1. *Nejdelší rostoucí posloupnost:* NRP nemusí být jednoznačně určena. Jak spočítat, kolik různých NRP obsahuje zadaná posloupnost?

2. *Kopec:* Kopcem nazveme podposloupnost, která nejprve roste a pak klesá. Vymyslete algoritmus, který v zadané posloupnosti nalezne nejdelší kopec.

3. *Knihovna:* Mějme posloupnost n knih. Každá kniha má nějakou šířku s_i a výšku v_i. Knihy chceme naskládat do knihovny s nějakým počtem polic tak, abychom dodrželi abecední pořadí. Prvních několik knih tedy půjde na první polici, další část na druhou polici, a tak dále. Máme zadanou šířku knihovny S a chceme rozmístit police tak, aby se do nich vešly všechny knihy a celkově byla knihovna co nejnižší. Tloušťku polic a horní a spodní desky přitom zanedbáváme.

4. *Edit v malém prostoru:* Algoritmus na výpočet editační vzdálenosti z Průvodce zabere $\Theta(nm)$ buněk paměti na uložení tabulky. Ukažte, jak spotřebu paměti snížit na $\Theta(n + m)$.

5. *Edit explicitně:* Kromě editační vzdálenosti můžeme chtít spočítat i příslušnou nejkratší posloupnost editačních operací. V grafové interpretaci našeho algoritmu je to triviální – prostě vypíšeme nalezenou nejkratší cestu. Ukažte, jak to udělat bez explicitního sestrojení grafu, třeba přímou úpravou algoritmu z Průvodce. Nemusíte se snažit zachovat paměťovou složitost z předchozího příkladu.

6. *Edit rychleji:* Na první pohled se zdá, že čím podobnější řetězce dostaneme tím by mělo být jednodušší zjistit jejich editační vzdálenost. Vymyslete jak zrychlit náš algoritmus aby běžel v čase $O((n+m)(L(x,y)+1))$.

7. *X-podposloupnost: X-posloupností* nazveme posloupnost kterou umíme rozdělit na dvě disjunktní posloupnosti tak, že jedna je rostoucí a druhá klesající - např 1,3,**6**,5,**2**,7,**0** je X-posloupnost (klesající podposloupnost je zvýrazněna tučně). Vymyslete algoritmus, který v zadané posloupnosti nalezne nejdelší X-podposloupnost.

8. *Vrstevnatá podposloupnost: Vrstevnatou posloupností* nazveme konkatenaci klesajících posloupností $P_1, P_2, \ldots, P_k$ takových že pro $i > j$ je každý prvek P_i větší než každý prvek P_j. Například 2,1 – 6,4,3 – 9,7 – 10 – 11 – 13,12 je vrstevnatou posloupností ("–"značí předěly jednotlivých klesajících posloupností). Vymyslete algoritmus, který v zadané posloupnosti nalezne nejdelší vrstevnatou podposloupnost.

9. *Knihovna 2.0:* Podobně jako ve cvičení Knihovna chceme navrhnout knihovnu, jež pojme dané knihy. Tentokrát ovšem máme zadanou maximální výšku knihovny a chceme najít minimální možnou šířku. Pokud vám to pomůže, předpokládejte, že všechny knihy mají jednotkovou šířku.

10. *Edit explicitně a v malé paměti:* I v cvičení Edit explicitně si lze vystačit s pamětí $\Theta(n+m)$. Existuje pěkný trik založený na metodě Rozděl a panuj. Budeme hledat editační vzdálenosti pomocí hledání nejkratší cestu v grafu. Graf postačí procházet po řádcích (to je topologické pořadí), ale obvyklé grafové algoritmy by si pro vypisování cesty musely zapamatovat předchůdce každého vrcholu. My si místo toho uložíme jen to, ve kterém sloupci proťala nejkratší cesta z počátku do daného vrcholu $(n/2)$-tý řádek. To si postačí pamatovat jen v okolí aktuálního řádku. Na konci výpočtu zjistíme, jaký je prostřední vrchol na nejkratší cestě, takže umíme problém rozdělit na dva poloviční podproblémy. Doplňte detaily a dokažte, že tomuto algoritmu postačí lineární paměť, a přesto celkově poběží v čase $\Theta(nm)$.

11. *Součin matic:* Násobíme-li matice $X \in \mathbb{R}^{a \times b}$ a $Y \in \mathbb{R}^{b \times c}$ podle definice, počítáme $a \cdot b \cdot c$ součinů čísel. Pokud chceme spočítat maticový součin $X_1 \times \ldots \times X_n$, výsledek nezávisí na uzávorkování, ale časová složitost (měřená pro jednoduchost počtem součinů čísel) ano. Vymyslete algoritmus, který stanoví, jak výraz uzávorkovat, abychom složitost minimalizovali.