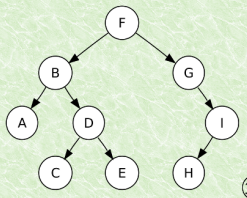
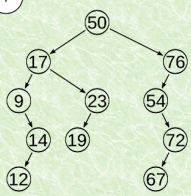
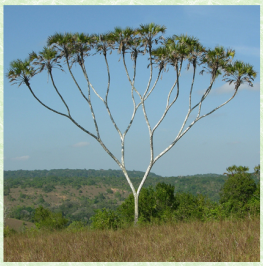
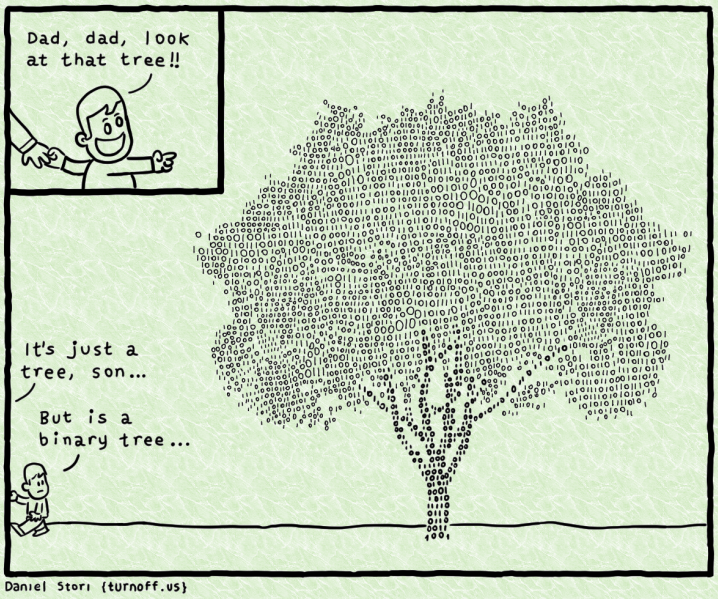








11. Binárne stromy, izomorfizmus stromov, hra NIM

11.1 Prehľadávacie binárne stromy

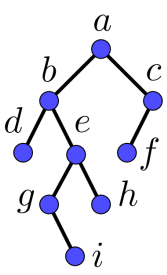
Binárne stromy boli definované a ich vlastnosti boli opísané v podkapitole 4.4. Bolo tam spomenuté, že binárne stromy sú vhodné na takú reprezentáciu a usporiadania množín dát, aby sa dali efektívne triediť a prehľadávať. Príslušná štruktúra sa nazýva *prehľadávací binárny strom*.

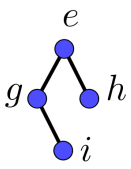
Z hľadiska organizácie dát pomocou binárnych stromov je dôležitá otázka, koľko vlastne existuje rôznych binárnych stromov s n vrcholmi. Ak si tento počet označíme pomocou b_n, tak sa pomocou vytvárajúcich funkcií (kapitola 8) dá odvodiť vzorec $b_n = \frac{1}{n+1}\binom{2n}{n}$. Takto definované čísla b_n sa nazývajú *Catalanove čísla* a uvedený vzorec bol odvodený v časti 8.3.4 (str. 171). Okrem toho ho prípadní záujemci môžu nájsť aj v knihe [15], v časti 10.4 (Binární stromy – český názov).

Už v definícii 4.4.1 sme intuitívne použili pojmy *ľavý podstrom* a *pravý podstrom*. Teraz si uvedieme aj formálnu definíciu týchto pojmov.

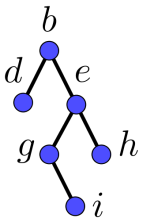
Definícia 11.1.1 — Ľavý a pravý podstrom vrcholu. Nech u je vnútorný vrchol binárneho stromu B. Potom ľavý (pravý) podstrom vrcholu u je podstrom stromu B, s koreňom v ľavom (pravom) synovi vrcholu u.

■ **Príklad 11.1** Vezmime si ako príklad strom na obrázku vľavo. Ľavý podstrom vrcholu b, je len vrchol d. Pravý podstrom vrcholu b je strom na obrázku 11.1. Ľavý podstrom vrcholu a je na obrázku 11.2 a pravý podstrom vrcholu a je na obrázku 11.3.





Obr. 11.1



Obr. 11.2

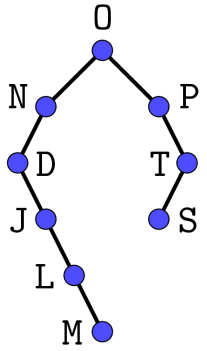
Obr. 11.3

■

Ďalej si formálne definujeme pojem *prehľadávací binárny strom*. Tento sa používa na organizáciu dát, napríklad na uchovávanie prvkov nejakej usporiadanej množiny. Pritom usporiadané množiny môžu byť množiny čísel, alebo množiny ľubovoľných reťazcov usporiadaných lexikograficky.

