9. Prehľadávanie grafov

9.1 Úvod

V celej tejto kapitole budeme pracovať len so súvislými obyčajnými grafmi. Na konci kapitoly 3 bol v podkapitole 3.5 (strana 62) definovaný pojem *kostra* grafu. Kostra grafu sa v aplikáciách často využíva a potrebujeme ju všade tam, kde je potreba nájsť „najmenší“ súvislý faktor daného grafu. Pritom pojem „najmenší“ je tu chápaný v zmysle počtu hrán. Algoritmus hľadania kostry grafu, použitý v dôkaze vety 3.5.1, bol veľmi neefektívny a prakticky nepoužiteľný. Avšak jeho triviálnou modifikáciou dostaneme nasledujúci algoritmus hľadania kostry daného grafu, ktorý je oveľa lepší/efektívnejší.

Konštrukcia kostry grafu

VSTUP: Súvislý obyčajný graf $G = (V,E)$.

VÝSTUP: $S = (V',E')$ – kostra grafu G.

INICIALIZÁCIA

$V' := V$;

$E' := \{\}$;

VYTVÁRANIE KOSTRY S

for $\{$ *pre všetky hrany* $h \in E$ $\}$ **do**
  if $\{$ *pridaním hrany* h *do* E' *nevznikne cyklus v* $S\}$ **then**
    $E' := E' \cup \{h\}$;
  end
  if $\{|E'| = |V'| - 1\}$ **then**
    STOP;
  end
end

Zistiť, či pridaním jednej konkrétnej hrany vznikne cyklus, je oveľa rýchlejšia operácia, než hľadanie všetkých cyklov v danom grafe. Preto je modifikovaný algoritmus, ktorý je uvedený na predošlej strane, oveľa efektívnejší než predtým uvedená alternatíva hľadania všetkých cyklov v danom grafe. Ale ani tento modifikovaný algoritmus hľadania kostry daného grafu ešte nie je optimálny. Ešte rýchlejšie a efektívnejšie sú dva prehľadávacie algoritmy, ktoré si uvedieme v nasledujúcich podkapitolách tejto kapitoly. Sú to algoritmy prehľadávania grafu *do šírky* a prehľadávania grafu *do hĺbky*.

9.2 Prehľadávanie grafu

Skôr, než uvedieme popis algoritmov prehľadávania grafu do šírky a do hĺbky, uvedieme si dátové štruktúry *fronta* a *zásobník*, ktoré nám popis oboch algoritmov zjednotia a zjednodušia. Pri prehľadávaní grafov budeme ich hrany ukladať práve do fronty, resp. do zásobníka.

9.2.1 Fronta (FIFO) a zásobník (LIFO)

Fronta (po anglicky *queue*) a *zásobník* (po anglicky *stack*) sú v informatike často používané dátové štruktúry, slúžiace na dočasné ukladanie dát. Tieto dve štruktúry sa odlišujú spôsobom prístupu k uloženým dátam.

Fronta – FIFO

Skratka FIFO znamená *„First-In-First-Out“*. Čiže dáta, ktoré ako prvé do fronty uložíme, z neho aj ako prvé vyberieme. Takúto dátovú štruktúru si môžeme predstaviť ako „rúru“, do ktorej z jednej strany veci vkladáme a z druhej vyberáme. Takže to, čo sme do „rúry“ skôr vložili, z nej na druhej strane musíme aj skôr vybrať. FIFO fronta je napríklad výstupný žľab pre ťah lotérie, aký vidíme na obrázku 9.1. Frontou je tiež rad ľudí stojacich pri pokladni v obchode. Ľudia, ktorí poznajú operačné systémy UNIX-ového typu, sa už určite stretli s pojmom *pipe*, alebo po slovensky *rúra*. Pomocou rúr sa v UNIX-e dajú zreťazovať výstupy a vstupy príkazov, čo je častá prax. *Pipe* v UNIX-e je vlastne fronta (FIFO).

Obr. 9.1: Príklad fronty (FIFO)

Obr. 9.2: Príklad zásobníka (LIFO)

Zásobník – LIFO

Skratka LIFO znamená *„Last-In-First-Out“*. Čiže dáta, ktoré sme ako posledné do zásobníka vložili, z neho ako prvé vyberieme. Typický príklad LIFO zásobníka, je zásobník do pištole, aký vidíme na obrázku 9.2.

Frontu budeme v algoritme označovať písmenom F a zásobník písmenom Z. Pri oboch budeme používať dve funkcie. Prvá bude funkcia F.`vyber`(), resp. Z.`vyber`(), ktorá nám z fronty, resp. zásobníka, vyberie nasledujúci objekt. Druhá funkcia bude F.`vloz`(x), resp. Z.`vloz`(x), ktorá nám do fronty resp. zásobníka, vloží objekt x.

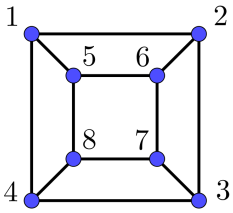
Označovanie hrán a ich vrcholov

Pri prehľadávaní grafu, či už do šírky, alebo do hĺbky, budeme pracovať s neorientovanými grafmi. Čiže hrana medzi vrcholmi u a v bude dvojprvková množina (neusporiadaná dvojica) $\{u,v\}$. Samotné hrany budeme označovať písmenami $h, h' \ldots$
Avšak pri prehľadávaní budeme konštruovať koreňový strom a jeho hranám dáme istú „orientáciu“. *Počiatočným vrcholom hrany* budeme nazývať ten jej vrchol, ktorý je bližšie ku koreňu stromu a *koncový vrchol hrany* bude ten jej vrchol, ktorý je ďalej od koreňa stromu. Z tohoto dôvodu zavedieme aj dve funkcie, ktoré v algoritmoch budeme používať. Funkcia `zaciatok`(h) nám vráti počiatočný vrchol hrany h a funkcia `koniec`(h) nám vráti koncový vrchol hrany h.

9.3 Prehľadávanie grafu do šírky

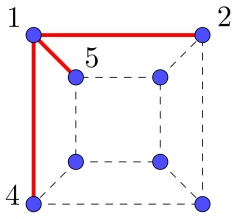
Kostra grafu je strom a pokiaľ si v strome vyberieme jeden konkrétny vrchol, tak dostávame koreňový strom. Algoritmus prehľadávania grafu do šírky, po anglicky *„Breadth-first search“* (BFS), je založený na myšlienke prehľadávania vrcholov grafu v poradí, ktoré závisí od ich vzdialenosti od koreňa stromu. Preto je tento algoritmus vhodný na nájdenie najkratšej vzdialenosti medzi zvoleným vrcholom grafu (koreňom) a ľubovoľným iným vrcholom grafu. Všetky vrcholy na jednej úrovni koreňového stromu sa spracujú skôr, než prejdeme na ďalšiu úroveň. Čiže na začiatku máme len koreň stromu, nultú úroveň. V prvom kroku prehľadávania spracujeme všetky vrcholy prvej úrovne atď. Ukážeme si to na príkladoch.

■ Príklad 9.1

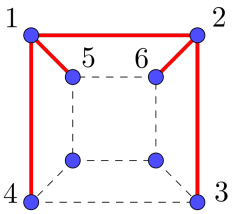


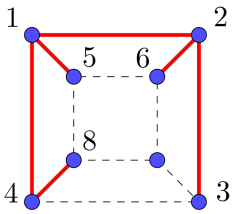
Na obrázku vľavo, je daný graf, ktorého vrcholy sú označené číslami 1 až 8. Nájdite jeho kostru prehľadávaním do šírky, pri lexikografickom usporiadaní vrcholov.

