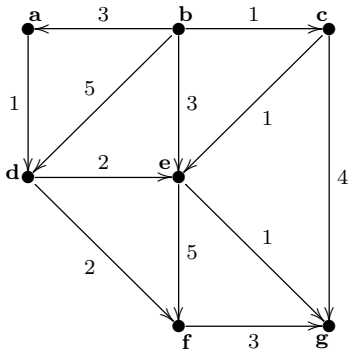
Základní grafové algoritmy

Dijkstrův algoritmus pro hledání nejkratších cest

Je dán orientovaný ohodnocený graf. (Lze uvažovat i neorientovaný s tím, že hrany můžeme procházet oběma směry.) Úkolem je najít nejkratší cestu z daného vrcholu do každého z ostatních vrcholů. Z daného vrcholu postupně „objevujeme“ další vrcholy (= navštívíme vrcholy, do kterých vede z daného vrcholu hrana) a zaznamenáváme délky doposud objevených cest. Když jsme některý vrchol už prozkoumali, jako další použijeme ten z dosud neprozkoumaných vrcholů, který má od počátečního vrcholu nejmenší vzdálenost. Pokud v některém kroku nalezneme kratší cestu, než máme doposud zaznamenáno, údaj přepíšeme. Tento algoritmus funguje pro každý graf bez záporně ohodnocených hran.

Příklad. Pomocí Dijkstrova algoritmu nalezněte v tomto grafu nejkratší cesty z vrcholu b do všech ostatních vrcholů. Průběh algoritmu zapište do připravené tabulky.



a	**b**	**c**	**d**	**e**	**f**	**g**
∞	0	∞	∞	∞	∞	∞

Algoritmy pro hledání minimální kostry

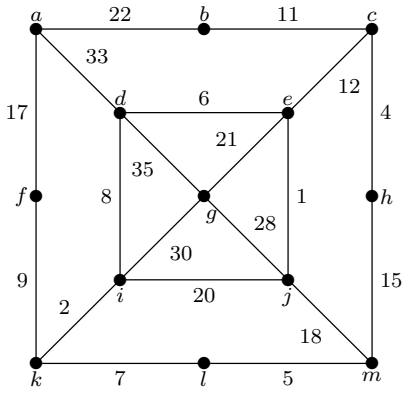
V ohodnoceném neorientovaném grafu chceme najít kostru minimální váhy. Tedy mezi všemi kostrami daného grafu hledáme tu s co nejmenším součtem ohodnocení na hranách. Pokud jsou ohodnocení všech hran grafu různá, má tato úloha vždy pouze jedno řešení. obecně to tak ale být nemusí.

1. **Kruskalův algoritmus:** Seřadíme hrany vzestupně podle jejich váhy a v tomto pořadí je přidáváme do kostry, přičemž dbáme na to, aby nevznikla kružnice. Algoritmus skončí ve chvíli, kdy už není možné přidat žádnou další hranu.

2. **Jarníkův-Primův algoritmus:** Začneme v libovolném vrcholu grafu a připojíme hranou vždy ten z dosud nepoužitých vrcholů, mezi nímž a již spojenou částí grafu vede hrana nejmenšího ohodnocení. Algoritmus skončí v momentě, kdy máme připojeny všechny vrcholy.

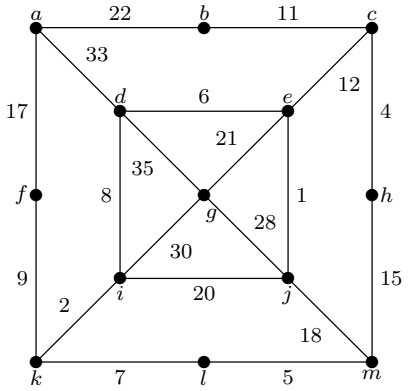
3. **Borůvkův algoritmus:** Zde je důležité, aby každá hrana v grafu měla unikátní ohodnocení. Na začátku jsou vrcholy samostatnými komponentami. V každém kroku se pro každou komponentu podíváme na hrany, které z ní vycházejí, a tu s nejmenší váhou přidáme do kostry (tj. spojili jsme některé komponenty, nyní máme komponent méně). Algoritmus skončí, když máme už jen jednu komponentu.

Příklad. Pomocí Kruskalova algoritmu nalezněte minimální kostru následujícího grafu a vyznačte ji. Zapište pořadí, ve kterém jste hrany přidávali.



Posloupnost přidávaných hran:

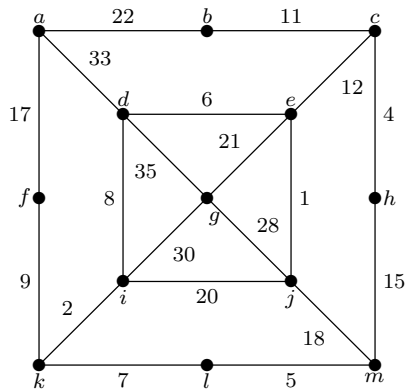
Příklad. Nyní vyřešte stejnou úlohu pomocí Jarníkova-Primova algoritmu. Zahajte jej ve vrcholu g.



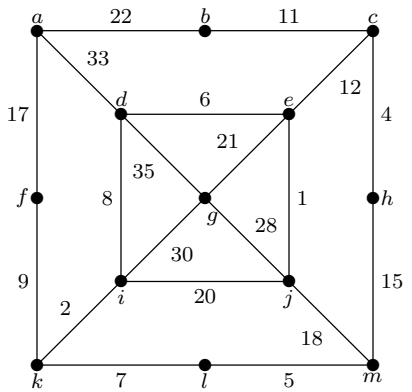
Posloupnost přidávaných vrcholů:

Příklad. Nakonec zkuste minimální kostru najít Borůvkovým algoritmem. Zakreslete stav po každém cyklu algoritmu a pokaždé zapište i příslušný rozklad množiny vrcholů.

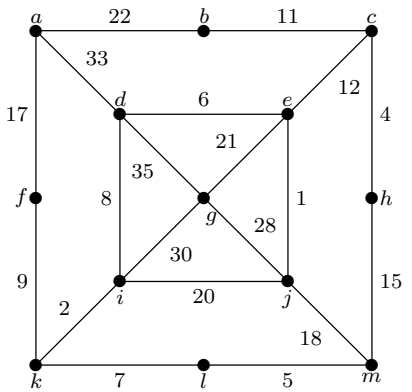
1. krok:



2. krok:



3. krok:



Odkazy na animace použitých algoritmů:

https://www.algoritmy.net/article/5108/Dijkstruv-algoritmus
https://www.algoritmy.net/article/1417/Kruskaluv-algoritmus
https://www.algoritmy.net/article/1393/Jarnik-Primuv-algoritmus
https://www.algoritmy.net/article/1396/Boruvkuv-algoritmus