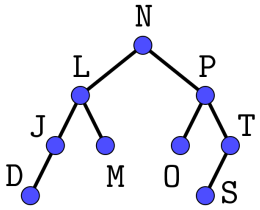
Definícia 11.1.2 — Prehľadávací binárny strom. Nech A je konečná usporiadaná množina, B je binárny strom a u jeho ľubovoľný vrchol. Potom B je prehľadávací binárny strom, ak každému jeho vrcholu je priradený práve jeden prvok množiny A tak, že prvok priradený vrcholu u je väčší, než prvok priradený ľubovoľnému vrcholu jeho ľavého podstromu a menší, než prvok priradený ľubovoľnému vrcholu jeho pravého podstromu.

■ **Príklad 11.2**



Množinu $A = \{\texttt{O}, \texttt{P}, \texttt{N}, \texttt{D}, \texttt{T}, \texttt{J}, \texttt{L}, \texttt{S}, \texttt{M}\}$, pri abecednom usporiadaní, môžeme napríklad umiestniť do prehľadávacieho binárneho stromu, ktorý máme zobrazený na obrázku vľavo. Strom, ktorý je na obrázku vľavo, dostaneme, ak doň prvky množiny A umiestňujeme v poradí uvedenom vyššie. Toto je niekedy potrebné, avšak z hľadiska prehľadávania a rýchleho prístupu nie ideálne. Pre rýchly prístup k uloženým dátam potrebujeme, aby mal prehľadávací binárny strom čo najmenšiu výšku.

Ak by sme prvky množiny A neukladali do prehľadávacieho binárneho stromu v poradí uvedenom vyššie, tak by sme mohli dostať napríklad prehľadávací binárny strom, ktorý vidíme na obrázku vpravo. Výška tohto stromu je len 3, čo je o 2 menej, než výška predošlého stromu. Pritom skutočne ide o prehľadávací binárny strom podľa definície 11.1.2, o čom sa ľahko môžeme presvedčiť.



■

Teraz si opíšeme postup, ktorým sme zostrojili prvý prehľadávací binárny strom z príkladu 11.2.

- ◇ Prvý prvok množiny A, čiže O, bude koreňom stromu.
- ◇ Ďalší prvok množiny A je P. V abecede je $\texttt{O} < \texttt{P}$, preto P musí ležať v pravom podstrome. Vrchol O zatiaľ nemá pravého syna, takže P bude je pravý syn.
- ◇ Ďalší prvok množiny A je N. V abecede je $\texttt{N} < \texttt{O}$, preto N musí ležať v ľavom podstrome. Keďže vrchol O zatiaľ nemá ľavého syna, bude N jeho ľavý syn.
- ◇ Ďalší prvok množiny A je D. V abecede je $\texttt{D} < \texttt{O}$, preto D musí ležať v ľavom podstrome. Ľavý podstrom vrcholu O je zatiaľ len vrchol N a $\texttt{D} < \texttt{N}$, takže D bude ľavý syn vrcholu N.
- ◇ Atď...

Tento postup si teraz zapíšeme ako pseudokód algoritmu. V popise pseudokódu budeme používať rovnakú funkciu $\texttt{prvy}(P)$, akú sme použili aj v pseudokóde Kruskalovho algoritmu (časť 10.2.2, strana 201). Táto funkcia nám z postupnosti P vráti jej prvý prvok a zároveň tento prvok z P odstráni. Ďalej budeme používať funkcie $\texttt{lavy.syn}(v)$, resp. $\texttt{pravy.syn}(v)$, ktoré nám vrátia ľavého, resp. pravého syna vrcholu v. Keďže sa jedná o binárny strom, tak syn vrcholu môže byť buď ľavý alebo pravý. Preto ak pridávame do stromu ďalšiu hranu s koncovými vrcholmi u a v, budeme rozlišovať medzi ľavou hranou $\texttt{L}(v,u)$ a pravou hranou $\texttt{P}(v,u)$. Okrem toho, v zápise stotožníme vrcholy grafu a prvky, ktoré zodpovedajú vrcholom. Takže ak u a v budú dva rôzne vrcholy stromu B, tak buď platí $u < v$, alebo platí $v < u$, podľa toho, v akom vzájomnom vzťahu sú prvky usporiadanej množiny A, priradené týmto vrcholom. Napokon je v pseudokóde použitá

funkcia `STOP` (**while**), ktorá ukončí cyklus while, v ktorom sa nachádza. Pseudokód algoritmu na konštrukciu prehľadávacieho binárneho stromu sa potom dá zapísať nasledovným spôsobom.

Konštrukcia prehľadávacieho binárneho stromu

VSTUP: Postupnosť P neopakujúcich sa prvkov usporiadanej množiny A.

VÝSTUP: Prehľadávací binárny strom B pre postupnosť P.