Riešenie: Začneme vrcholom 1 a v prvom kroku pridáme všetky s ním incidujúce hrany a ich druhé koncové vrcholy, pri ktorých nám nevznikne cyklus. Dostaneme strom na nasledovnom obrázku



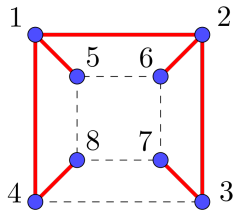
Týmto je zostrojená 1. úroveň koreňového stromu. K vrcholu 1 nám pribudli vrcholy 2, 4 a 5. Samozrejme pribudli aj hrany medzi týmito vrcholmi a koreňom stromu (vrcholom 1). V druhom kroku postupne prehľadáme vrcholy 2, 4 a 5 v uvedenom poradí a pridáme všetky s nimi incidujúce hrany a ich druhé koncové vrcholy, pri ktorých nám nevznikne cyklus. Postup je zobrazený na nasledujúcich obrázkoch





Týmto je zostrojená 2. úroveň koreňového stromu. Prehľadaním vrcholu 2 nám pribudli vrcholy 3 a 6 a dve hrany medzi nimi a vrcholom 2. Prehľadaním vrcholu 4 nám pribudol vrchol 8 a

hrana medzi ním a vrcholom 4. Prehľadaním vrcholu 5 nám nepribudlo nič ďalšie, pretože ak by sme pridali do existujúceho stromu akúkoľvek s ním incidujúcu hranu, tak by nám vznikol cyklus. V ďalšom kroku ideme prehľadať vrcholy 3 a 6. Výsledok vidíme na ďalšom obrázku



Týmto je zostrojená 3. úroveň koreňového stromu. Prehľadaním vrcholu 3 nám pribudol vrchol 7 a hrana medzi ním a vrcholom 3. V tejto chvíli už koreňový strom obsahuje všetky vrcholy pôvodného grafu, takže je jeho kostrou. ∎

Pseudokód algoritmu na prehľadávanie grafu do šírky môže vyzerať napríklad takto

---

Prehľadávanie grafu do šírky

VSTUP: Súvislý obyčajný graf $G = (V,E)$ a pevne zvolený vrchol $v_1 \in V$.

VÝSTUP: $S = (V',E')$ – kostra grafu G.

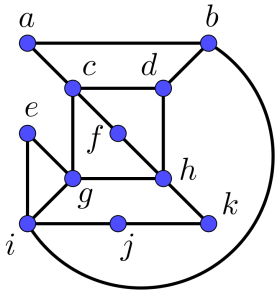
INICIALIZÁCIA

for $\{$ *pre všetky hrany* $h \in E$, *kde* $\texttt{zaciatok}(h) = v_1$ $\}$ **do**
  $F.\texttt{vloz}(h)$;
end
$E' := \{\}$;
$V' := \{v_1\}$;

VYTVÁRANIE KOSTRY S

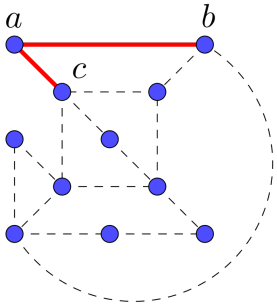
while $\{\ |V'| < |V|\ \}$ **do**
  $h := F.\texttt{vyber}()$;
  if $\{\ h \notin E'$ *a pridaním hrany* h *do* E' *nevznikne cyklus v* $S\ \}$ **then**
    $E' := E' \cup \{h\}$;
    $V' := V' \cup \{\texttt{koniec}(h)\}$;
    for $\{$ *pre všetky hrany* $h' \in E$, *kde* $\texttt{zaciatok}(h') = \texttt{koniec}(h)$ $\}$ **do**
      $F.\texttt{vloz}(h')$;
    end
  end
end

■ **Príklad 9.2**

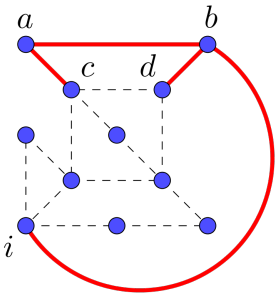


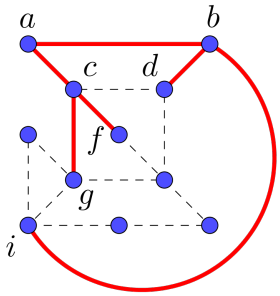
Na obrázku vľavo je daný graf, ktorého vrcholy sú označené písmenami a až k. Nájdite jeho kostru prehľadávaním do šírky, pri lexikografickom usporiadaní vrcholov.

Riešenie: Začneme vrcholom a a v prvom kroku pridáme všetky s ním incidujúce hrany a ich druhé koncové vrcholy, pri ktorých nám nevznikne cyklus. Dostaneme strom na nasledovnom obrázku

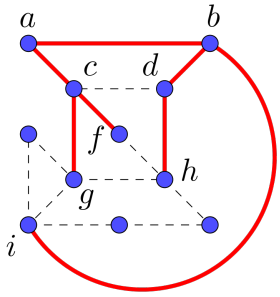


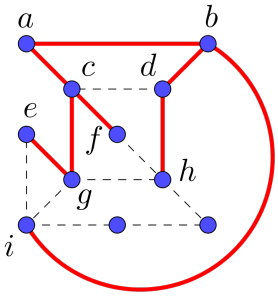
Týmto je zostrojená 1. úroveň koreňového stromu. Pribudli nám vrcholy b, c a 2 hrany medzi nimi a koreňom stromu. V druhom kroku prehľadáme vrcholy b a c v uvedenom poradí. Výsledok vidíme na ďalších obrázkoch

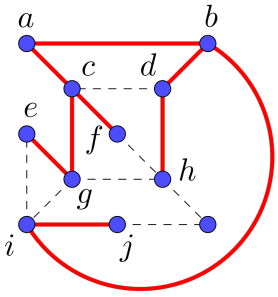




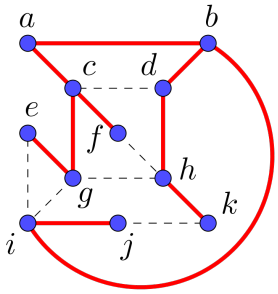
Týmto je zostrojená 2. úroveň koreňového stromu. Prehľadaním vrcholu b nám pribudli vrcholy d a i a prehľadaním vrcholu c nám pribudli vrcholy f a g. A samozrejme pribudli aj 4 zodpovedajúce hrany. V treťom kroku ideme prehľadať vrcholy d, f, g a i v uvedenom poradí. Dostávame







Týmto je zostrojená 3. úroveň koreňového stromu. Prehľadaním vrcholu d sme dostali vrchol h, prehľadaním vrcholu f nepribudol žiadny vrchol, prehľadaním vrcholu g pribudol vrchol e a prehľadaním vrcholu i pribudol vrchol j. Spolu s vrcohlmi pribudli aj príslušné hrany. V poslednom štvrtom kroku prehľadáme vrcholy e, h a j v uvedenom poradí. Dostávame

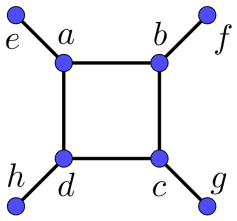


Týmto je zostrojená 4. úroveň koreňového stromu. Ako posledný pribudol vrchol k, prehľadaním vrcholu h a pribudla aj hrana medzi týmito dvoma vrcholmi. Koreňový strom už obsahuje všetky vrcholy daného grafu, takže je jeho kostrou. ■

Uvedieme ešte dva príklady prehľadávania grafu do šírky a okrem grafickej interpretácie si v nich zapíšeme postup prehľadávania grafu aj formou tabuľky.

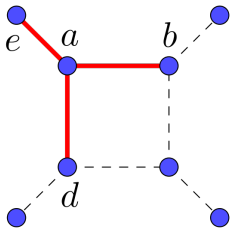
■ Príklad 9.3

Na obrázku vpravo je daný graf, ktorého vrcholy sú označené písmenami a až h. Nájdite jeho kostru prehľadávaním do šírky, pri lexikografickom usporiadaní vrcholov.



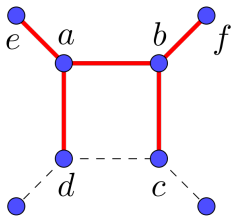
Riešenie: Tabuľkový zápis prehľadávania grafu bude mať 3 stĺpce. V prvom stĺpci tabuľky budú vrcholy, ktoré už patria do stromu S a ktorých incidujúce hrany ideme prehľadávať. V druhom stĺpci tabuľky budú hrany, ktoré sú už súčasťou stromu S, čiže tie, ktoré sú na obrázku červené. A napokon, v treťom stĺpci tabuľky budú hrany, ktoré sú momentálne vo fronte a čakajú na spracovanie. Červenou farbou budú označené tie hrany vo fronte, ktorých pridaním do S nevznikne cyklus a budú preto do S pridané v nasledujúcom kroku.

