12. cvičení

Datové struktury I, 20. 12. 2022 https://iuuk.mff.cuni.cz/~chmel/2223/ds1/

Úloha 1 (Sufixové a LCP pole v praxi)
Společně si ukážeme jak sestavit obě pole na řetězci `CAGTAGCTGTA`.
Zkuste si to potom sami na řetězci `TATGTCAGTATCTC`.

Úloha 2 (Skoro advent of code)
Ze sopky se vám podařilo zachránit hromadu slonů[1]. Díky vašim expertním znalostem různých sloních druhů jste zpozorovali, že ve skutečnosti máte slony dvou různých druhů, a protože oba druhy mají jiné preference potravy, chcete je od sebe rozdělit, ať se na vás některý ze slonů nenaštve. Bohužel je od sebe nejste schopni odlišit přímo, ale víte, že se liší svou DNA. Navíc jako správní matfyzáci s sebou máte přístroj na sekvenování DNA, který zároveň automaticky vygeneruje i sufixové a LCP pole daného řetězce DNA.
Každému slonovi tedy (s jejich souhlasem) vysekvenujete relevantní kousek DNA, který si můžeme představit jako řetězec. Po důkladném prostudování příručky *Jak se liší sloni* jste zjistili, že hlavní rozdíl jejich DNA je v délce nejdelšího opakujícího se podřetězce bází DNA. Konkrétně, pokud je délka takového řetězce lichá, jedná se o slona indického, zatímco slon africký má nejdelší takový podřetězec sudé délky. Pro každého slona určete, zda se jedná o slona afrického či slona indického.
Pokud vás nezajímají příběhy: nalezněte v řetězci, když máte jeho sufixové i LCP pole, délku nejdelšího opakujícího se podřetězce.

Úloha 3 (Sloni jsou nějak zvědaví)
Když se sloni dozvěděli, co všechno váš sekvencer umí, zajímala by je ještě další věc: přečetli si, že podřetězec P délky n odpovídá zvýšené inteligenci, a tak by je zajímalo, jestli mají takové predispozice. Když máte relevantní část jejich genomu G délky m, nejprve si připomeňte triviální algoritmus hledání jehly v seně v čase $\mathcal{O}(n \log m)$ (můžete předpokládat, že sufixové pole máte už sestavené).
Dále si představte, že pro libovolné dva řetězce x, y umíte spočítat $\mathrm{LCP}(x, y)$ v konstantním čase. Upravte triviální algoritmus tak, aby trval jenom $\mathcal{O}(\log m)$.

Bonusové úlohy

Úloha 4 (Advent of code intensifies)
Sloni jsou ale realisté, takže si jsou vědomi toho, že obecné LCP nejde spočítat v konstantním čase. Na druhou stranu ale objevili hromadu dalších podřetězců, které údajně odpovídají všem možným vlastnostem, takže by byli moc rádi, kdyby algoritmus byl co nejrychlejší.
Co ale kdybychom uměli v konstantním čase spočítat LCP libovolných dvou *sufixů* v genomu (seně)?[2] Nalezněte v tom případě algoritmus na hledání jehly délky n v seně délky m běžící v čase $\mathcal{O}(n + \log m)$.
Hint: *V sufixovém poli máme všechny sufixy G lexikograficky seřazené, takže v něm můžeme binárně vyhledávat a zajímá nás takový sufix, jehož prefixem je právě P. Jenom je potřeba při porovnání řetězců porovnávat jenom ty správné části, aby nás všechna porovnání dohromady stále $\mathcal{O}(n)$. Cílem je tedy využívat informace o společných prefixech mezi prvky sufixového pole k udržování informací o společném prefixu P s relevantními řetězci v S.*

Úloha 5 (Vlastně můžeme trošku zeslabit předpoklady)
Víme, že v lineárním čase lze k S dopočítat pole LCP (Kasaiovým algoritmem). Ukažte, že potom je v lineárním čase možné dopočítat všechny hodnoty, které algoritmus v předchozí úloze potřebuje.

[1] Viz příběh advent of code 2022, dny 16-18: `https://adventofcode.com/2022/day/16`.
[2] Tento předpoklad je silnější než jenom mít LCP pole, protože umíme počítat i LCP lexikograficky nesousedních sufixů.