INICIALIZÁCIA

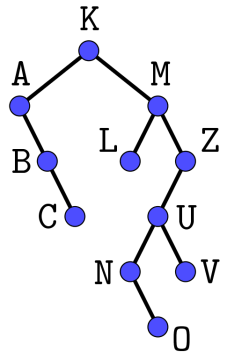
$r := \texttt{prvy}(P)$; (koreň prehľadávacieho stromu)
$V := \{r\}$;
$E := \{\}$;

VYTVÁRANIE PREHĽADÁVACIEHO BINÁRNEHO STROMU B

```
while { postupnosť P je neprázdna } do
    u := prvy(P);
    v := r;
    while { 1 = 1 } do
        if { u < v } then
            F := L;          (návestie určujúce smer pridávanej hrany)
            if { existuje ľavý syn vrcholu v } then
                v := lavy.syn(v);
            else
                STOP (while);
            end
        else
            F := P;          (návestie určujúce smer pridávanej hrany)
            if { existuje pravý syn vrcholu v } then
                v := pravy.syn(v);
            else
                STOP (while);
            end
        end
    end
    V := V ∪ {u};
    E := E ∪ {F(v,u)};
end
```

Teraz si ukážeme dva príklady koštrukcie prehľadávacích binárnych stromov pomocou uvedeného algoritmu.

■ **Príklad 11.3**

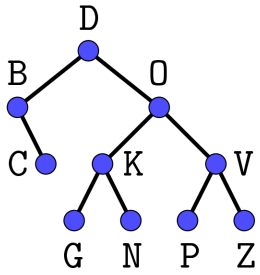


Ak pre množinu $A = \{\texttt{K}, \texttt{M}, \texttt{A}, \texttt{Z}, \texttt{U}, \texttt{B}, \texttt{C}, \texttt{N}, \texttt{V}, \texttt{O}, \texttt{L}\}$ pomocou uvedeného algoritmu zostrojíme prehľadávací binárny strom, pričom prvky množiny A budeme brať v uvedenom poradí, tak dostaneme graf, ktorý vidíme na obrázku vľavo.

■

■ **Príklad 11.4**

Ak pre množinu $A = \{\texttt{D}, \texttt{O}, \texttt{V}, \texttt{K}, \texttt{G}, \texttt{N}, \texttt{P}, \texttt{Z}, \texttt{B}, \texttt{C}\}$ pomocou uvedeného algoritmu zostrojíme prehľadávací binárny strom, pričom prvky množiny A budeme brať v uvedenom poradí, tak dostaneme graf, ktorý vidíme na obrázku vpravo.



■

Prehľadávacie binárne stromy sú užitočné pri lokalizácii dát, čiže pri overovaní, či sa nejaká položka nachádza v danom súbore dát, a ak áno, na ktorom mieste. Zoberme si napríklad prehľadávací binárny strom z príkladu 11.4. Chceli by sme vedieť, či sa v množine A nachádza aj písmeno N. Postup overovania by bol nasledovný.

Začneme od koreňa stromu, ktorý obsahuje písmeno D. Vieme, že $\texttt{N} > \texttt{D}$, takže ak má vrchol D pravého syna, prejdeme naň. Je to vrchol O. Platí $\texttt{N} < \texttt{O}$, takže ak má vrchol O ľavého syna, tak naň prejdeme. To je vrchol K. Platí $\texttt{N} > \texttt{K}$, takže ak má vrchol K pravého syna, prejdeme naň. Dostali sme sa na vrchol N a hľadanie skončilo.

Ako by prebiehalo hľadanie, ak by sme chceli zistiť, či sa v množine A nachádza písmeno R? Znovu začneme koreňom, vrcholom D. Vieme, že $\texttt{R} > \texttt{D}$, takže ak má vrchol D pravého syna, prejdeme naň. Je to vrchol O. Platí $\texttt{R} > \texttt{O}$, takže ak má vrchol O pravého syna, tak naň prejdeme. To je vrchol V. Platí $\texttt{R} < \texttt{V}$, takže ak má vrchol V ľavého syna, prejdeme naň. Je ním vrchol P a platí $\texttt{R} > \texttt{P}$, takže ak by mal vrchol P pravého syna, prešli by sme naň. Avšak vrchol P nemá pravého syna, takže hľadanie skončilo a vieme, že písmeno R sa v množine A nevyskytuje.

V praxi nás bude zaujímať, koľko krokov v binárnom strome musíme spraviť, aby sme dostali odpoveď na otázku, či sa nejaký prvok v súbore dát nachádza, alebo nenachádza. Potrebný počet krokov bude maximálny vtedy, ak musíme v binárnom strome prejsť až na koniec jeho najhlbšej vetvy. V strome z príkladu 11.4 by to boli otázky na písmena G, N, P a Z, ktoré sa v množine A nachádzajú, alebo otázky na písmená, ktoré sa v množine A nenachádzajú a pri ich hľadaní musíme prejsť, cez uvedené 4 písmená (napríklad písmeno R).

