1. Mějme variantu MergeSortu, která rozkládá vstup místo dvou na k částí. Vyjádřete její časovou složitost pomocí n (délky vstupu) i k. Jak dopadnou různé volby k?
2. QuickSort a hashovací tabulky dosahují své obvyklé časové složitosti jen v průměrném případě. Jak vypadá pravděpodobnostní prostor, ve kterém tuto složitost počítáme – „přes co“ počítáme pravděpodobnost?
3. Ukažte, jak pomocí algoritmu na hledání k-tého nejmenšího prvku v lineárním čase získat variantu QuickSortu běžící v čase $\mathcal{O}(n \log n)$ i v nejhorším případě.
4. Ve stejném algoritmu používáme dělení vstupu na pětice. Záleží na tom, jak skládáme ze vstupu hodnoty do pětic? Jak by se změnila časová složitost algoritmu, kdybychom místo pětic použili nějaké p-tice?
5. Najděte algoritmus, který pro zadanou posloupnost čísel zjistí, kolik obsahuje nejdelších rostoucích podposloupností.
6. Problém editační vzdálenosti zformulujte jako grafový problém. Vyřešte modifikovaný problém, kdy každému znaku (konkrétně každému indexu řetězce) dáme číselnou váhu a chceme najít posloupnost úprav s nejmenším součtem vah editovaných znaků.
7. Posloupnost je *bitonická*, pokud se skládá z rostoucí a klesající části. Popište algoritmus, který v posloupnosti čísel najde nejdelší bitonickou podposloupnost.