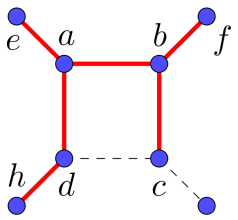
Prehľadávanie začneme vrcholom a a v prvom kroku pridáme všetky s ním incidujúce hrany a ich druhé koncové vrcholy, pri ktorých nám nevznikne cyklus. Dostaneme strom na nasledovnom obrázku



Krok	Prehľadávané vrcholy	Hrany v S	Obsah fronty
0.	a		$\{a,b\}, \{a,d\}, \{a,e\}$
1.	b,d,e	$\{a,b\},\{a,d\},\{a,e\}$	$\{b,a\},\{b,c\},\{b,f\},\{d,a\},$ $\{d,c\},\{d,h\},\{e,a\}$

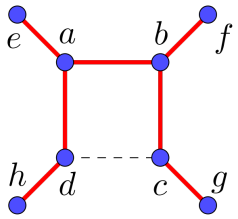
Týmto je zostrojená 1. úroveň koreňového stromu. Pribudli nám vrcholy b, d, e a hrany medzi nimi a vrcholom a. V druhom kroku prehľadaním týchto troch vrcholov dostávame





Krok	Prehľadávané vrcholy	Hrany v S	Obsah fronty
2.	c,f,h	$\{b,c\},\{b,f\},\{d,h\}$	$\{c,b\},\{c,d\},\{c,g\},\{f,b\},\{h,d\}$

Týmto je zostrojená 2. úroveň koreňového stromu. Pribudli nám vrcholy c, f, h a s nimi tri ďalšie hrany. V treťom kroku prehľadáme tieto tri vrcholy a dostaneme

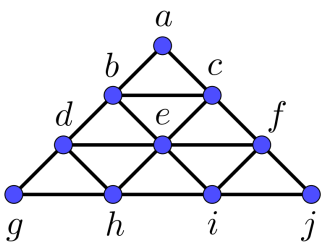


Krok	Prehľadávané vrcholy	Hrany v S	Obsah fronty
3.	g	$\{c,g\}$	

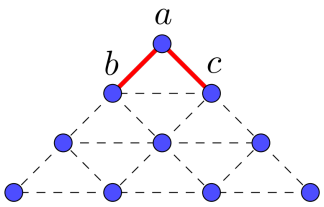
Týmto je zostrojená 3. úroveň koreňového stromu. V strome už máme všetky vrcholy pôvodného grafu, takže máme jeho kostru. ■

■ Príklad 9.4

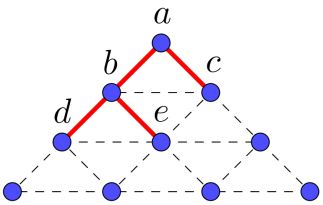
Na obrázku vpravo je daný graf, ktorého vrcholy sú označené písmenami a až j. Nájdite jeho kostru prehľadávaním do šírky pri lexikografickom usporiadaní vrcholov.

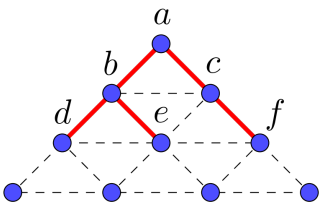


Riešenie: Úlohu najskôr vyriešime graficky a potom tabuľkou. Začneme vrcholom a a v prvom kroku pridáme všetky s ním incidujúce hrany a ich druhé koncové vrcholy, pri ktorých nám nevznikne cyklus. Dostaneme strom na nasledovnom obrázku

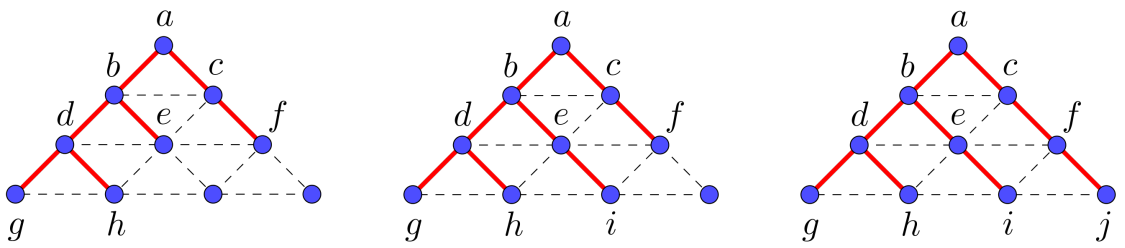


Týmto je zostrojená 1. úroveň koreňového stromu. Pribudli nám vrcholy b, c a hrany medzi nimi a vrcholom a. V druhom kroku prehľadaním týchto dvoch vrcholov dostávame





Týmto je zostrojená 2. úroveň koreňového stromu. Pribudli nám vrcholy d, e, f a s nimi tri ďalšie hrany. V treťom kroku prehľadáme tieto tri vrcholy a dostaneme



Týmto je zostrojená 3. úroveň koreňového stromu. V strome už máme všetky vrcholy pôvodného grafu, takže máme jeho kostru. Riešenie úlohy pomocou tabuľky vyzerá nasledovne.

Krok	Prehľadávané vrcholy	Hrany v S	Obsah fronty
0.	a		$\{a,b\},\{a,c\}$
1.	b,c	$\{a,b\},\{a,c\}$	$\{b,a\},\{b,c\},\{b,d\},\{b,e\},$ $\{c,a\},\{c,b\},\{c,e\},\{c,f\}$
2.	d,e,f	$\{b,d\},\{b,e\},\{c,f\}$	$\{d,b\},\{d,e\},\{d,g\},\{d,h\},$ $\{e,b\},\{e,c\},\{e,d\},\{e,f\},$ $\{e,h\},\{e,i\},\{f,c\},\{f,e\},$ $\{f,i\},\{f,j\}$
3.	g,h,i,j	$\{d,g\},\{d,h\},\{e,i\},\{f,j\}$	

■

V ďalšej časti si ukážeme iný spôsob prehľadávania grafov.

9.4 Prehľadávanie grafu do hĺbky

Prehľadávanie grafu do hĺbky, po anglicky *„Depth-first search“* (DFS), je založené na tom, že od koreňa stromu postupujeme v grafe tak hlboko, ako to je len možné. Keď sa dostaneme do vrcholu, z ktorého sa už nevieme dostať ďalej, tak sa vrátime na najbližší predchádzajúci vrchol, z ktorého sa dá postupovať ďalej inou cestou atď., až pokiaľ nie sú prehľadané všetky vrcholy daného grafu. Spomenutý „krok späť“ sa nazýva *backtracking*.

Algoritmus prehľadávania grafu do hĺbky sa dá zapísať viacerými spôsobmi. Pseudokód algoritmu prehľadávania grafu do hĺbky, ktorý je uvedený na strane 187 hore, je úmyselne napísaný tak, aby bol takmer identický s pseudokódom prehľadávania grafu do šírky (strana 182). Jediný rozdiel medzi týmito dvoma algoritmami je v použitej dátovej štruktúre. Kým pri prehľadávaní grafu do šírky sme na ukladanie hrán používali frontu, pri prehľadávaní grafu do hĺbky budeme používať zásobník.

Prehľadávanie grafu do hĺbky si ilustrujeme na rovnakých štyroch príkladoch grafov, na akých sme si ilustrovali aj prehľadávanie grafu do šírky. Všimnime si rozdiel medzi postupom prehľadávania a výslednými kostrami grafu získanými jedným a druhým spôsobom prehľadávania.