V aplikáciach bude maximálny čas potrebný na lokalizáciu dátovej položky závisieť od výšky prehľadávacieho binárneho stromu. Pri každom kroku postúpime o 1 úroveň ďalej od koreňa stromu. To znamená, že čím nižší je prehľadávací strom, tým skôr vieme lokalizovať dátovú položku. V praxi sa preto na účely organizácie a lokalizácie dát používajú vyvážené prehľadávacie binárne stromy, pretože v ich prípade je čas potrebný na lokalizáciu dát minimálny. Algoritmus konštrukcie prehľadávacieho binárneho stromu, uvedený vyššie, vo všeobecnosti nedáva optimálne prehľadávacie stromy. Existuje ale viacero spôsobov ako výšku prehľadávacieho stromu minimalizovať a určite sa s nimi študenti aplikovanej informatiky stretnú na odborných informatických predmetoch. Poďme sa však teraz venovať odhadu, koľko najviac krokov je potrebných na lokalizáciu dát v prehľadávacom binárnom strome s n vrcholmi.

Majme prehľadávací binárny strom B, ktorého výška je h. Ako už bolo spomenuté, v každom kroku prehľadávania, t. j. pri každom porovnaní hľadanej dátovej položky s aktuálnym vrcholom stromu, postúpime o 1 úroveň ďalej od koreňa. Takže maximálny počet krokov, ktoré môžeme spraviť v binárnom strome je $(h+1)$, čiže výška stromu plus 1. Koreň stromu je úroveň 0, preto ak má koreňový strom výšku h, tak má $(h+1)$ úrovni.

Ľubovoľný prehľadávací binárny strom B s n vrcholmi vieme doplniť na úplný binárny strom B^* tým, že pridáme hrany a vrcholy ku jeho pôvodným vrcholom tak, aby všetky jeho pôvodné vrcholy mali 2 synov. Takto vznikne úplný binárny strom B^*, ktorého vnútorné vrcholy sú práve všetky vrcholy pôvodného binárneho stromu B. Takže strom B^* má n vnútorných vrcholov. Podľa vety 4.4.1 vieme, že úplný binárny strom s n vnútornými vrcholmi má $(n+1)$ koncových vrcholov. Takže strom B^* bude mať $(n+1)$ koncových vrcholov a jeho výška je $(h+1)$. Podľa vety 4.4.2 potom vieme, že platí nerovnosť $\log_2(n+1) \leq (h+1)$. Platí preto nasledujúca veta.

Veta 11.1.1 — O optimálnej lokalizácii v prehľadávacom binárnom strome. Pre počet krokov k, potrebných na lokalizáciu dátovej položky v prehľadávacom binárnom strome s n vrcholmi, platí

$$k \geq \log_2(n+1)\,.$$

Dôkaz: vyplýva z viet 4.4.1, 4.4.2 a vyššieuvedeného.

Dá sa ukázať, že v prípade minimalizovanej výšky prehľadávacieho binárneho stromu, pri vyvážených binárnych stromoch, platí

$$k = \lceil \log_2(n+1) \rceil\,,$$

kde $\lceil a \rceil$ označuje hornú celú časť reálneho čísla a. Ak napríklad potrebujeme zorganizovať $n = 2\,000\,000$ položiek, tak ich vieme uložiť do prehľadávacieho binárneho stromu tak, aby sme ľubovoľnú položku vedeli lokalizovať na maximálne

$$k = \lceil \log_2 2\,000\,000 \rceil = \lceil 20.93\ldots \rceil = 21$$

krokov, t. j. stačí nám na to prehľadávací binárny strom s výškou 20.

11.2 Izomorfizmus stromov

V podkapitole 5.2 (strana 98) bol definovaný izomorfizmus grafov vo všeobecnosti. Pripomeňme, že podľa definície 5.2.1, sú dva grafy $G = (V,E)$ a $G' = (V',E')$ izomorfné, ak existuje bijekcia $\varphi : V \to V'$, ktorá zachováva susednosť vrcholov a násobnosť hrán.

Stromy sú grafy, t. j. izomorfizmus stromov sa definuje rovnako ako izomorfizmus grafov. V častiach 5.2.2 (strana 105) a 5.2.3 (strana 106) bol definovaný izomorfizmus koreňových a binárnych stromov. Tieto kategórie stromov majú niektoré vlastnosti naviac[1], a preto aj v definícii izomorfizmu týchto kategórií stromov potrebujeme doplňujúce podmienky.

V podkapitole o izomorfizme bolo spomenuté, že zistiť, či dané dva grafy sú, alebo nie sú, izomorfné, je vo všeobecnosti ťažký problém. Sú však kategórie grafov, pre ktoré problém zistenia ich izomorfizmu je ľahký. Napríklad stromy sú takouto kategóriou grafov a v nasledujúcich častiach budú opísané algoritmy overovania izomorfizmu dvoch binárnych stromov, dvoch koreňových stromov a dvoch stromov všeobecne. Časová zložitosť overovania izomorfizmu dvoch stromov je pre všetky typy stromov lineárna, resp. takmer lineárna.

