11. cvičení

Datové struktury I, 18. 12. 2025 https://iuuk.mff.cuni.cz/~chmel/2526/ds1/

Úloha 1 (Nejhorší případ pro k-d strom)
Mějme 2-d strom, tedy binární strom, kde rozdělujeme rovinu střídavě podle os x a y. Z přednášky víme, že dvourozměrný intervalový dotaz (na počet bodů v obdélníku) může trvat až $\Omega(\sqrt{n})$, konkrétně si připomeňte jak najít množinu bodů uloženou ve stromě a dotaz, který bude trvat takto dlouho, a přitom nenajde žádný bod.
Zkuste toto rozšířit na obecné k-d stromy s časovou složitostí $\Omega(n^{1-\frac{1}{k}})$.

Úloha 2 (Zrychlení intervalových stromů)
Mějme zjednodušený dvourozměrný intervalový strom, tedy binární strom podle souřadnice x, který má v každém vrcholu pole bodů z daného x-ového intervalu seřazené podle y. Dvourozměrné intervalové dotazy (na počet bodů v obdélníku) zde trvají $\mathcal{O}(\log^2 n)$, protože potřebujeme binárně vyhledávat v $\mathcal{O}(\log n)$ polích. Ukažte, že přidáním (lineárně mnoha) pointerů z každého pole na správná místa v obou polích v synech můžeme složitost snížit na $\mathcal{O}(\log n)$.
(Pro d-rozměrný strom pak dostaneme $\mathcal{O}(\log^{d-1} n)$.)

Úloha 3 (Hledání minima a mazání v k-d stromě)
Máme k-d strom. Navrhněte, jak na něm implementovat operaci nalezení nejmenšího prvku v dané dimenzi. (Tedy funkci FindMin(d), která nalezne prvek, jehož d-tá souřadnice je minimální.)
Dále navrhněte implementaci operace Delete. Pro jednoduchost můžete předpokládat, že žádné dva body spolu nesdílejí žádnou souřadnici.
(V obou případech neřešte časovou složitost.)

Úloha 4 (Nejbližší soused a k-NN)
Máme k-d strom. Navrhněte, jak na něm implementovat operaci nalezení nejbližšího souseda, a operaci nalezení m nejbližších sousedů.
Hint: pro m-NN se může hodit mít k dispozici haldu.

Bonusové úlohy

Úloha 5 (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 LCP(x, y) v konstantním čase. Upravte triviální algoritmus tak, aby trval jenom $\mathcal{O}(\log m)$.

Úloha 6 (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)?[1] 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 stála $\mathcal{O}(n)$. 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 7 (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] Tento předpoklad je silnější než jenom mít LCP pole, protože umíme počítat i LCP lexikograficky nesousedních sufixů.

Tahák pro řetězce

- Pro řetězec α s pythoním značením podřetězců (tj. indexujeme od nuly, $\alpha[i:j]$ obsahuje všechny znaky od indexu i po index $j-1$, vynechání jedné souřadnice znamená "od začátku", resp. "do konce") máme následující:
- sufixové pole S: $S[i]$ říká index takový, že $\alpha[S[i]:]$ je lexikograficky i-tý sufix řetězec α
- rankové pole R: $R[i]$ říká, kolikátý je lexikograficky sufix $\alpha[i:]$
- LCP pole L: $L[i]$ říká, kolik znaků na začátku spolu sdílejí $\alpha[S[i]:], \alpha[S[i+1]:]$ (tedy lexikograficky i-tý a $(i+1)$-ní sufix α)

Domácí úkol

Úloha 1 (Hezké svátky!)
Užijte si vánoční prázdniny!