Pri prehľadávaní grafu do hĺbky pribudne, na rozdiel od prehľadávania do šírky, do konštruovanej kostry v každom kroku vždy len jedna hrana. Okrem toho, ak chceme vrcholy grafu prehľadávať v lexikografickom usporiadaní tak, ako sme to robili v predošlých príkladoch, tak musíme hrany do zásobníka vkladať v opačnom lexikografickom usporiadaní. Toto poradie vkladania hrán do fronty, alebo zásobníka, však z hľadiska prehľadávania grafu nie je nijako podstatné. Je to len kozmetická záležitosť, ktorá slúži výlučne na to, aby výsledná kostra grafu bola jednoznačná. Bez určenia poradia prehľadávania vrcholov grafu totiž môžeme, oboma prehľadávaniami, dostať viacero rôznych kostier daného grafu.

Prehľadávanie grafu do hĺbky

VSTUP: Súvislý obyčajný graf $G = (V,E)$ a pevne zvolený vrchol $v_1 \in V$.

VÝSTUP: $S = (V',E')$ – kostra grafu G.

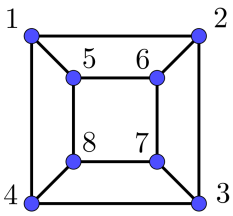
INICIALIZÁCIA

for { *pre všetky hrany* $h \in E$, *kde* `zaciatok`$(h) = v_1$ } **do**
  Z.`vloz`(h);
end
$E' := \{\}$;
$V' := \{v_1\}$;

VYTVÁRANIE KOSTRY S

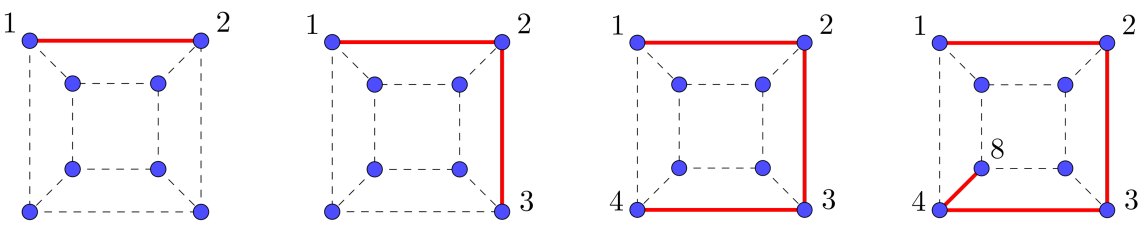
while { $|V'| < |V|$ } **do**
  $h := Z$.`vyber`$()$;
  if { $h \notin E'$ *a pridaním hrany* h *do* E' *nevznikne cyklus v* S } **then**
    $E' := E' \cup \{h\}$;
    $V' := V' \cup \{$`koniec`$(h)\}$;
    for { *pre všetky hrany* $h' \in E$, *kde* `zaciatok`(h') = `koniec`(h) } **do**
      Z.`vloz`(h');
    end
  end
end

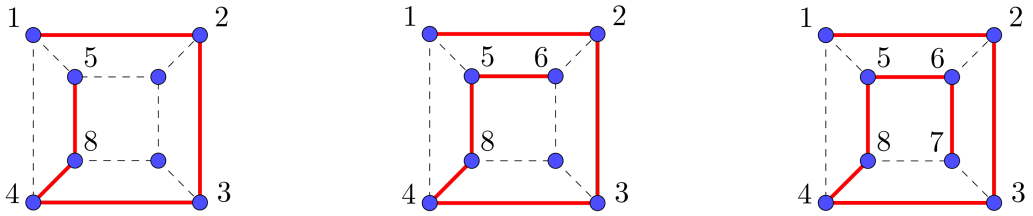
■ **Príklad 9.5**



Na obrázku vľavo je daný graf, ktorého vrcholy sú označené číslami 1 až 8. Nájdite jeho kostru prehľadávaním do hĺbky pri lexikografickom usporiadaní vrcholov.

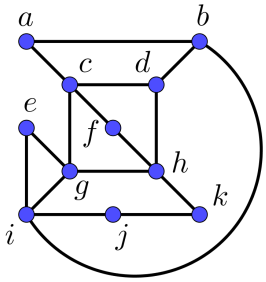
Riešenie: Začneme vrcholom 1, v prvom kroku pridáme vrchol 2, v druhom kroku pridáme vrchol 3, v treťom kroku pridáme vrchol 4, v štvrtom kroku pridáme vrchol 8, v piatom kroku pridáme vrchol 5, v šiestom kroku pridáme vrchol 6 a v poslednom, siedmom kroku, pridáme vrchol 7. V každom kroku samozrejme pridáme aj zodpovedajúcu hranu. Celý postup je zobrazený na nasledovnej sérii obrázkov





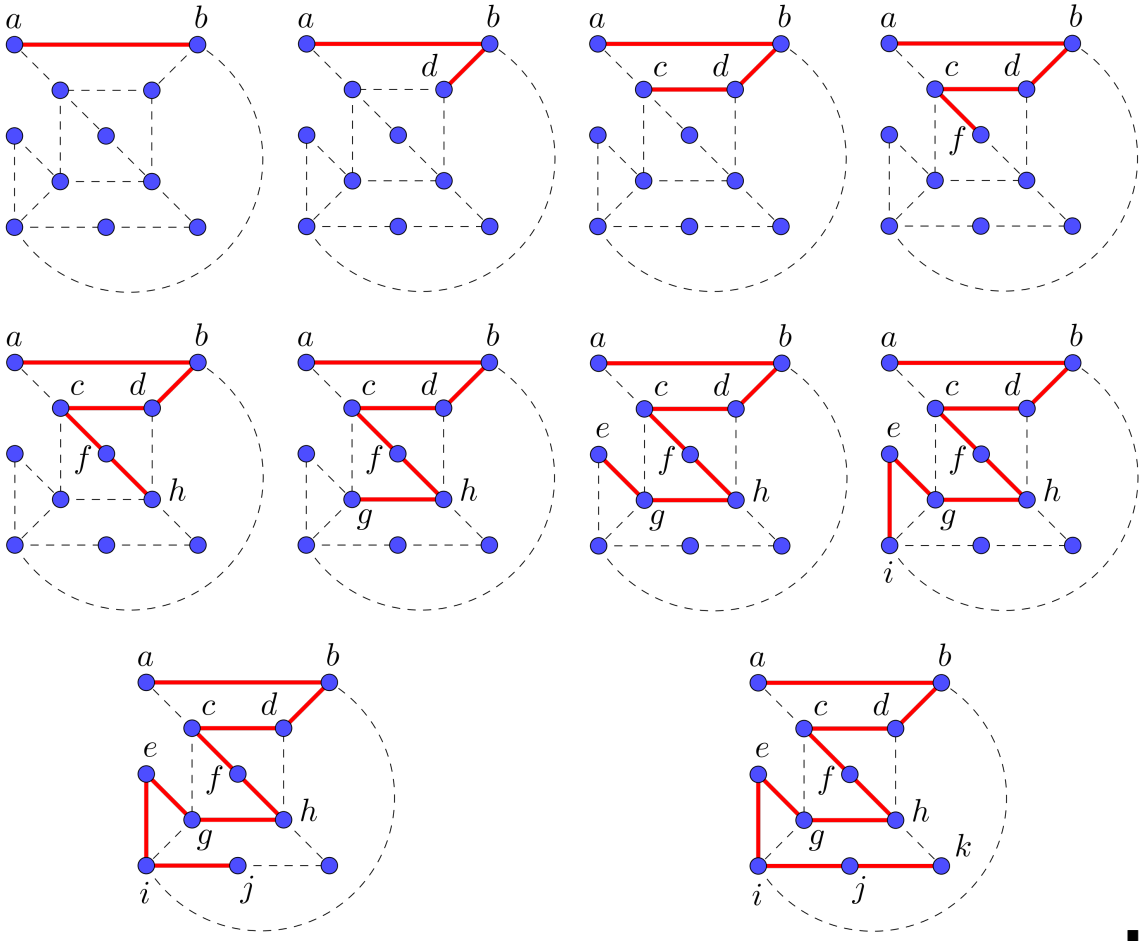
Ako vidíme z obrázkov, počas prehľadávania tohto grafu do hĺbky sme ani raz nemuseli použiť backtracking, t. j. spraviť „krok späť“. Z každého vrcholu sme vždy mohli pokračovať ďalej, až pokým sme nedostali kostru grafu, ktorá má podobu cesty dĺžky 7. ■

■ Príklad 9.6



Na obrázku vľavo je daný graf, ktorého vrcholy sú označené písmenami a až k. Nájdite jeho kostru prehľadávaním do hĺbky pri lexikografickom usporiadaní vrcholov.

