5. CVIČENÍ Z DATOVÝCH STRUKTUR 1, ZS24/25

Kešujeme

1. *I/O rekurzivního MergeSort.* Zanalyzujte I/O složitost rekurzivního MergeSortu a porovnejte s nerekurzivním na přednášce. (Bonus: zanalyzujte k-cestný rekurzivní MergeSort.)

2. *TransposeAndSwap naivně.* Připomeňte si cache-oblivious transpozici matic. Poté ukažte, že pokud TransposeAndSwap upravíme tak, že nejprve transponujeme obě zadané matice (rekurzivně) a teprve poté je prohazujeme (průchodem po řádcích), tak dostaneme horší časovou složitost i počet přenosů.

3. *Násobení matic.* Chceme spočíst matici $\mathbf{C} = \mathbf{A} \cdot \mathbf{B}$, kde $\mathbf{A}$ a $\mathbf{B}$ jsou zadané čtvercové matice $n \times n$. Pro následující přístupy určete I/O složitost (můžete předpokládat, že $n > M$):

a) Nejprve předpokládejte, že $\mathbf{A}$ je uložena po řádcích a $\mathbf{B}$ po sloupcích, a použijte přímočarý algoritmus pro násobení dle definice.

b) Nyní každou z matic rozdělíme na 4 bloky, které vynásobíme rekurzivně. (Pro jednoduchost předpokládejme, že n je mocnina dvojky.) Pro analýzu předpokládejte, že cache je „štíhlá“, tedy $M \geq c \cdot B^2$ pro nějakou konstantu c, tj. *tall cache assumption*. (Těžší bonus: analyzujte Strassenův algoritmus :-)

4. *Analýza algoritmu dle starověkého „Rozděl a panuj!“* Proveďte analýzu počtu přenesených bloků, tedy I/O složitost, pro algoritmus QuickSelect pro nalezení k-tého nejmenšího prvku s náhodným výběrem pivota. Jelikož je pravděpodobnostní, určete střední hodnotu.

(Pro jednoduchost můžete nejprve odstranit náhodnost a předpokládat, že v každém kroku vybereme pivota, který je *pseudomediánem*, tedy leží v prostředních dvou kvartilech.)

5. *Hledáme medián deterministicky.* 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 $c \approx 0.83978$.

6. *Binární vyhledávání optimálně, byť cache-aware.* Binární vyhledávání v uspořádaném poli má I/O složitost $\Theta(\log N - \log B + 1)$. Navrhněte způsob uložení prvků do pole tak, abychom mohli vyhledávat s I/O složitostí $O(\log_B N + 1) = O(\log N / \log B + 1)$ (tato složitost je dokonce optimální). Předpokládejte, že znáte B. Hint: sestrojte nejprve dokonale vyvážený BVS.