[1] Koreňové stromy majú oproti stromom naviac koreň a binárne stromy majú oproti koreňovým stromom naviac orientáciu hrán a to, že každý vrchol môže mať najviac 2 synov.

11.2.1 Algoritmus overovania izomorfizmu binárnych stromov

Algoritmus overovania izomorfizmu dvoch binárnych stromov bude spočívať v tom, že každému binárnemu stromu jednoznačne priradíme kód. O tomto kóde ukážeme, že jednoznačne určuje príslušný binárny strom a naviac, že dva izomorfné binárne stromy budú mať rovnaký kód.

Vrcholy binárneho stromu budeme označovať písmenami $v_1, \ldots, v_i, \ldots$ Každému vrcholu budeme prideľovať značku, ktorú pre vrchol v_i budeme označovať $Z(v_i)$. Keďže ide o binárny strom, v ktorom každý vrchol môže mať ľavého a pravého syna, budeme pre vrchol v_i označovať jeho ľavého syna v_{iL} a jeho pravého syna v_{iP}.

Pseudokód algoritmu vytvárania kódu binárneho stromu, máme na strane 219 hore. Vytváranie kódu sa začína tým, že každému koncovému vrcholu dáme značku 01. Potom postupujeme od koncových vrcholov ku koreňu a každému vnútornému vrcholu v_i budeme dávať značku zostavenú zo značiek jeho synov nasledovne

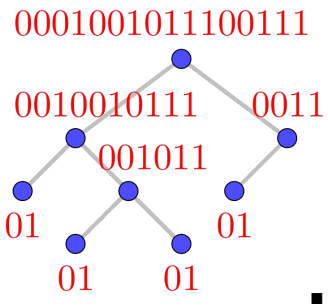
- ▷ Ak vrchol v_i má ľavého aj pravého syna, tak jeho značka bude $0 + Z(v_{iL}) + Z(v_{iP}) + 1$.
- ▷ Ak vrchol v_i má len ľavého syna, tak jeho značka bude $0 + Z(v_{iL}) + 1$.
- ▷ Ak vrchol v_i má len pravého syna, tak jeho značka bude $0 + \mathtt{X} + Z(v_{iP}) + 1$.

V uvedených zápisoch znamená symbol $+$ „zreťazenie“ dvoch značiek. Takže ak napríklad vrchol v_i má oboch synov a jeho synovia sú koncové vrcholy so značkami 01 a 01, tak značka vrcholu v_i bude $Z(v_i) = 001011$. Podrobnejšie si postup ukážeme na nasledujúcich príkladoch.

Ako vidíme z uvedeného algoritmu, značka každého vrcholu bude pozostávať zo znakov $\{0, 1, \mathtt{X}\}$. Ďalej si všimnime, že značky vrcholov sú zostavované tak, že je dodržaná parita znakov 0 a 1. To znamená, že v značke každého vrcholu je vždy rovnako veľa núl ako jedničiek. Napokon si ešte všimnime, že dĺžka značiek narastá smerom od koncových vrcholov ku koreňu stromu. Keď všetkým vrcholom binárneho stromu opísaným spôsobom pridelíme značky, tak značku koreňa zoberieme ako kód binárneho stromu. Teraz to ilustrujeme na sľúbených príkladoch. Najskôr ukážeme zostavovanie kódu daného binárneho stromu, potom dokážeme, že takto vygenerovaný kód je jednoznačný, že izomorfné stromy majú rovnaký kód a nakoniec ukážeme ako z kódu binárneho stromu môžeme nakresliť pôvodný binárny strom.

■ Príklad 11.5

Na obrázku vpravo máme binárny strom. Pri zostavovaní jeho kódu začneme tým, že najskôr všetkým jeho koncovým vrcholom dáme značky 01. Potom všetkým vnútorným vrcholom dávame značky podľa ich detí. Značky jednotlivých vrcholoch máme na obrázku a výsledný kód binárneho stromu z obrázku, je 0001001011100111.



■

V predošlom príklade mali všetky vnútorné vrcholy daného stromu ľavých synov, a preto výsledný kód pozostáva len z núl a jedničiek. Symbol `X` v kóde potrebujeme preto, lebo pri binárnych stromoch rozlišujeme medzi ľavým a pravým synom vrcholu. Symbol `X` v kóde označuje chýbajúceho ľavého syna vrcholu. Takže tým vlastne daný binárny strom fiktívne „doplníme“ na úplný binárny strom. Tam, kde pri vnútornom vrchole chýba ľavý syn, „doplníme“ namiesto neho `X`. Na ďalšom príklade predvedieme vytváranie kódu pre binárny strom, v ktorom chýbajú ľaví synovia niektorých vnútorných vrcholov.

Vytváranie identifikačného kódu binárneho stromu

VSTUP: Binárny strom B s koreňom v_0.