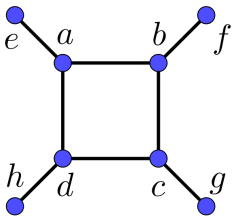
Riešenie: Celý postup riešenia je zobrazený na nasledovnej sérii desiatich obrázkov



■

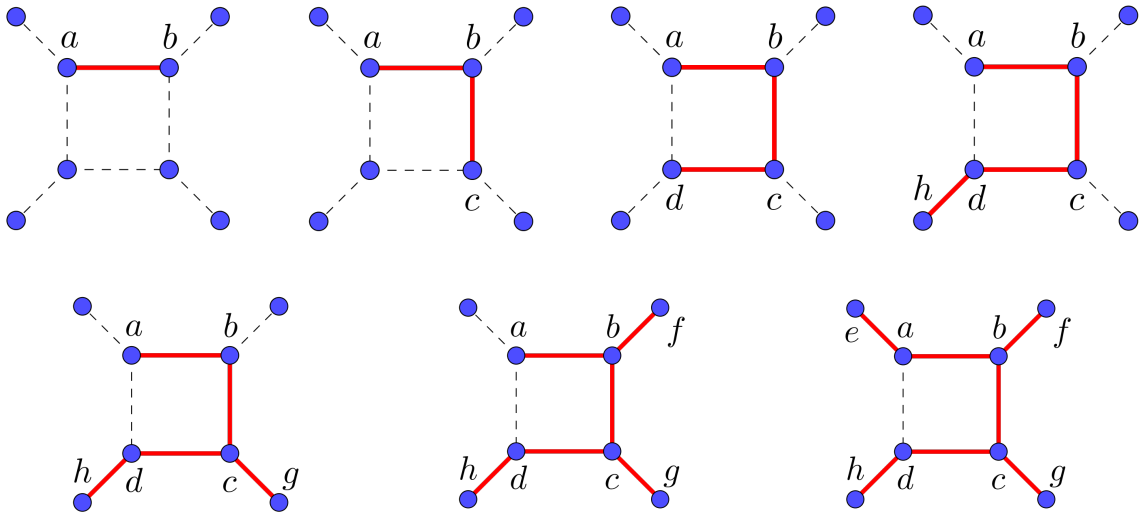
Pri ďalších dvoch príkladoch si okrem grafického riešenia zapíšeme riešenie aj pomocou tabuľky.

■ **Príklad 9.7**



Na obrázku vľavo je daný graf, ktorého vrcholy sú označené písmenami a až j. Nájdite jeho kostru prehľadávaním do hĺbky pri lexikografickom usporiadaní vrcholov.

Riešenie: prehľadávanie grafu do hĺbky znázornime najskôr graficky na sérii obrázkov



V tomto príklade sme použili aj backtracking, a to dokonca trikrát. Najskôr sme sa z vrcholu h museli vrátiť späť až do vrcholu c, potom z vrcholu g sme sa museli vrátiť do vrcholu b a nakoniec sme sa z vrcholu f museli vrátiť do vrcholu a.

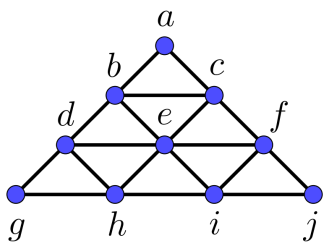
Teraz si riešenie znázornime pomocou tabuľky. Toto bude podobné tomu, ktoré sme robili pri prehľadávaní grafu do šírky, v príkladoch 9.3 a 9.4. Tabuľkový zápis prehľadávania grafu bude mať 3 stĺpce. V prvom stĺpci tabuľky budú vrcholy, ktoré už patria do stromu S a ktorých incidujúce hrany ideme prehľadávať. V druhom stĺpci tabuľky budú hrany, ktoré sú už súčasťou stromu S, čiže tie, ktoré sú na obrázku červené. A napokon, v treťom stĺpci tabuľky budú hrany, ktoré sú momentálne v zásobníku a čakajú na spracovanie. Červenou farbou bude v zásobníku vždy označená tá hrana, ktorá je najbližšia na rade, t. j. bola daná do zásobníka ako posledná.

Krok	Prehľadávaný vrchol	Hrany v S	Obsah zásobníka
0.	a		$\{a,e\},\{a,d\},\{a,b\}$
1.	b	$\{a,b\}$	$\{a,e\},\{a,d\},\{b,f\},\{b,c\},\{b,a\}$
2.	c	$\{b,c\}$	$\{a,e\},\{a,d\},\{b,f\},\{c,g\},\{c,d\},\{c,b\}$
3.	d	$\{c,d\}$	$\{a,e\},\{a,d\},\{b,f\},\{c,g\},\{d,h\},\{d,c\},\{d,a\}$
4.	h	$\{d,h\}$	$\{a,e\},\{a,d\},\{b,f\},\{c,g\},\{h,d\}$
5.	g	$\{c,g\}$	$\{a,e\},\{a,d\},\{b,f\},\{g,c\}$
6.	f	$\{b,f\}$	$\{a,e\},\{a,d\},\{f,b\}$
7.	e	$\{a,e\}$	$\{e,a\}$

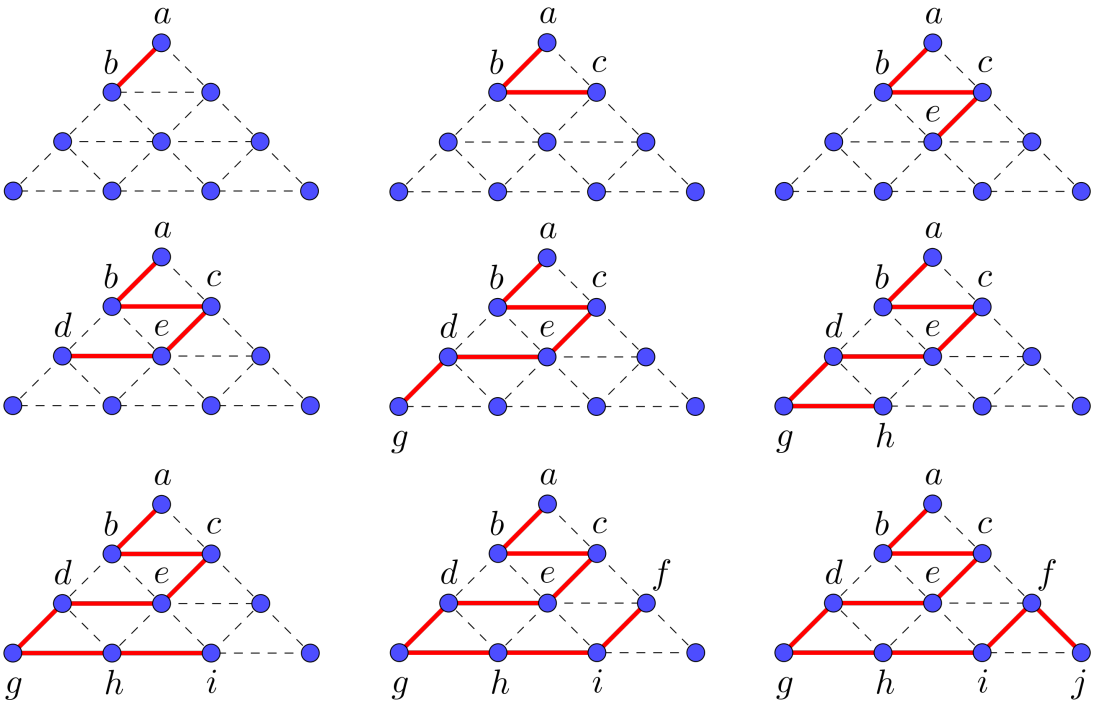
■

■ **Príklad 9.8**

Na obrázku vpravo je daný graf, ktorého vrcholy sú označené písmenami a až j. Nájdite jeho kostru prehľadávaním do hĺbky pri lexikografickom usporiadaní vrcholov.



