

1. *Aho-Corasicková*: Sestrojte vyhledávací automat pro slova *ara*, *bar*, *arab*, *baraba*, *barbara*.
2. *Příliš mnoho výskytů*: Nalezněte příklad jehel a sena, v němž je asymptoticky více než lineární počet výskytů. Přesněji řečeno ukažte, že pro každé n existuje vstup, v němž je součet délek jehel a sena $\Theta(n)$ a počet výskytů není $\mathcal{O}(n)$.
3. *Naivní skákání*: Uvažujme zjednodušený algoritmus AC, který nepoužívá zkratkové hrany a vždy projde po zpětných hranách až do kořene. Ukažte vhodnými příklady vstupů, že tento algoritmus je asymptoticky pomalejší.
4. *Předpočítané množiny*: Jednoduchý způsob, jak si poradit s hlášením výskytů, je předpočítat si pro každý stav s množinu $M(s)$ slov k ohlášení. Dokažte, že tyto množiny není možné sestavit v lineárním čase s velikostí slovníku, protože součet jejich velikostí může být pro některé vstupy superlineární.
5. *Frekvence výskytů*: Popište algoritmus, který v lineárním čase pro každou jehlu spočítá, kolikrát se v seně vyskytuje. Časová složitost by neměla záviset na počtu výskytů – ten, jak už víme, může být superlineární.
6. *Podposloupnosti*: Popište algoritmus, který pro danou množinu jehel zjistí které z nich se v seně vyskytují jako podposloupnosti.
7. *Cenzor*: Cenzor dostane množinu zakázaných podřetězců a text. Vždy najde nejlevější výskyt zakázaného podřetězce v textu (s nejlevějším koncem; pokud jich je více, tak nejdelší takový), vystřihne ho a postup opakuje. Ukažte, jak text cenzurovat v lineárním čase.

8. *Dynamické vyhledávání*: Navrhněte datovou strukturu pro dynamické vyhledávání v textu. Jehla je pevná, v seně lze průběžně měnit jednotlivé znaky a struktura odpovídá, zda se v seně právě vyskytuje jehla.
9. *Nejmenší slovo bez výskytu*: Na vstupu dostanete množinu slov ι_1, \dots, ι_n nad anglickou abecedou a přirozené číslo l . Najděte lexikograficky nejmenší slovo délky l , které neobsahuje ι_i pro každé i .
10. *Nejkratší univerzální slovo*: Na vstupu dostanete množinu slov ι_1, \dots, ι_n nad anglickou abecedou. Najděte nejkratší slovo délky obsahující všechny ι_i . Cílíme na algoritmus s časovou složitostí $\mathcal{O}(2^n \sum_{i=1}^n J_i)$.
11. *Permutační podslovo*: Řekneme že dvě posloupnosti x a y stejné délky jsou stejně uspořádané, pokud pro každé i a j platí $x[i] \leq x[j]$ právě tehdy když $y[i] \leq y[j]$. Permutace délky n je posloupnost čísel $\{1, \dots, n\}$ v libovolném pořadí. Navrhněte algoritmus, který pro permutaci σ délky n a permutaci ι délky m rozhodne jestli existuje podslovo σ délky m stejně uspořádané jako ι . Například **3, 1, 5, 2, 4** obsahuje 2, 1, 3 (tučně vyznačeno), ale neobsahuje 3, 2, 1 jelikož nejdelší souvislý klesající úsek má délku pouze 2.