7. CVIČENÍ Z DATOVÝCH STRUKTUR 1, ZS24/25

Cache-oblivious transpozice, kompetitivnost LRU, trocha pravděpodobnosti a konečně to hešování!!!

1. *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.

2. *Kompetitivnost LRU v problému online správy cache.* Určete nejhorší možný poměr počtu výpadků (cache misses) algoritmu LRU (Least Recently Used) oproti optimálnímu algoritmu, mají-li tyto algoritmy k dispozici stejně velkou velikost cache. Srovnejte se Sleator-Tarjanovou větou.

3. *Lemma o džbánu.* Mějme událost, která se stane v jednom kroku s pravděpodobností $p > 0$ (nezávisle na ostatních krocích). Ukažte, že

$$\mathbb{E}[\#\text{ kroků než nastane událost}] = \frac{1}{p}.$$

4. *Pojďme rozbít hešování!* Dostali jste hešovací funkci $h : \mathcal{U} \to [m]$. Pokud o této funkci nic dalšího nevíte, kolik nejvýše vyhodnocení funkce potřebujete, abyste našli k-tici prvků, které se všechny zobrazí do téže přihrádky?

5. *Pravděpodobnost kolize* Mějme množinu S velikosti n a plně náhodnou hešovací funkci $h : U \to [m]$, kde $m = n^2$. Odhadněte pravděpodobnost, že nastane v rámci S kolize, tedy dva prvky se zahešují do stejné přihrádky.

6. *A ještě pravděpodobnost: pevné body permutací.* Mějme uniformně náhodnou permutaci na n prvcích. Určete střední hodnotu počtu pevných bodů této permutace. (Hint: uvažte indikátory události $\pi(i) = i$, tedy že i je pevný bod náhodné permutace π.)

7. *Bonus: Binární vyhledávání optimálně.* Binární vyhledávání v uspořádaném poli má I/O složitost $\Theta(\log N - \log B + 1)$. Navrhněte cache-aware 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). Hint: sestrojte nejprve dokonale vyvážený BVS.

8. *Bonus: Optimální správa cache, která není online.* Ukažte optimalitu „offline“ algoritmu Longest Forward Distance (LFD), který vyhodí z cache blok, jež bude potřeba nejpozději v budoucnu. (Hint: spor.)