11. cvičení z ADS1

13. 5. 2021

Příklad 1.

Uvažujme hešování s otevřenou adresací. Na adresaci použijeme funkci $h(x, i) = (f(x) + c \cdot i) \bmod m$, kde c je konstanta nesoudělná s m. Jak se toto hešování chová ve srovnání s lineárním přidáváním?

Příklad 2.

Jak byste pro hešování s otevřenou adresací naimplementovali operaci Delete, pokud chcete, aby po nalezení políčka, ze kterého se má mazat, už operace trvala amortizovaně $\mathcal{O}(1)$?

Příklad 3.

Nechť $h_{a,b}(x) = ((ax + b) \bmod p) \bmod m)$. Potom systém hešovacích funkcí $\mathcal{L} = \{h_{a,b} \mid a, b \in [p], a \neq 0\}$ je 1-univerzální.

Ukažte, že pokud v systému $\mathcal{L}$ zafixujeme parametr b na nulu, bude 2-univerzální. Podobně ukažte, že pokud dovolíme nulové a, bude systém taky 2-univerzální.

Definice.

Bloomův filtr je datová struktura pro přibližnou reprezentaci množiny. Skládá se z pole bitů $B[1, \ldots, m]$ a hešovací funkce h, která prvkům univerza přiřazuje indexy v poli. Operace Insert(x) nastaví $B[h(x)]$ na 1; Member(x) otestuje, zda $B[h(x)] = 1$.

Příklad 4.

Vložme nyní do Bloomova 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 dostaneme špatnou odpověď. Spočítejte, s jakou pravděpodobností se to pro dané m a n stane.

Příklad 5.

Vícecestný Mergesort pro k cest dělí posloupnost na k stejně velkých částí, které rekurzivně seřadí, a pak je slije. Nejprve ukažte, jak slévat k sežazených posloupností celkové délky n v čase $\mathcal{O}(n \log k)$. Pak zanalyzujte časovou složitost vícecestného Mergesortu.

Čemu odpovídá n-cestný Mergesort?

Příklad 6.

Mějme rekurzivní algoritmus, který umi násobit matice velikosti $2^{n+1} \times 2^{n+1}$. Ten dělá to, že si matici rozdělí na *bloky* velikosti $2^n \times 2^n$:

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}, B = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}, C = AB = \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix}.$$

Zřejmě $C_{ij} = A_{i1} \cdot B_{1j} + A_{i2} \cdot B_{2j}$.

Tyto dvě matice tedy odpovídají nějakým maticím velikosti 2×2 skládajících se z bloků. Tak prostě algoritmus vynásobí tyto dvě matice běžným způsobem, ale součin bloků spočítá rekurzivně (součet je jednoduchý).

Rozmyslete si, že takto definované násobení matic bude fungovat a analyzujte jeho časovou složitost.