Riešenie: prehľadávanie grafu do hĺbky si opäť znázornime graficky na sérii obrázkov



Pri tomto grafe sme ani raz nemuseli použiť backtracking a nájdená kostra je cesta dĺžky 9. Teraz si priebeh prehľadávania grafu do hĺbky zapíšeme tabuľkou.

Krok	Prehľadávaný vrchol	Hrany v S	Obsah zásobníka
0.	a		$\{a,c\},\{a,b\}$
1.	b	$\{a,b\}$	$\{a,c\},\{b,e\},\{b,d\},\{b,c\},\{b,a\}$
2.	c	$\{b,c\}$	$\{a,c\},\{b,e\},\{b,d\},\{c,f\},\{c,e\},\{c,b\},\{c,a\}$
3.	e	$\{c,e\}$	$\{a,c\},\{b,e\},\{b,d\},\{c,f\},\{e,i\},\{e,h\},\{e,f\},$ $\{e,d\},\{e,c\},\{e,b\}$
4.	d	$\{e,d\}$	$\{a,c\},\{b,e\},\{b,d\},\{c,f\},\{e,i\},\{e,h\},\{e,f\},$ $\{d,h\},\{d,g\},\{d,e\},\{d,b\}$
5.	g	$\{d,g\}$	$\{a,c\},\{b,e\},\{b,d\},\{c,f\},\{e,i\},\{e,h\},\{e,f\},$ $\{d,h\},\{g,h\},\{g,d\}$
6.	h	$\{g,h\}$	$\{a,c\},\{b,e\},\{b,d\},\{c,f\},\{e,i\},\{e,h\},\{e,f\},$ $\{d,h\},\{h,i\},\{h,g\},\{h,e\},\{h,d\}$
7.	i	$\{h,i\}$	$\{a,c\},\{b,e\},\{b,d\},\{c,f\},\{e,i\},\{e,h\},\{e,f\},$ $\{d,h\},\{i,j\},\{i,h\},\{i,f\},\{i,e\}$
8.	f	$\{i,f\}$	$\{a,c\},\{b,e\},\{b,d\},\{c,f\},\{e,i\},\{e,h\},\{e,f\},$ $\{d,h\},\{i,j\},\{i,h\},\{f,j\},\{f,i\},\{f,e\},\{f,c\}$
9.	j	$\{f,j\}$	$\{a,c\},\{b,e\},\{b,d\},\{c,f\},\{e,i\},\{e,h\},\{e,f\},$ $\{d,h\},\{i,j\},\{i,h\},\{j,i\},\{j,f\}$

Všimnime si, že algoritmus skončil a v zásobníku ešte zostali hrany. Na tomto príklade vidíme, že pri popísanom algoritme sa nemusí zásobník vždy vyprázdniť tak, ako sa to stalo pri predošlom príklade. Závisí to od daného grafu. ∎

9.5 Dôkaz korektnosti algoritmov prehľadávania do šírky a do hĺbky

Ako sme mohli vidieť v tabuľkových zápisoch algoritmov prehľadávania grafov do šírky (podkapitola 9.3) aj do hĺbky (podkapitola 9.4), nami použité algoritmy vôbec nie sú optimálne. Tejto skutočnosti sme si plne vedomí a uvedené pseudokódy algoritmov neboli písané s cieľom dosiahnuť optimálnu rýchlosť algoritmu. Naším cieľom pri písaní pseudokódov prehľadávacích algoritmov bolo napísať oba pseudokódy tak, aby

1. boli až na použitú dátovú štruktúru identické,
2. boli popisované algoritmy čo najzrozumiteľnejšie a čo najstručnejšie zapísané,
3. sa dal dôkaz ich korektnosti spraviť čo najjednoduchšie.

V nasledujúcej vete dokážeme, že algoritmy prehľadávania grafu do šírky aj do hĺbky sú korektné.

Veta 9.5.1 — O korektnosti algoritmov prehľadávania grafu do šírky a do hĺbky. Uvedené algoritmy prehľadávania grafu do šírky a do hĺbky sú korektné, čiže po ich skončení bude strom S kostrou daného grafu G.

Dôkaz: Oba algoritmy (str. 182 a 187) sú, až na použitú dátovú štruktúru (fronta/zásobník), identické. Preto dôkazy korektnosti oboch algoritmov budú rovnaké a budeme písať len o „prehľadávaní“ grafu, bez rozlišovania „do šírky“ a „do hĺbky“.

Algoritmom prehľadávania grafu postupne vzniká strom. V každom kroku pridávame k už existujúcemu stromu vrcholy a hrany tak, aby nevznikol cyklus. Okrem toho je z algoritmu zrejmé, že pre každý pridávaný vrchol existuje cesta od tohto vrcholu ku koreňu stromu. Preto medzi ľubovoľnými dvoma vrcholmi konštruovaného grafu existuje cesta. Takže konštruovaný graf je acyklický a súvislý, teda strom. Treba už len ukázať, že po skončení algoritmu strom S obsahuje všetky vrcholy daného grafu G, t. j. je jeho kostrou. To dokážeme sporom.

Algoritmus sa riadne ukončí vtedy, keď nastane situácia $|V'| = |V|$, čo je podmienka v cykle **while**. Ak je podmienka $|V'| = |V|$ splnená, tak strom S obsahuje všetky vrcholy daného grafu G, a preto je jeho kostrou. Problém môže nastať len v prípade, ak by algoritmus prehľadávania skončil predčasne. Takéto predčasné skončenie by mohlo byť spôsobené len na dvoch miestach algoritmu. Prvý možný dôvod predčasného ukončenia je ten, že fronta/zásobník sú už prázdne a zlyhá priradenie hrany $h := F.\text{vyber}()$, resp. $h := Z.\text{vyber}()$. Druhý možný dôvod predčasného ukončenia **while** cyklu je test **if**, kedy by sme už nedokázali pridať žiadnu ďalšiu hranu tak, aby nevznikol cyklus.

Takže predpokladajme, že skonštruovaný strom S nie je kostrou. Potom existuje nejaký vrchol x, ktorý nie je súčasťou vzniknutého stromu S. Daný graf G bol ale súvislý, takže v ňom musí existovať cesta medzi vrcholom x a koreňom stromu v_1. Označme si túto cestu $(x = u_1, u_2, \ldots, u_{i-1}, u_i, \ldots, u_k = v_1)$. Nech u_i je prvý vrchol na tejto ceste, ktorý je obsiahnutý aj v strome S. Je zrejmé, že $1 < i \leq k$. Keďže vrchol u_i je v strome S, musela byť v priebehu algoritmu aj hrana $\{u_{i-1}, u_i\}$ vložená do fronty/zásobníka, a teda musela byť v niektorom ďalšom kroku preskúmaná. Pridaním vrcholu u_{i-1} a hrany $\{u_{i-1}, u_i\}$ ku stromu S nemôže vzniknúť cyklus, a teda aj vrchol u_{i-1} musí byť súčasťou stromu S. To je však spor s tým, že vrchol u_i bol prvý vrchol na ceste $(x = u_1, u_2, \ldots, u_{i-1}, u_i, \ldots, u_k = v_1)$, ktorý bol súčasťou stromu S. Preto všetky vrcholy pôvodného grafu G sú súčasťou S a strom S je kostra grafu G. Q.E.D.

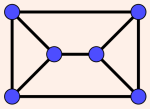
Každý, kto si beh algoritmov uvedených na stranách 182 a 187 skúsi simulovať na papieri, veľmi rýchlo príde na to, ako by sa dali uvedené algoritmy zoptimalizovať. Na túto tému sú aj cvičenia 9.10 a 9.11. Samozrejme, že pri riešení cvičení sa nemusí používať ťažkopádna forma algoritmov podľa uvedených pseudokódov a ich tabuľkový zápis prehľadávania použitý v príkladoch 9.3, 9.4, 9.7 a 9.8. My sme sa v týchto príkladoch otrocky pridržiavali v texte uvedených algoritmov. Tabuľkové zápisy oboch algoritmov sa malými modifikáciami pseudokódov dajú skrátiť, pri prehľadávaní do hĺbky dokonca výrazne skrátiť.

