Za domácí úkol máte příklady 1 a 2. Důkladně si přečtěte zadání!

U všech níže uvedených příkladů se snažte najít algoritmus, který zadanou úlohu vyřeší co možná nejrychleji s co nejmenší spotřebou paměti. Vždy uvádějte časovou a paměťovou složitost vašich algoritmů i s důkazy správnosti vašich tvrzení.

Příklad 1. Rozhodněte a dokažte, zda pro každé dvě *neklesající* funkce $f, g : \mathbb{Z}^+ \to \mathbb{Z}^+$ platí, že $f \in \mathcal{O}(g)$ nebo $g \in \mathcal{O}(f)$.

Příklad 2. Mějme orientovaný graf G bez orientovaných kružnic a dva jeho vrcholy u a u. Najděte *nejdelší orientovanou* cestu z u do v. Jak rychle byste zvládli najít nejdelší cestu mezi libovolnými dvěma vrcholy?

Příklad 3. Uvažujme obecný algoritmus na průchod grafu G z počátečního vrcholu s využívající libovolnou datovou strukturu DS.

```
1 Označ všechny vrcholy jako nenavštívené
2 Vlož počáteční vrchol s do DS a označ jej jako navštívený
3 d_u ← 0 while DS je neprázdná do
4     Nechť u je libovolný vrchol u z DS
5     for hrany z vrcholu u do vrcholu v do
6         if v je nenavštívený then
7             Vlož v do DS a označ jej jako navštívený
8             Označ hranu uv jako stromovou d_v ← d_u + 1
9     Odstraň vrchol u z DS a označ jej u jako prozkoumaný
```

Tento algoritmus má následující vlastnosti

1. DS obsahuje všechny aktuálně navštívené, ale neprozkoumané vrcholy. Nenavštívené vrcholy ještě v DS nebyly a prozkoumané byly z DS odstraněny.
2. Stromové hrany tvoří strom T v grafu G. V orientovaných grafech vytvoří strom orientovaný z s do všech dosažitelných vrcholů. V neorientovaných grafech je T kostrou komponenty obsahující s.
3. Algoritmus vždy skončí.
4. Hodnota d_u je délka cesty z s do u ve stromě T.

Otázky:

1. Pro stromovou hranu uv platí, že $|d_u - d_v| = 1$. Platí podobná vlastnost i pro nestromové hrany? Může existovat hrana uv taková, že $d_u = d_v$? Jak velký může být rozdíl $d_u - d_v$? Změní se odpovědi na tyto otázky, jestliže DS je zásobník nebo fronta?
2. Jak můžeme najít nejkratší cestu do všech vrcholů?

Příklad 4. Mějme bludiště zadané grafem. Víme, ve kterém vrcholu se nachází princezna a ve kterém se nachází vchod. Dále víme, ve kterých chodbách stojí drak, kterého je nutné zabít, abychom mohli projít. Najděte cestu k princezně, po které je nutné zabít co nejmenší počet draků (je jich málo, a tak jsou chráněni).

Příklad 5. Mějme bludiště zadané grafem. Víme, ve kterém vrcholu se nachází princezna a ve kterém se nachází vchod. Dále pro každou hranu známe vrchol, ve kterém se nachází klíč odemykající danou hranu. Najděte algoritmus, který rozhodne, zda je možné princeznu vysvobodit.

Příklad 6. Mějme neorientovaný graf G a počáteční vrchol s. Vymyslete algoritmus, který pro každý vrchol v najde nejen délku nejkratší cesty z s do v, ale i počet nejkratších cest. Dvě cesty z s do v považujeme za různé, pokud se liší v alespoň jednom vrcholu.

Příklad 7. V zadaném stromu najděte nejdelší cestu.

Příklad 8. Máme šachovnici velikosti $n \times n$. Jak najít nejmenší počet tahů koně, kterými přesuneme z daného počátečního místa do daného cílového místa.

Příklad 9. Navrhněte algoritmus, který rozhodne, zda zadaný graf je bipartitní. Tak se říká grafům, jejichž vrcholy lze rozdělit na dvě množiny tak, aby koncové vrcholy každé hrany patřily do různých množin.