Algoritmy pracující s grafy. Prohledávání grafů do hloubky a do šířky, využití prohledávání grafů v dalších úlohách.

Slovo **graf** se používá v různých významech. V matematice také existuje pojem grafu, který nemá takměř nic společné se známými grafy funkcí, až na název, který je odvozený z řeckého slova "grafein", co znamená "psát".
Pojem "graf" vykrystalizoval jako matematický objekt (univerzální zobrazovací technika), který se skládá z prvků dvojakého druhu – z vrcholů a hran.

Obyčejný graf G rozumíme uspořádanou dvojicí (V,E), tzn. **G = (V,E)**, V je libovolná množina vrcholů a E je podmnožina hran.

Sled v grafu G je konečná posloupnost S=(v0,e1,v1,e2,...vn-1,en,vn), v které se střídají vrcholy a hrany, a která se začíná a končí ve vrchole, přičemž pro všechny i=1,2,3,....n je ei (vi-1, vi). V sledu se hrany a vrcholy můžou opakovat. Číslo n nazýváme délkou sledu S. Sled nazýváme uzavřený, když jeho začátek a konec jsou totožné. V opačném případě ho nazýváme otevřený.

Tah je takový sled, v kterém se žádna hrana nevyskytuje dvakrát.
Otevřený tah, v kterém jsou všechny vrcholy navzájem různé, se nazývá cesta. Počet hran v cestě se nazývá délka cesty.

Strom je souvislý graf, neobsahující kružnici (acyklický souvislý graf, tj. uzavřený sled).

Podgraf G´ = (V´,E´) je indukovaný podgraf grafu G , když G´ obsahuje všechny hrany spojující vrcholy z V´ v grafu G. Podgraf G´ indukovaný množinou vrcholů V´ v grafu G

Stupeň vrcholu v je počet hran incidentních s vrcholem v, označení: degG v (deg v)
δ(G) - minimální, Δ(G) – maximální, izolovaný vrchol v ... deg v = 0

Úplný (kompletní) graf - graf, ve kterém jsou každé dva vrcholy spojené hranou.
Označení Kn, n - počet vrcholů

Regulární graf - takový graf, jehož všechny vrcholy mají stejný stupeň. Regulární graf s vrcholy, které mají stupeň k se nazývá k-regulární.

Věta (princip sudosti):
Když má graf G má m hrán, tak součet stupňů všech jeho vrcholů se rovná 2m:

$$\sum_{v \in V} \deg_G v = 2m$$

Důsledek:
Počet vrcholů lichého stupně v grafu je sudý.

Algoritmy k nalezení minimální kostry

Jarníkův algoritmus

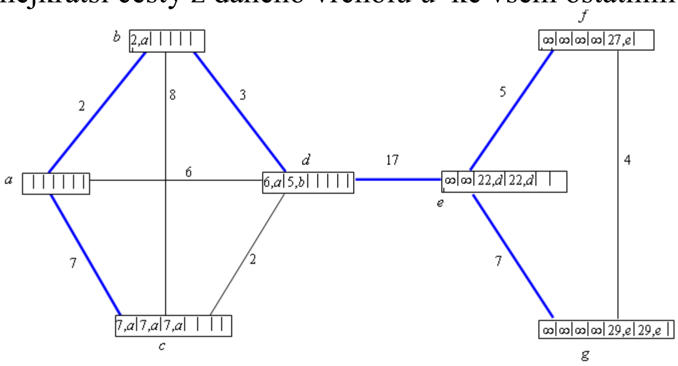
Algoritmus hledající minimální kostru ohodnoceného grafu. Najde takovou podmnožinu hran grafu, která tvoří strom obsahující všechny vrcholy původního grafu a součet ohodnocení hran z této množiny je minimální. Algoritmus začíná s jedním vrcholem a postupně přidává další nejbližší vrcholy(nejníže ohodnocená hrana) a tím zvětšuje velikost stromu do té doby, než obsahuje všechny vrcholy.

Kruskalův algoritmus
Algoritmus hledající minimální kostru ohodnoceného grafu jehož hrany mají nezáporné ohodnocení. Najde takovou podmnožinu hran grafu, která tvoří strom obsahující všechny vrcholy původního grafu a součet ohodnocení hran z této množiny je minimální. Vybírají se hrany s nejnižším ohodnocení, tak aby nevznikla kružnice.

Borůvkův algoritmus
Poprvé byl publikován roku 1926 Otakarem Borůvkou jako metoda pro konstrukci efektivní elektrické sítě na Moravě. Algoritmus pracuje tak, že postupně spojuje komponenty souvislosti (na počátku je každý vrchol komponentou souvislosti) do větších a větších celků, až zůstane jen jediný, a to je hledaná minimální kostra. V každé fázi vybere pro každou komponentu souvislosti hranu s co nejnižší cenou, která směřuje do jiné komponenty souvislosti a tu přidá do kostry. V každé fázi se počet komponent souvislosti sníží nejméně dvakrát, počet fází bude tedy maximálně log2N, kde N je počet vrcholů grafu.

