1. *F-F s celými čísly:* Jak rychle doběhne Ford-Fulkersonův algoritmus pro jednotkové váhy a jak rychle pro celočíselné?

2. *Dinic s celými čísly:* Jak rychle doběhne Dinicův algoritmus pro jednotkové váhy a jak rychle pro celočíselné? (Ideálně bychom chtěli aby běžel alespoň stejně rychle jako F-F).

3. *Hranově disjunktní cesty:* Navrhněte algoritmus pro nalezení maximálního počtu hranově disjunktních cest mezi danými dvěma vrcholy $u, v \in V(G)$.

4. *Vrcholově disjunktní cesty:* Navrhněte algoritmus pro nalezení maximálního počtu vrcholově disjunktních cest mezi danými dvěma vrcholy $u, v \in V(G)$.

5. *Věže na šachovnici:* Mějme šachovnici $r \times s$, z níž políčkožrout sežral některá políčka. Chceme na ni rozestavět co nejvíce šachových věží tak, aby se navzájem neohrožovaly. Věž můžeme postavit na libovolné nesežrané políčko a ohrožuje všechny věže v témže řádku i sloupci. Navrhněte efektivní algoritmus, který takové rozestavění najde.

6. *Věže podruhé:* Stejně jako předchozí příklad, ale tentokrát věž „nevidí“ přes sežraná políčka.

7. *Nekonečný F-F:* Najděte síť s reálnými kapacitami, na níž Fordův-Fulkersonův algoritmus nedoběhne. Lze zařídit, aby k maximálnímu toku ani nekonvergoval?

8. *Nečekaná záchrana:* Přímočará implementace Fordova-Fulkersonova algoritmu bude nejspíš graf prohledávat do šířky, takže vždy najde nejkratší nenasycenou cestu. Pak překvapivě platí, že algoritmus zlepší tok jen $\mathcal{O}(nm)$-krát, než se zastaví. Návod k důkazu: Nechť $\ell(u)$ je vzdálenost ze zdroje do vrcholu u po nenasycených hranách. Nejprve si rozmyslete, že $\ell(u)$ během výpočtu nikdy neklesá. Pak dokažte, že mezi dvěma nasyceními libovolné hrany uv se musí $\ell(u)$ zvýšit. Proto každou hranu nasytíme nejvýše $\mathcal{O}(n)$-krát.

9. *k-souvislost:* Díky 2. cvičení umíme pro dva vrcholy u, v zjistit, zda mezi nimi vede k hranově disjunktních cest. Pomocí tohoto algoritmu zjistěte zda je neorientovaný graf G hranově k-souvislý. (Graf je hranově k-souvislý pokud mezi každými dvěma vrcholy vede k hranově disjunktních cest.) Váš algoritmus nemá dostatek času na to, aby zavolal $\Theta(n^2)$-krát algoritmus z 2. cvičení.

10. *Algoritmus tří Indů:* Blokující tok ve vrstevnaté síti lze nalézt chytřejším způsobem v čase $\mathcal{O}(n^2)$, čímž zrychlíme celý Dinicův algoritmus na $\mathcal{O}(n^3)$. Následuje stručný popis, doplňte k němu detaily.

    Pro každý vrchol v definujme $r^+(v)$ jako součet rezerv na všech hranách vedoucích do v. Nechť dále $r^-(v)$ je totéž přes hrany vedoucí z v a $r(v) = \min(r^+(v), r^-(v))$ „rezerva vrcholu“. Pokud je $r(v)$ všude 0, tok už je blokující.

    V opačném případě opakovaně vybíráme nejmenší $r(v)$ a snažíme se ho vynulovat. Potřebujeme tedy dopravit $r(v)$ jednotek toku ze zdroje do v a totéž množství z v do stoku. Popišme dopravu do stoku (ze zdroje postupujeme symetricky): ve vrcholech udržujeme plán $p(w)$, který říká, kolik potřebujeme z w dopravit do stoku. Na začátku je $p(v) = r(v)$ a všechna ostatní $p(w) = 0$. Procházíme po vrstvách od v ke stoku a pokaždé plán převedeme po hranách s kladnou rezervou do vrcholů v další vrstvě. Jelikož $r(v) \leq r(w)$ pro všechna w, vždy nám to vyjde. Průběžně čistíme slepé uličky.