6. cvičení

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.

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.

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 $M \geq 3B$.

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.