9.6 Cvičenia

Cvičenie 9.1 Pomocou prehľadávacích algoritmov dokážte, že všetky kostry daného grafu majú rovnaký počet hrán. ■

Cvičenie 9.2 Koľko navzájom neizomorfných kostier má graf K_6? ■

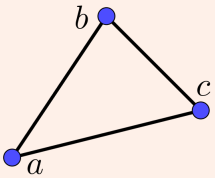
Cvičenie 9.3



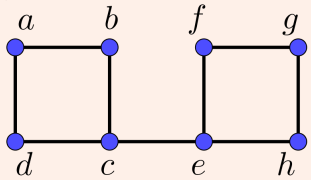
Nájdite aspoň tri rôzne kostry grafu zobrazenom na obrázku vľavo prehľadávaním do šírky a tri rôzne kostry prehľadávaním do hĺbky. ■

Cvičenie 9.4 Nájdite všetky kostry nasledujúcich grafov

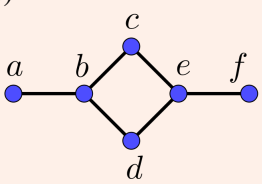
(a)



(b)



(c)



■

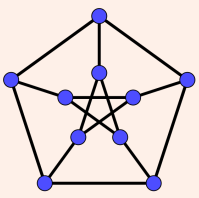
Cvičenie 9.5 Nech S je kostra grafu G. Dokážte, že ak do S pridáme ľubovoľnú hranu pri nezmenenej množine vrcholov, tak vznikne cyklus. ■

Cvičenie 9.6 Prehľadávaním do šírky aj do hĺbky nájdite kostru grafu Q_3.
Definíciu grafu Q_n nájdete v cvičení 5.21 (strana 118). ■

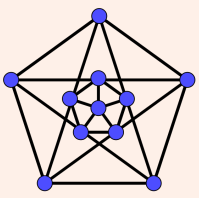
Cvičenie 9.7 Prehľadávaním do šírky aj do hĺbky nájdite kostru grafu $K_{3,7}$. ■

Cvičenie 9.8 Nájdite kostry nasledujúcich grafov, ich prehľadaním do šírky aj do hĺbky, vychádzajúc z toho istého vrcholu.

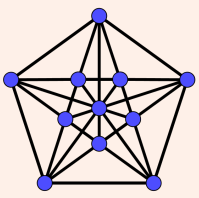
(a)



(b)



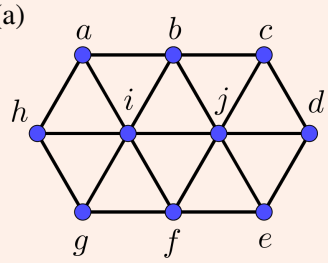
(c)



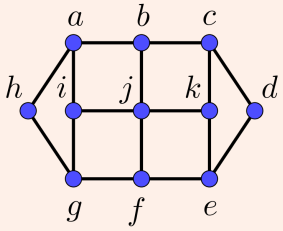
■

Cvičenie 9.9 Nájdite kostry nasledujúcich grafov, ich prehľadaním do šírky aj do hĺbky, vychádzajúc z toho istého vrcholu. V oboch prípadoch prehľadávajte vrcholy grafov v lexikografickom usporiadaní.

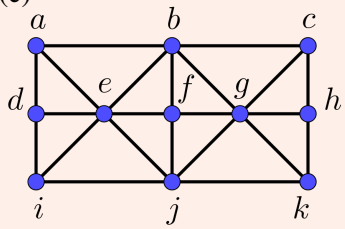
(a)



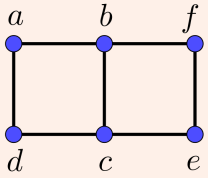
(b)



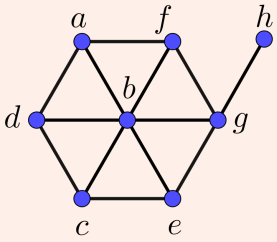
(c)



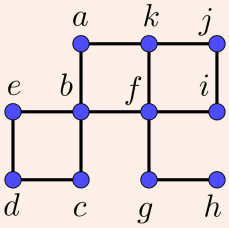
(d)



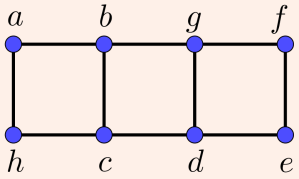
(e)



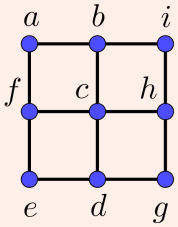
(f)



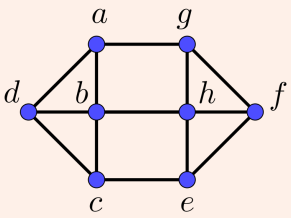
(g)



(h)



(i)



■

Cvičenie 9.10 Modifikujte algoritmus na prehľadávanie grafu do šírky (strana 182) tak, aby v ňom nebolo treba použiť podmienku **if** a jediným testom v celom algoritme zostal len test v cykle **while**. ■

Cvičenie 9.11 Dá sa modifikovať algoritmus na prehľadávanie grafu do hĺbky (strana 187) tak, aby v ňom nebolo treba použiť podmienku **if** a jediným testom v celom algoritme zostal len test v cykle **while**? ■

Cvičenie 9.12 Prečo pridaním vrcholu u_{i-1} a hrany $\{u_{i-1}, u_i\}$, v dôkaze vety 9.5.1 (str. 191), do stromu S, nemôže vzniknúť cyklus? ■

Cvičenie 9.13 Naprogramujte prehľadávanie daného grafu do šírky v Pythone alebo v C. ■

Cvičenie 9.14 Naprogramujte prehľadávanie daného grafu do hĺbky v Pythone alebo v C. ■

Cvičenie 9.15 Modifikujte algoritmus na prehľadávanie grafu do šírky (strana 182) tak, aby sa do fronty ukladali vrcholy, nie hrany. (Pomôcka: každý vrchol ukladaný do fronty dostane značku identifikujúcu vrchol, z ktorého sme sa doň dostali.) ■

Cvičenie 9.16 Modifikujte algoritmus na prehľadávanie grafu do hĺbky (strana 187) tak, aby sa do zásobníka ukladali vrcholy, nie hrany. (Pomôcka: každý vrchol ukladaný do zásobníka dostane značku identifikujúcu vrchol, z ktorého sme sa doň dostali.) ■

Cvičenie 9.17 Modifikujte algoritmy na prehľadávanie grafu do šírky aj hĺbky (strany 182 a 187) tak, aby sme zistili, či je vstupný graf súvislý (v algoritmoch uvedených v skriptách je to ako predpoklad). ■

Cvičenie 9.18 Modifikujte algoritmus na prehľadávanie grafu do šírky (strana 182) tak, aby sme pomocou neho zistili vzdialenosť každého vrcholu v danom grafe od vrcholu v_1, z ktorého sa začína prehľadávanie. ■

Cvičenie 9.19 Ukážte, že pri modifikovanom algoritme prehľadávania do hĺbky z cvičenia 9.16, sa v zásobníku nachádzajú len vrcholy tvoriace cestu medzi práve vyšetrovaným vrcholom po vrchol v_1, z ktorého sa začína prehľadávanie. ■

Cvičenie 9.20 Ukážte, že každá hrana grafu G spája iba vrcholy, ktoré sú v strome získanom prehľadávaním do hĺbky vo vzťahu predok–potomok. ■

Použitá a odporúčaná literatúra

[1] Martin Aigner: **Combinatorial Theory**, SPRINGER VERLAG, HEIDELBERG 1979.

[2] Bohuslav Balcar, Petr Štěpánek: **Teorie množin**, ACADEMIA, PRAHA 1986.

[3] Berežný, Draženská, Kravecová: **Zbierka úloh z diskrétnej matematiky**, FEI TU, KOŠICE 2005. `http://web.tuke.sk/fei-km/sites/default/files/prilohy/10/Zbierka-DM.pdf`

[4] J. A. Bondy, U. S. Murty: **Graph Theory With Applications**, NORTH-HOLLAND 1976.

