6. cvičení

Datové struktury I, 6. 11. 2023 https://iuuk.mff.cuni.cz/~chmel/2324/ds1/

Úloha 1 (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.

Řešení
Máme tři bloky, začínáme délkou 1, a pokračujeme délkou 2, takže máme $\log_2(n)$ iterací, a v každé iteraci přečteme všechny bloky (některé potenciálně dvakrát), takže máme celkem $\mathcal{O}((\frac{n}{B}+1)\log n)$.
U rekurze můžeme získat až $\mathcal{O}(\frac{n}{B}\log(\frac{n}{M}))$.

Úloha 2 (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.

Řešení
Časová složitost: $\mathcal{O}(n\log(n))$.
I/O složitost: vlastně skoro stejně jako předtím, ale logaritmus bude se základem k.

Úloha 3 (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 $M \geq 3B$.

Řešení
Rekurence pro přístupy na disk: $T(n) = T(n/5) + T(7n/10) + \mathcal{O}(1+n/B)$, protože v prvním kroku nic neděláme, ve druhém kroku stačí dva bloky pro iterování přes hlavní pole a pole pro zapisování mediánů. Ve třetím kroku rekurzivně voláme pro $n/5$. Ve čtvrtém kroku použijeme tři bloky pro iterování přes hlavní pole, pole menších prvků a pole větších prvků, což je $\mathcal{O}(1+n/B)$, a v posledním kroku voláme pro nejvýše $7n/10$.
Podíváme se na strom rekurze - ten má N^c listů pro c řešení $\left(\frac{1}{5}\right)^c + \left(\frac{7}{10}\right)^c = 1$, která vznikla z rekurence pro počet listů $L(n) = L(n/5) + L(7n/10), L(1) = 1$ dosazením $L(n) = n^c$. Zpozorujeme, že $T(\mathcal{O}(B)) \in \mathcal{O}(1)$, a tím pádem máme jenom $(n/B)^c$ listů stromu rekurze, které stojí $\mathcal{O}((n/B)^c) \in o(n/B)$ přístupů na disk, a tím si můžeme všimnout, že složitost úrovní se snižuje od kořene geometrickou řadou, a tedy celková složitost je cena v kořeni. (Alternativně, můžeme tipnout, že to vyjde lineárně a indukcí ověřit řešení.)

Úloha 4 (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.

Řešení
Víme M, B – použijeme $(\lfloor M/B \rfloor - \mathcal{O}(1))$-cestný merge sort.