VÝSTUP: Kód $Z(v_0)$, jednoznačne určujúci daný binárny strom.

INICIALIZÁCIA

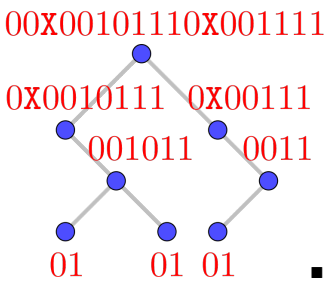
Každému koncovému vrcholu v_i priraď značku $Z(v_i) = 01$;

VYTVÁRANIE JEDNOZNAČNÉHO KÓDU BINÁRNEHO STROMU B

for { *pre každý vnútorný vrchol* v_i, *ktorého všetci synovia už majú značky* } **do**
  if { *ak* v_i *má ľavého aj pravého syna* } **then**
    $Z(v_i) := 0 + Z(v_{iL}) + Z(v_{iP}) + 1$;
  end
  if { *ak* v_i *má len ľavého syna* } **then**
    $Z(v_i) := 0 + Z(v_{iL}) + 1$;
  end
  if { *ak* v_i *má len pravého syna* } **then**
    $Z(v_i) := 0 + \mathtt{X} + Z(v_{iP}) + 1$;
  end
end

■ **Príklad 11.6**

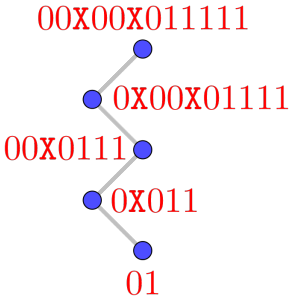
Výsledný kód binárneho stromu, ktorý vidíme na obrázku vpravo, je 00X00101110X001111. Značky jeho vrcholov sú uvedené na obrázku. Prideľovanie značiek sa začína od koncových vrcholov stromu.



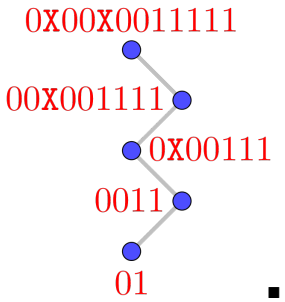
■

Postup vytvárania kódu ukážeme ešte na dvoch ďalších binárnych stromoch. Pôjde o dva vertikálne symetrické binárne stromy, ktoré by ako stromy, aj ako koreňové stromy, boli izomorfné. Ako binárne stromy však izomorfné nie sú, a preto budú mať rôzne kódy.

■ **Príklad 11.7**



Výsledný kód binárneho stromu na obrázku vľavo je 00X00X011111 a výsledný kód binárneho stromu na obrázku vpravo je 0X00X0011111. Ako vidíme, tieto kódy sú rôzne, čo znamená, že dané binárne stromy nie sú izomorfné.



■

Z algoritmu vytvárania identifikačného kódu binárneho stromu, ako aj z uvedených príkladov je zrejmé, že dva izomorfné binárne stromy musia mať rovnaké kódy. Všetky vlastnosti, ktoré

využívame pri zostavovaní kódov vrcholov (koncové vrcholy a ich počet, ľavý a pravý syn vrcholu, počet úrovní podstromu . . .) sú invarianty binárnych stromov. Takže, ak dva binárne stromy sú izomorfné, tak musia mať rovnaké kódy.

Opačným smerom dokážeme správnosť algoritmu tak, že ukážeme, ako sa z daného kódu dá **jednoznačne** zostrojiť pôvodný binárny strom. To znamená, že z jedného kódu nemôžeme dostať dva rôzne (neizomorfné) binárne stromy. Dôkaz nebudeme robiť celý, iba načrtneme jeho myšlienku. Najkratší kód, aký môžeme dostať, je 01. Takýto kód zodpovedá jednému izolovanému vrcholu. Pri rekonštrukcii binárneho stromu z kódu budeme postupovať od koreňa nadol, po úrovniach stromu. Zoberme si kód ľubovoľného vrcholu, označme si tento vrchol v_i. Tento kód má vždy tvar `0K1`, kde `K` je „zreťazenie“ kódov ľavého a pravého syna vrcholu v_i. Pre kód `K`, čiže kód vrcholu v_i bez ľavej krajnej nuly a pravej krajnej jednotky, budeme rozlišovať 2 možnosti. Kód `K` sa buď začína znakom `X`, alebo sa ním nezačína.

- ▷ Pokiaľ sa `K` začína znakom `X`, tak to znamená, že vrchol v_i nemá ľavého syna a `K` je kód jeho pravého syna.
- ▷ Ak sa `K` nezačína znakom `X`, tak využijeme skutočnosť, že kód každého vrcholu musí mať rovnako veľa znakov 0 ako znakov 1. Takže zo začiatku `K` vezmeme najkratšiu časť, v ktorej je rovnaký počet núl ako jedničiek. Táto časť bude kód ľavého syna vrcholu v_i a zvyšok `K`, bude kód pravého syna vrcholu v_i. Ak by nám z `K`, po zobratí najkratšej časti s rovnakým počtom 0 ako 1 z jeho začiatku, už nič nezvýšilo, tak to znamená, že vrchol v_i nemá pravého syna.

