost hešovací funkce, o které stačí předpokládat, že je c-univerzální / 2-nezávislá).

b) Po smazání prvku nám může vzniknout mezera v řetízku a pak bychom nemohli vyhledat prvky, které jsou v řetízku za mezerou. Náprava je jednoduchá.

3. *Lineární přidávání a multiply-shift.* Na přednášce 6.12. byl na slajdu 11 zajímavý graf: Pokud přidáváme aritmetickou posloupnost $1, 2, 3, \ldots, 0.9 \cdot m$ do tabulky s lineárním přidáváním, tak se hešování multiply-shift chová lépe než náhodná hešovací funkce. Umíte vysvětlit, proč tomu tak je?

Řešení. Hešování multiply-shift pravidelně rozmístí prvky (kolize nastanou, ale jsou rozmístěné rovnoměrně), kdežto při náhodném hešování to funguje jako "balls and bins"(náhodné házení míčků do koše) — intuitivně bude relativně velká část přihrádek prázdných a pak se budou vyskytovat dlouhé řetízky, zhruba logaritmické velikosti (viz délka nejdelšího řetízku pro separované řetězce na přednášce). Tyto dlouhé řetízky podstatně zvednou počet kroků.

Ještě pár zbylých příkladů k důkazům c-univerzality a k-nezávislosti (pokud si to chcete procvičit):

4. *Lineární kongruence.* Ukažte, že systém funkcí $\mathcal{H}_{\text{lin}} = \{(ax+b \mod p) \mod m \mid 0 < a < p, 0 \leq b < p\}$ je 1-univerzální, kde p je libovolné prvočíslo a $m \leq p$. (Rozdíl oproti 9. přednášce: na přednášce mohlo být $a = 0$, kdežto tady ho nepovolíme, a pak se pomocí Lemma o zobecněném modulu m dokázala jen 2-univerzalita, resp. $(2, 2)$-nezávislost.)

Řešení. Viz Průvodce labyrintem algoritmů (sekce 11.5, podsekce konstrukce z lineární kongruence)

5. *škálování modulo prvočíslo.* Ukažte univerzalitu systému funkcí $\mathcal{H}_{\text{skal}} = \{(ax \mod p) \mod m\}$. Musíme vyloučit $a = 0$?

Řešení. Pro dva různé prvky $x \neq y$ z univerza je pravděpodobnost $ax = ay \mod p$ (přes volbu a) jen $1/p$ (nastane jen pro $a = 0$). Pokud toto nenastane, pak se postupuje podobně jako v důkazu v předchozím cvičení.