5. cvičení

Datové struktury I, 30. 10. 2024 https://iuuk.mff.cuni.cz/~vesely/vyuka/

Úloha 1 (Správná volba parametrů)
Z přednášky víme, že libovolná posloupnost m operací Insert a Delete na $(a, 2a)$-strom celkem změní jenom $\mathcal{O}(m)$ vrcholů (když začínáme s prázdným stromem). Ukažte, že toto neplatí pro $(a, 2a-1)$-stromy, tedy pro libovolné m, n navrhněte posloupnost m operací na stromě s $\Theta(n)$ vrcholy, která celkem změní $\Omega(m \log n)$ vrcholů.
Můžete začít s (2,3)-stromy, a potom zobecnit pro libovolné a. Zároveň můžete začít s libovolným (validním) n-vrcholovým stromem, a až na konci ukázat, že jej opravdu lze vyrobit z prázdného stromu.

Úloha 2 ((a, b)-Join)
Navrhněte operaci Join pro (a, b)-stromy: máte tedy dva stromy T_1, T_2 s tím, že všechny klíče v T_1 jsou menší než v T_2, a cílem operace je spojit dva stromy do jednoho. Pozor na to, že stromy mohou být různě vysoké. (Pro analýzu SplITu se může hodit analyzovat složitost přesněji než jen $\mathcal{O}(\log n)$.)

Úloha 3 (Třídíme bez rekurze)
Analyzujte I/O složitost mergesortu, kde máme vždy v jednom poli za sebou setřízené posloupnosti a po dvojicích je sléváme do posloupností dvojnásobné délky ve druhém poli.
(Základem je tedy cyklus, ne rekurze.)
Můžete předpokládat $M \geq 3B$ – speciálně máme alespoň tři bloky.

Úloha 4 (Programování II flashbacks)
Spočítejte I/O (a časovou) složitost k-cestného mergesortu. Ten narozdíl od předchozího algoritmu slévá vždy k posloupností současně a pro výběr minima používá k-prvkovou haldu. Předpokládejme $M \geq 2k \cdot B$, aby se nám do paměti vešla halda i potřebné části rozečtených posloupností.

Úloha 5 (Násobíme matice)
Spočtěte I/O složitost následujících algoritmů pro výpočet součinu matic $C = A \cdot B$, kde A, B jsou zadané čtvercové matice $n \times n$.

a) Předpokládejme, že A je uložená po řádcích a B je uložená po sloupcích. Prozkoumejte přímočarý algoritmus dle definice.

b) Dále se podívejme na rekurzivní násobení matic. (To naivní[1], kde matice rozdělíme na čtvrtiny a násobíme spolu tyto čtvrtiny, které pak k sobě přičítáme.) Můžete předpokládat vysokou cache: $M \in \Omega(B^2)$.

Bonusové úlohy

Úloha 6 (QuickSelect)
Proveďte analýzu počtu přenesených bloků následující implementace QuickSelectu pro nalezení mediánu:

1. Pokud $N < \mathcal{O}(B)$, hledáme medián naivně.
2. Vybereme uniformně náhodně pivota. Tuto volbu opakujeme, dokud neplatí, že vybraný pivot je v prostředních dvou kvartilech prvků v současném poli.
3. Pole přeskládáme tak, že začne prvky menšími než pivot, pak bude pivot, a nakonec máme prvky větší než pivot.
4. Pokud je pivot na indexu $N/2$, vrátíme jej jako medián.
5. Pokud je pivot na indexu menším než $N/2$, rekurzivně jej hledáme v prvcích větších než pivot, jinak jej hledáme v prvcích menších než pivot. Zároveň si vhodně upravíme, kolikátý prvek hledáme.

Můžou se vám hodit následující nápovědy:

- Uvědomte si, že počet přenesených bloků je nyní Náhodná Veličina, bude nás tedy zajímat jeho střední hodnota.

[1] Ale pokud se vám chce počítat Strassena, bránit vám v tom nehodlám.

- Linearita střední hodnoty: pro Náhodné Veličiny X, Y a $\alpha, \beta \in \mathbb{R} : \mathbb{E}[\alpha X + \beta Y] = \alpha\mathbb{E}[X] \cdot \beta\mathbb{E}[Y]$
- Pro $X \sim \text{Geom}(p)$ máme $\mathbb{E}[X] = 1/p$

Úloha 7 ((a, b)-Split)
Navrhněte operaci Split: máte tedy (a, b)-strom T a klíč k, a chcete T rozdělit na dva stromy tak, že je v jednom je vše menší než k a ve druhém je zbytek.

Úloha 8 (I/O-optimální třídění)
Dá se ukázat, že nejde třídit s lepší I/O složitostí než $\mathcal{O}(n/B \cdot \log_{M/B}(n/B))$ (jsou-li prvky blackboxy, které umíme jen porovnávat a přesouvat vcelku). Navrhněte *cache-aware* algoritmus s touto složitostí. Jako základ použijte mergesort, který upravte tak, že bude opravdu využívat veškerou cache, kterou má k dipozici.

Úloha 9 (Hledáme medián)
Uvažme následující (Blumův) algoritmus pro počítání mediánu:

1. Představme si, že pole rozdělíme na $\lceil n/5 \rceil$ pětic.
2. V každé pětici spočteme medián.
3. Rekurzivně spočteme medián mediánů M.
4. Rozdělíme prvky do dvou množin, podle toho, jestli jsou větší nebo menší než M.
5. Podle velikosti těchto množin se zarekurzíme do množiny, která obsahuje více prvků.

Připomeňte si, že pro normální RAM model bez hierarchie pamětí je rekurence pro složitost následovná: $T(n) = 7n/5 + T(n/5) + (n-1) + T(7n/10) \in \mathcal{O}(n)$. Určete I/O složitost tohoto algoritmu, můžete předpokládat, že $M \geq 3B$.
Také se může hodit fakt, že řešení rovnice $\left(\frac{1}{5}\right)^c + \left(\frac{7}{10}\right)^c = 1$ je $c \approx 0.83978$.