14. cvičení z ADS1

3. 6. 2021

Příklad 1.

Zatím umíme pro mnoho úloh dynamického programování najít optimální řešení. Jakým způsobem ale zařídíme to, abychom obecně uměli řešení zrekonstruovat?

Například pro editační vzdálenost kromě minimální vzdálenosti rovněž vrátíme posloupnost úprav, kterou se dostaneme z prvního na druhé slovo.

Příklad 2.

Ukažte, že pokud známe horní odhad na editační vzdálenost $D' \geq D$, tak potom umíme zrychlit algoritmus na nalezení editační vzdálenosti na čas $\mathcal{O}((n+m) \cdot D')$.

Příklad 3.

Ukažte, jak využít předchozího pozorování k tomu, aby algoritmus na editační vzdálenost běžel v čase $\mathcal{O}((n+m) \cdot D)$, i když D neznáme.

***Příklad 4.**

Chceme nalézt optimální vyhledávací strom pomocí dynamického programování. Knuthova nerovnost říká, že $R[\ell, p-1] \leq R[\ell, p] \leq R[\ell+1, p]$, kde R je kořen optimálního podstromu. Pomocí této nerovnosti zrychlete algoritmus, aby běžel v čase $\mathcal{O}(n^2)$.

Příklad 5.

Dešifrovali jsme tajnou depeši, ale chybí v ní mezery. Známe však slovník všech slov, která se v depeši mohou vyskytnout. Chceme tedy rozdělit depeši na co nejméně slov ze slovníku.

Příklad 6.

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 $X_1 \times \cdots \times X_n$, výsledek nezávisí na uzávorkování, ale časová složitost ano. Vymyslete algoritmus, který stanoví, jak výraz uzávorkovat, abychom složitost minimalizovali.