[5] Reinhard Diestel: **Graph Theory**, SPRINGER 2000.

[6] Harary, Frank: **Graph theory**, ADDISON-WESLEY 1969.

[7] Richard Johnsonbaugh: **Discrete Mathematics**, PEARSON 2017.

[8] Josef Kaucký: **Kombinatorické identity**, VEDA, VYDAVATEĽSTVO SAV, BRATISLAVA 1975.

[9] Martin Knor: **Úvod do matematickej logiky**, FIIT STU, BRATISLAVA 2016. `http://www.math.sk/jmkollar/literatura/Knor-Uvod_do_matematickej_logiky.pdf`

[10] Martin Knor: **Teória grafov**, SvF STU, BRATISLAVA 2008. `http://www.svf.stuba.sk/docs/dokumenty/skripta/teoria_grafov_martin_knor.pdf`

[11] Martin Knor, Jozef Kollár: **Matematika pre architektov**, STU BRATISLAVA, 2002.

[12] Donald E. Knuth: **The Art of Computer Programming, Volume 1**, ADDISON-WESLEY, 1997.

[13] Sergei K. Lando: **Lectures on Generating Functions**, AMERICAN MATHEMATICAL SOCIETY, STUDENT MATHEMATICAL LIBRARY 2003.

[14] Milan Mareš: **Příběhy matematiky**, PISTORIUS & OLŠANSKÁ, PŘÍBRAM 2011.

[15] J. Matoušek, J. Nešetřil: **Kapitoly z diskrétní matematiky**, KAROLINUM, PRAHA 2002.

[16] М. В. Меньшиков, А. М. Ревякин, А. Н. Копылова, Ю. Н. Макаров, Б. С. Стечкин: **Комбинаторный Анализ – Задачи и упражнения**, Издательство »НАУКА«, Москва 1982.

[17] В. А. Носов: **Комбинаторика и теория графов**, Москва 1999. `http://intsys.msu.ru/staff/vnosov/combgraph.htm` (PDF verzia)

[18] Stanislav Palúch: **Algoritmická teória grafov**, Žilinská univerzita, 2008. `http://frcatel.fri.uniza.sk/users/paluch/grafy.pdf`

[19] Ján Plesník: **Grafové algoritmy**, VEDA, Bratislava 1983.

[20] Franco P. Preparata, Raymond T. Yeh: **Úvod do téorie diskrétnych matematických štruktúr**, Alfa, Bratislava 1982.

[21] Edward M. Reingold, Jurg Nievergelt, Narsingh Deo: **Combinatorial Algorithms – Theory and Practice**, Prentice-Hall, New Jersey 1977.

[22] Beloslav Riečan a kol.: **Úlohy z matematiky pre 4. ročník gymnázia**, SPN, Bratislava 1976.

[23] Beloslav Riečan a kol.: **Matematika pre 4. ročník gymnázia**, SPN, Bratislava 1987.

[24] Beloslav Riečan a kol.: **Zbierka úloh z matematiky pre 4. ročník gymnázia**, SPN, Bratislava 1991.

[25] Karol Rovan a kol.: **Zbierka riešených úloh z algebry pre SVŠ a odborné školy**, SPN, Bratislava 1969.

[26] К. А. Рыбников: **Введение в Комбинаторный Анализ**, Издательство Московского Университета, Москва 1985.

[27] Raymond M. Smullyan: **A Beginner's Guide to Mathematical Logic**, Dover Publications, New York 2014.

[28] František Vejsada, František Talafous: **Zbierka úloh z matematiky pre SVŠ a gymnázia**, SPN, Bratislava 1973.

[29] Naum J. Vilenkin: **Rozhovory o množinách**, SPN, Bratislava 1972.

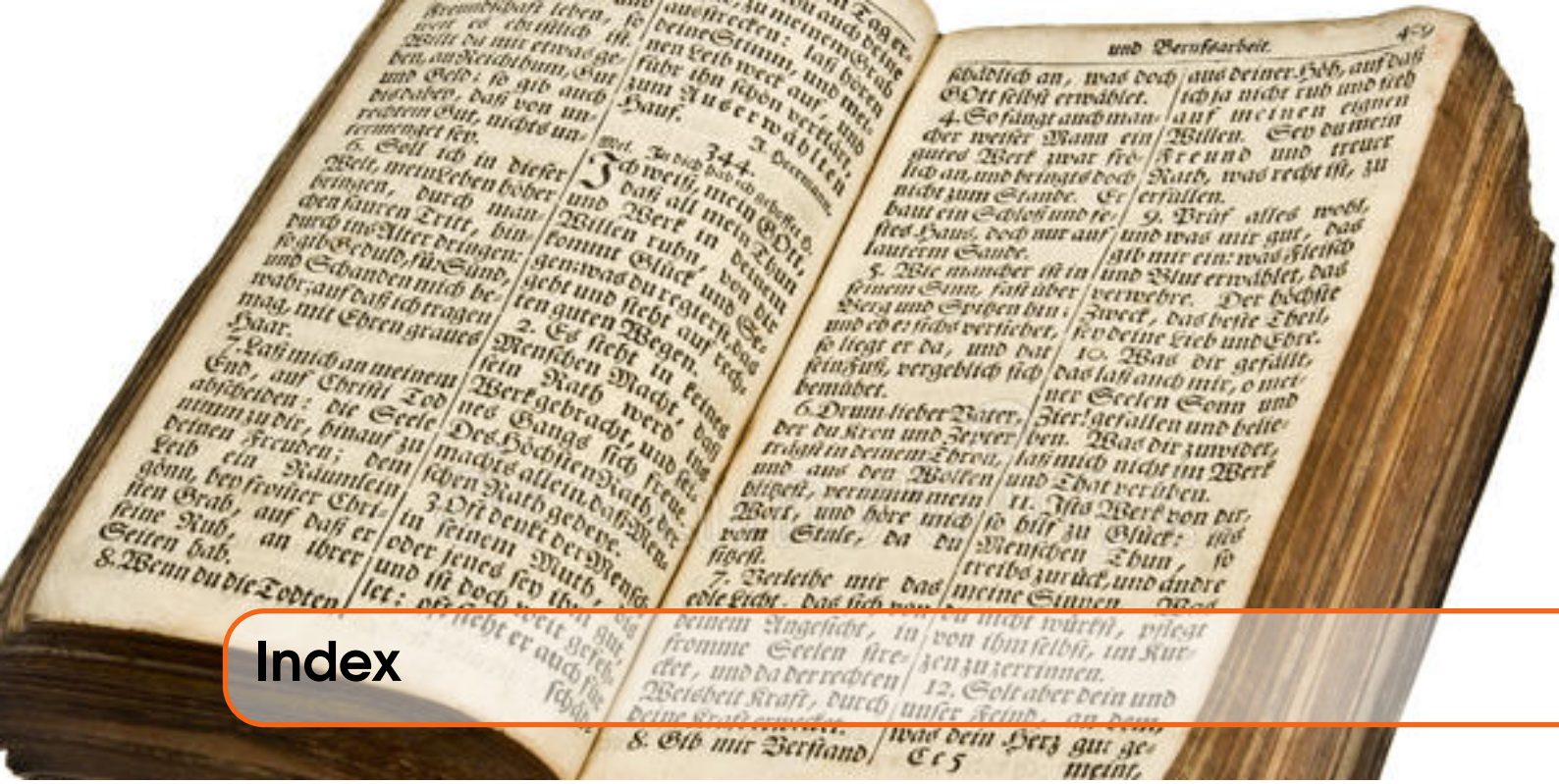
[30] Herbert S. Wilf: **generatingfunctionology**, Academic Press, Inc. 1994. `https://www.math.upenn.edu/~wilf/DownldGF.html`

[31] Niklaus Wirth: **Algoritmy a štruktúry údajov**, Alfa, Bratislava 1989.

[32] Štefan Znám: **Kombinatorika a teória grafov**, PF UK, Bratislava 1978.

IV

Index



Index

Symboly

A

B

C

Č

D

E

F

G

H

I

J

K

L

M

N

P

R

S

T

Ť

U

Ú

V

W

Z