Uvedený postup rekonštrukcie stromu je deterministický a jednoznačný, čo znamená, že z jedného kódu môžeme spätne dostať len jeden binárny strom. Tento postup je vhodný na ukázanie toho, že stromy s rovnakým kódom sú izomorfné. Ale na reálne nakreslenie pôvodného stromu je tento postup príliš ťažkopádny. Na to si ukážeme iný postup. Ak chceme z daného kódu spätne zostrojiť pôvodný binárny strom, tak budeme postupovať nasledovne

1. Z kódu odstránime ľavú krajnú nulu a pravú krajnú jednotku. Ak si napríklad zoberieme kód binárneho stromu z príkladu 11.5, tak dostaneme $0001001011100111 \rightarrow 00100101110011$.
2. Všetky nuly v kóde nahradíme šípkou dole a všetky jednotky nahradíme šípkou hore. Pre lepšiu prehľadnosť si nuly a jednotky očíslujeme podľa poradia a ak sú v kóde znaky `X`, tak tie ignorujeme, t. j. nepridelíujeme im žiadne čísla. Kód binárneho stromu z príkladu 11.5 bude mať podobu

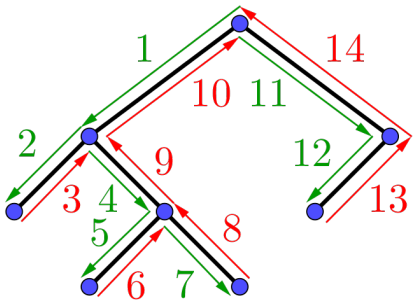
1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.	14.
0	0	1	0	0	1	0	1	1	1	0	0	1	1
↓	↓	↑	↓	↓	↑	↓	↑	↑	↑	↓	↓	↑	↑

3. Zostrojíme vrchol, ktorý bude koreňom stromu a, vychádzajúc z neho, začneme kresliť šípky. V binárnom strome môže mať každý vrchol najviac dvoch synov a rozlišujeme ich na ľavého a pravého syna. Takže ak sme v nejakom vrchole a nasleduje
   - ▷ znak `X`, tak ľavého syna tohto vrcholu zablokujeme.
   - ▷ šípka dole a tento vrchol ešte nemá ľavého syna a ani tento nie je blokovaný znakom `X`, tak nakreslíme šípku šikmo doľava, dole a na konci šípky zostrojíme nový vrchol.
   - ▷ šípka dole a tento vrchol už má ľavého syna, alebo je tento blokovaný znakom `X`, tak nakreslíme šípku šikmo doprava, dole a na konci šípky zostrojíme nový vrchol.
   - ▷ šípka hore, tak nakreslíme šípku idúcu šikmo hore, po protismerne idúcej šípke (orientovanej smerom dole).

▷ Ak sme už nakreslili všetky šípky, tak stačí len spojiť hranami zostrojené vrcholy. Každá dvojica protismerne idúcich šípok zodpovedá hrane.

Uvedený postup si ilustrujeme najskôr na príklade 11.5, ktorého kód, aj s prepísaním na šípky, už máme uvedený vyššie.

■ **Príklad 11.8**

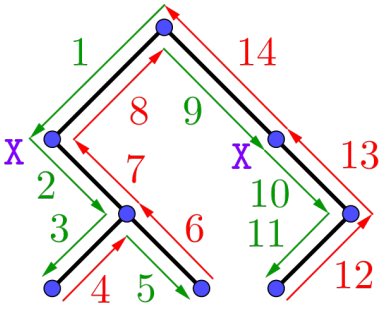


Zadaný je kód binárneho stromu 0001001011100111. Tabuľku s upraveným kódom, aj s jeho transkripciou pomocou šípok, máme uvedenú pred týmto príkladom na predošlej strane. Postup rekonštrukcie pôvodného binárneho stromu, ako aj zrekonštruovaný binárny strom, vidíme na obrázku vľavo.

■

V ďalšom texte si ešte ukážeme rekonštrukciu binárnych stromov z príkladov 11.6 a 11.7.

■ **Príklad 11.9**

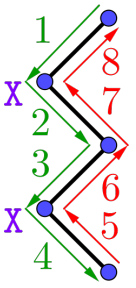


Zadaný je kód binárneho stromu 00X00101110X001111. Upravený kód, s vynechanou ľavou krajnou 0 a pravou krajnou 1 a jeho transkripciu pomocou šípok, máme zobrazený v nižšie uvedenej tabuľke. Postup rekonštrukcie binárneho stromu vidíme na obrázku vľavo. Všimnime si, že po 1. a po 9. šípke sme zablokovali ľavého syna príslušného vrcholu znakom X, takže nasledujúca šípka dole musí ísť smerom doprava.