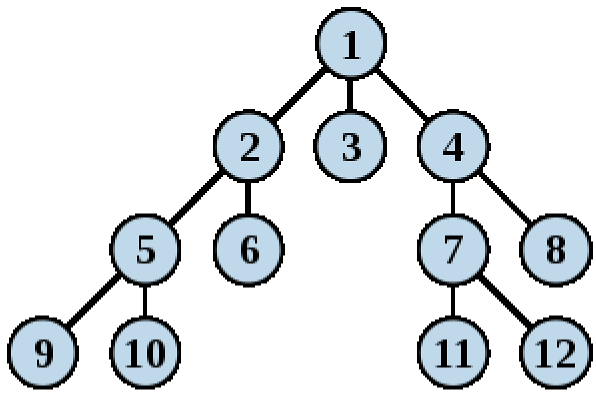
Dijkstrův algoritmus
Sloužící k nalezení nejkratší cesty v grafu. Je konečný (pro jakýkoliv konečný vstup algoritmus skončí), protože v každém průchodu cyklu se do množiny navštívených uzlů přidá právě jeden uzel, průchodů cyklem je tedy nejvýše tolik, kolik má graf vrcholů. Funguje nad hranově kladně ohodnoceným grafem (neohodnocený graf lze však na ohodnocený snadno převést).

Na počátku nechť je každá hrana grafu G neobarvená. Za modrý strom považujme počáteční vrchol u, od kterého hledáme nejkratší cesty k ostatním vrcholům.
Pro každý sousední vrchol v vrcholu u provedeme dv := délka hrany (u, v) a Vv := u.
V každém z (n - 1) kroků vybereme z hran, které se modrého stromu dotýkají (tj. mají jeden koncový vrchol x v modrém stromu a druhý koncový vrchol v nikoli), hranu (x, v) pro kterou platí, že hodnota dv je minimální (existuje-li jich více, zvolíme libovolnou z nich), obarvíme ji modře a pro všechny sousední vrcholy w vrcholu v provedeme v případě, že platí dv + délka hrany (v, w) < dw, příkazy Vw := v a dw := dv + délka hrany (v, w).
Algoritmus končí získáním modrého stromu obsahujícího všechny vrcholy grafu G, tj. určením nejkratší cesty z daného vrcholu u ke všem ostatním vrcholům.



Algoritmy prohledávání do hloubky a do šířky

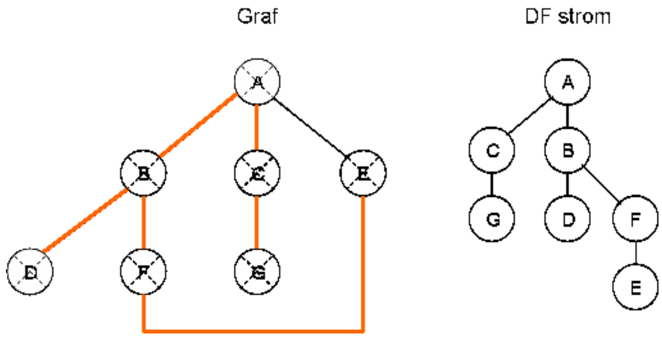
Prohledávání do šířky
Postupně prochází všechny vrcholy v dané komponentě souvislosti. Algoritmus nejprve projde všechny sousedy startovního vrcholu, poté sousedy sousedů atd. až projde celou komponentu souvislosti. Prohledávání grafu do šířky se realizuje pomocí fronty (FIFO). Začneme od libovolného vrcholu r. Výstup algoritmu – kostra grafu.



Pořadí v jakém je přistupováno k vrcholům

Prohledávání do hloubky

Pracuje tak, že vždy expanduje prvního následníka každého vrcholu, pokud jej ještě nenavštívil. Pokud narazí na vrchol, z nějž už nelze dále pokračovat (nemá žádné následníky nebo byli všichni navštíveni), vrací se zpět backtrackingem. Prohledávání grafu do hloubky se realizuje pomocí zásobníku (LIFO). Začneme od libovolného vrcholu r. Výstup algoritmu - kostra grafu



Binární strom

Každý vrchol má nejvýše dva následníky, levý a pravý syn

procedure PREORDER (*T*, *v*)

- probrání vrcholu v
- jestliže existuje levý syn v_L vrcholu v, pak PREORDER (T, v_L)
- jestliže existuje pravý syn v_P vrcholu v, pak PREORDER (T, v_P)
- návrat k přímému předchůdci (otci) vrcholu v

procedure INORDER (*T*, *v*)

- jestliže existuje levý syn v_L vrcholu v, pak INORDER (T, v_L)
- probrání vrcholu v
- jestliže existuje pravý syn v_P vrcholu v, pak INORDER (T, v_P)
- návrat k přímému předchůdci (otci) vrcholu v

procedure POSTORDER (*T*, *v*)

- jestliže existuje levý syn v_L vrcholu v, pak POSTORDER (T, v_L)
- jestliže existuje pravý syn v_P vrcholu v, pak POSTORDER (T, v_P)
- probrání vrcholu v
- návrat k přímému předchůdci (otci) vrcholu v

Využití prohledávání grafů v dalších úlohách

- Určení souvislosti daného grafu – musí být zpracovány všechny vrcholy ze zásobníku

- Určení počtu komponent daného grafu – do šířky, pokud nejsou zpracovány všechny vrcholy, navýšíme počitadlo o 1 a pokračujeme dalším vrcholem dle abecedy
- Určení, zda daná hrana je či není most daného grafu – prohledávání do hloubky, pokud jsou oba vrcholy odebrané hrany zpracovány, tak hrana není most
- Určení, zda dva dané vrcholy leží v téže komponentě – po prohledání musí být ve stejné množině
- Určení nejkratší cesty (co do počtu hran) a její délky – do šířky, délka cesty je úroveň, ve které se nachází cílový vrchol
- Určení, zda je graf bipartitní – do šířky, pokud není propojení větví na stejné úrovni
- Zda existuje kružnice obsahující daný vrchol – do šířky, pokud existuje propojení mezi dvěma podstromy, pak existuje kružnice obsahující vrchol