1. **Jak dobrý je LRU?** Jaký je worst-case poměr[1] mezi počtem výpadků algoritmu LRU oproti optimálnímu algoritmu, mají-li tyto algoritmy k dispozici stejně velkou cache velikosti k?

   Nápověda. Vymyslete co nejhorší posloupnost stránek pro LRU, na které si optimální algoritmus poradí dobře. Příklad optimálního algoritmu najdete v Úloze 3.

2. **Jak *neřešit* domácí úkol.** Uvažme následující variantu rekurzivního algoritmu pro transpozici matice.

   „Nejprve rekurzivně transponujeme podmatice a pak je teprve prohazujeme."

   Jakou má tento algoritmus časovou složitost a jaký je jeho počet výpadků?

3. **Jak může vypadat nějaký optimální algoritmus?** Ukažte, že následující offline[2] algoritmus pro cachování, známý jako *Longest Forward Distance (LFD)* je optimální.

   „Pokud je potřeba uvolnit z cache nějakou stránku, pak se uvolní ta, která bude potřeba nejpozději v budoucnu."

4. **Online algoritmy.** Byli jsme uvězněni do číselné osy v bodě 0. Východ je v bodě $x \in \mathbb{R} \setminus \{0\}$. Problém je, že nevíme, kde východ je a dokážeme jej poznat jen tím, že na něj stoupneme. (Jednou z komplikací je například, že nevíme, jestli je x kladné či záporné. Kdybychom to věděli, tak jdeme rovně tím správným směrem, dokud netrefíme cíl. Tohle je mimochodem optimální algoritmus.)

   Vymyslete algoritmus, který najde cíl a nachodíme při něm co nejméně v poměru k $|x|$.

5. **Online algoritmy podruhé.** Jsme na neomezeně dlouhé dovolené v horách a chceme lyžovat. Každý den si můžeme buď lyže pronajmout na ten den, nebo si je koupit a pak je používat navždy. Pronájem lyží stojí \$1 na den, nákup lyží stojí \$C a smíme je používat následující dny bez omezení.

   Máme však předtuchu, že se jednoho dne objeví Medvěd a zláme nám nohy, čímž ukončí naši lyžařskou kariéru. Problémem však je, že nevíme, kdy ten den nastane. Jaká je optimální strategie, abychom mohli co nejvíce lyžovat za co nejméně peněz?

   Příklad 1. Nechť $C = 100$ a víme, že Medvěd přijde druhý den. Pak by za pronájem první den bychom zaplatili \$1 (což je optimální), zatímco koupě první den by nás stála \$100. Tím pádem by strategie nákupu měla kompetitivní poměr 100.

   Příklad 2. Nechť $C = 100$ a víme, že Medvěd přijde miliontý den. Pak bychom za koupi první den zaplatili \$100 (což je optimální).

6. Zvládnete v předchozích dvou úlohách dokázat, že váš algoritmus je optimální?

7. Pomůžeme si v Úloze 4, pokud máme k dispozici dokonalý generátor náhodných čísel?

   Poznámka. Pak bude nachozená vzdálenost náhodná veličina blablabla.

8. Pokud vás bavilo přemýšlení nad online algoritmy, zapište si předmět Aproximační a Online Algoritmy (`NDMI018`). Žel budou až ten další rok.

9. Proveďte cache-oblivious analýzu následující implementace `QuickSortu`.

   1. Pokud $N < \mathcal{O}(B)$, naivně vrátíme seřazené pole.
   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 následuje pivot a nakonec máme prvky větší než pivot.
   4. Rekurzivně necháme seřadit pole před pivotem a pole za pivotem.

[1] Tomuhle se říká *competitive ratio*, česky asi *kompetitivní poměr*.
[2] Tedy vidí budoucnost a zná celou posloupnost dotazů dopředu.