1. Může mít algoritmus časovou složitost o(n)? Jakou mají takové algoritmy vlastnost?

2. Dokažte nebo vyvraťte:
    a. f(n) je v O(g(n)), potom g(n) je v O(f(n))
    b. f(n) je v O(g(n)), potom 2^f(n) = O(2^g(n))
    c. f(n) je v O(g(n)), právě tehdy když g(n) je v Omega(f(n)),

3. Najděte chybu. Dokazujeme, že f(n) = n^2 je v O(n) indukcí:
    a. n = 1 platí.
    b. Indukční krok: f(n) = O(n)+f(n−1) = O(n)+O(n) (z predpokladu) = O(n).

4. Koupili jste na inzerát dvojici skvělých robotů. Lacino, neboť jsou právě uvěznění v bludišti (čtvercová síť s některými políčky blokovanými). Znáte jejich polohy a můžete jim rádiem vysílat povely pro posun o políčko na sever, jih, východ či západ, abyste je dostali na okraj bludiště. Háček je ale v tom, že na každý povel reagují oba roboti. Vymyslete algoritmus, který najde nejkratší posloupnost povelů, jež vysvobodí oba roboty. Dodejme ještě, že robot ignoruje povel, který by způsobil okamžitý náraz do zdi, a že jakmile se robot dostane na okraj, odchytíme ho a další povely neposlouchá. (Poznamenejme, že úloha by byla řešitelná i tehdy, kdybychom počáteční polohy robotů neznali. To je ovšem mnohem obtížnější a potřebná „univerzální posloupnost povelů“ mnohem delší.)

5. Jak probíhá BFS a DFS úplného neorientovaného bipartitního grafu? Jaké druhy hran vznikají? Zkuste pro stejně velké i různě velké partity.

6. Průměr grafu je definovaný jako nejdelší z nejkratších cest mezi libovolnými dvěma vrcholy. Spočítejte průměr grafu:
    a. Pro jednotkově ohodnocený graf pomocí BFS
    b. Pro kladně ohodnocený graf pomocí Dijkstry

7. Navrhněte algoritmus, který pro orientovaný graf spočítá tranzitivní uzávěr, tj. doplní hranu (x,y), pokud vede cesta z x do y.

8. V Tramtárii jezdí po železnici samé rychlíky, které nikde po cestě nestaví. V jízdním řádu je pro každý rychlík uvedeno počáteční a cílové nádraží, čas odjezdu a čas příjezdu. Nyní stojíme v čase *t* na nádraží *a* a chceme se co nejrychleji dostat na nádraží *b*. Navrhněte algoritmus, který najde takové spojení. Co když chceme mezi všemi nejrychlejšími spojeními najít takové, v němž je nejméně přestupů.

9. Silnice v mapě máme ohodnocené dvěma čísly: délkou a mýtem (poplatkem za projetí). Jak najít nejlevnější z nejkratších cest?

10. Jak hledat minimální kostru za předpokladu, že se určené vrcholy musí stát jejími listy? Jako další listy můžete využívat i neoznačené vrcholy.

11. Okénkový medián: Na vstupu postupně přicházejí čísla. Kdykoliv přijde další, vypište medián z posledních k čísel. Dosáhněte časové složitosti O(log k) na operaci.

12. Spletitý kabel: Mějme dlouhý kabel, z jehož obou konců vystupuje po n drátech. Každý drát na levém konci je propojen s právě jedním na konci druhém a my chceme zjistit, který s kterým. K tomu můžeme používat následující operace: (1) přivést napětí na daný drát na levém konci, (2) odpojit napětí z daného drátu na levém konci, (3) změřit napětí na daném drátu na pravém konci. Navrhněte algoritmus, který pomocí těchto operací zjistí, co je s čím propojeno. Snažte se počet operací minimalizovat.

13. Inverze matice: Navrhněte algoritmus typu Rozděl a panuj na výpočet inverze trojúhelníkové matice n × n v čase lepším než Ω(n 3). Jako podprogram se může hodit Strassenovo násobení matic. Můžete předpokládat, že n je mocnina dvojky.

14. Inverze v posloupnosti x_1, ... , x_n říkáme každé dvojici (i, j) takové, že i < j a současně x_i > x_j. Vymyslete algoritmus, který spočítá, kolik daná posloupnost obsahuje inverzí.

15. Šroubky a matičky: Na stole leží n různých šroubků a n matiček. Každá matička pasuje na právě jeden šroub a my chceme zjistit, která na který. Umíme ale pouze porovnávat šroub s matičkou – tím získáme jeden ze tří možných výsledků: matička je příliš velká, příliš malá, nebo správně velká. Nalezněte co nejefektivnější algoritmus.