12. CVIČENÍ Z ADS 1, ČTVRTEK 15:40, LS '24

Dynamické programování (bez programování a bez dynamických dat)

1. *Kopec.* Kopcem nazveme podposloupnost, která nejprve roste a pak klesá. Vymyslete algoritmus, který v zadané posloupnosti nalezne nejdelší kopec. (Můžete používat algoritmy z přednášky.)

2. *Počítání NRP.* Nejdelší rostoucích podposloupností může být v posloupnosti více. Jak spočítat, kolik různých nejdelší rostoucích podposloupností obsahuje zadaná posloupnost?

3. *Knihovna.* Mějme posloupnost n knih. Každá kniha má nějakou šířku s_i a výšku v_i. Knihy chceme naskládat do knihovny s nějakým (předem neurčeným) 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 přitom zanedbáváme.

4. *Nejdelší společná podposloupnost.* Navrhněte algoritmus pro nalezení nejdelší společné podposloupnosti daných posloupností $x_1, \ldots, x_n$ a $y_1, \ldots, y_m$. Jak tento problém souvisí s editační vzdáleností? Jaký je grafový pohled na výsledný algoritmus?

5. *Editační vzdálenost rychleji.* Na první pohled se zdá, že čím podobnější řetězce dostaneme, tím by mělo být jednodušší zjistit jejich editační vzdálenost. Algoritmus z přednášky ovšem pokaždé vyplňuje celou tabulku dynamického programování. Ukažte, jak ho zrychlit, aby počítal v čase $O((n+m)\cdot(ED(s,t)+1))$, kde $|s|=n$, $|t|=m$ a $ED(s,t)$ je editační vzdálenost s a t.

Hint: Nejprve zkuste vymyslet algoritmus, který dostane číslo $D \geq 1$ a v čase $O((n+m)\cdot D)$ buď korektně ohlásí, že $ED(s,t) > D$, nebo vrátí $ED(s,t)$.

6. *Násobení mnoha 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 (používáme přímočarý algoritmus z definice násobení). Pokud chceme spočítat maticový součin $X_1 \cdot X_2 \cdots X_n$, výsledek nezávisí na uzávorkování, ale časová složitost ano (pro jednoduchost ji budeme měřit počtem součinů čísel). Vymyslete algoritmus, který stanoví, jak výraz uzávorkovat, abychom počet součinů čísel minimalizovali.

7. *Knihovna 2.* Podobně jako úloze 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.