jedenácté cvičení ADS 1

Příklad 1 (slovní žebřík): Je dán slovník. Sestrojte co nejdelší slovní žebřík. To je posloupnost slov ze slovníku taková, že $(i+1)$-ní slovo získáme z i-tého smazáním jednoho písmene. Například pro anglický slovník AGID-4 vychází žebřík glassiness, glassines, glassine, glassie, lassie, lassi, lass, ass, as, a.

Příklad 2 (hladký sled): Mějme orientovaný graf s celočíselně ohodnocenými hranami. Sled nazveme k-hladký, pokud se ohodnocení každých dvou na sebe navazujících hran liší nejvýše o k. Navrhněte algoritmus, který nalezne 3-hladký sled o co nejméně hranách mezi zadanými dvěma vrcholy.

Příklad 3 (podposloupnost): Je dána posloupnost celých čísel. Najděte nejdelší úsek, ve kterém se neopakují hodnoty.

Příklad 4 (nejdelší nejkratší cesta): Mějme graf ohodnocený kladnými čísly s počátečním a koncovým vrcholem. Najděte tu z nejkratších cest mezi těmito vrcholy, která obsahuje největší možný počet hran.

Příklad 5 (mazání vrcholů): Je dán souvislý neorientovaný graf. Nalezněte pořadí, v jakém můžeme mazat vrcholy tak, abychom postupně smazali všechny a graf byl v průběhu mazání stále souvislý.

Příklad 6 (strážníci): Mějme plán městečka ve tvaru stromu. Hrany jsou ulice, vrcholy křižovatky. Na křižovatku lze umístit strážníka, ten pak hlídá všechny ulice sousedící s křižovatkou. Strážníkovi ovšem musíme zaplatit, mezi křižovatkami se liší, kolik. Vymyslete, jak co nejlevněji rozmístit strážníky tak, aby všechny ulice byly hlídané.

Příklad 7 (díra): Mějme množinu celých čísel. Díra budeme říkat dvojici prvků množiny, mezi nimiž neleží žádný další prvek. Jak najit největší díru v lineárním čase?

Příklad 8 (hledání hran): Navrhněte datovou strukturu pro udržování neorientovaného grafu s ohodnocenými hranami, která bude umět operace

1. Insert(u,v,váha): vložení hrany uv,
2. Min(v): nalezení nejlevnější hrany v komponentě souvislosti obsahující vrchol v.

Příklad 9 (zásobník): Navrhněte datovou strukturu pro zásobník, která podporuje vedle operací Push a Pop ještě operaci FindMin, která vrátí nejmenší hodnotu na zásobníku. Zkuste najít řešení s konstantní časovou složitostí pro každou operaci a která oproti normálnímu zásobníku potřebuje jen o $O(1)$ více paměti.

Příklad 10 (Fibonacci): Fibonacciho číselná soustava funguje tak, že čísla zapisujemepomocí nul a jedniček a zápis $(c_n \dots c_2)$ odpovídá číslu $\sum_i c_i F_i$, kde F_i je i-té Fibonacciho číslo $(F_0 = 0, F_1 = 1, F_{n+2} = F_n + F_{n+1})$. Zápisu, v němž se nevyskytují dvě jedničky za sebou, budeme říkat hezký. Platí, že každé přirozené číslo má právějeden hezký zápis. Vymyslete algoritmus, který v lepším než kvadratickém čase sečte dvě čísla zadaná hezkými Fibonacciho zápisy a vydá hezký zápis výsledku.