00X00101110X001111 → 0X00101110X00111															
1.	—	2.	3.	4.	5.	6.	7.	8.	9.	—	10.	11.	12.	13.	14.
0	X	0	0	1	0	1	1	1	0	X	0	0	1	1	1
↓	X	↓	↓	↑	↓	↑	↑	↑	↓	X	↓	↓	↑	↑	↑

■

■ **Príklad 11.10**

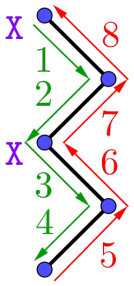


Zadaný kód binárneho stromu je 00X00X011111. Vynechaním ľavej krajnej 0 a pravej krajnej 1 a následnou transkripciou na šípky dostaneme

00X00X011111 → 0X00X01111									
1.	—	2.	3.	—	4.	5.	6.	7.	8.
0	X	0	0	X	0	1	1	1	1
↓	X	↓	↓	X	↓	↑	↑	↑	↑

Postup rekoštrukcie binárneho stromu a výsledný strom, vidíme na ľavom obrázku.

Zadaný kód binárneho stromu je 0X00X0011111. Vynechaním ľavej krajnej 0 a pravej krajnej 1 a následnou transkripciou na šípky dostaneme

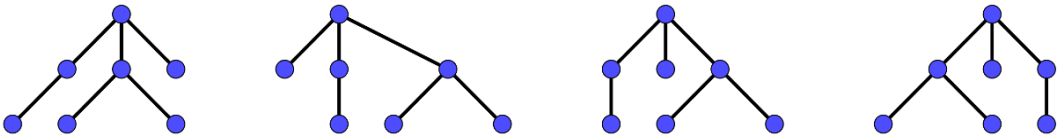


0X00X0011111 → X00X001111									
—	1.	2.	—	3.	4.	5.	6.	7.	8.
X	0	0	X	0	0	1	1	1	1
X	↓	↓	X	↓	↓	↑	↑	↑	↑

Postup rekoštrukcie binárneho stromu a výsledný strom, vidíme na obrázku vpravo. ∎

11.2.2 Algoritmus overovania izomorfizmu koreňových stromov

V predošlej časti sme ukázali, ako sa overuje izomorfizmus binárnych stromov. Teraz ukážeme, ako sa dá uvedený postup modifikovať, aby sme pomocou neho mohli overiť izomorfizmus koreňových stromov. Koreňové stromy sa od binárnych líšia tým, že každý ich vnútorný vrchol môže mať aj viac než 2 synov a na ľavo-pravom poradí jeho synov nezáleží. Takže štyri koreňové stromy na obrázku 11.4 sú všetky navzájom izomorfné. Pri overovaní izomorfizmu dvoch koreňových stromov pomocou ich identifikačného kódu musíme preto synov a príslušné podstromy každého vrcholu usporiadať. Na druhej strane nebudeme pri zápise kódu koreňového stromu potrebovať znak X, pretože v koreňovom strome nerozlišujeme, či syn nejakého vrcholu je ľavý, alebo pravý. Algoritmus konštrukcie identifikačného kódu koreňového stromu je uvedený nižšie.



Obr. 11.4: Štyri izomorfné koreňové stromy

Vytváranie identifikačného kódu koreňového stromu

VSTUP: Koreňový strom R s koreňom v_0.

VÝSTUP: Kód $Z(v_0)$, jednoznačne určujúci daný koreňový strom.

INICIALIZÁCIA

Každému koncovému vrcholu v_i priraď značku $Z(v_i) = 01$;

VYTVÁRANIE JEDNOZNAČNÉHO KÓDU KOREŇOVÉHO STROMU R

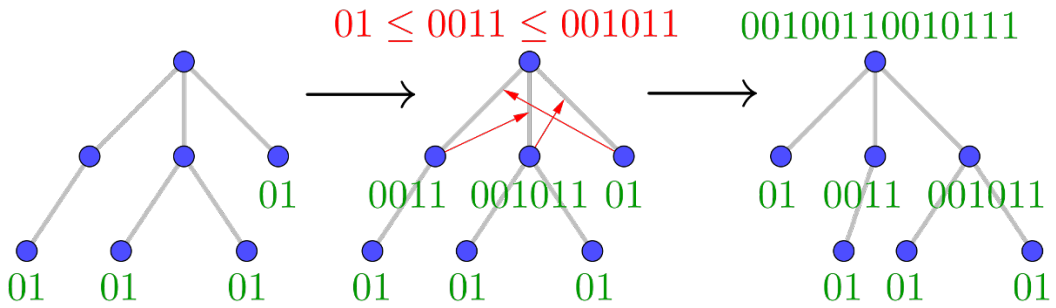
for { *pre každý vnútorný vrchol* v_i } **do**
  if { *vrchol* v_i *má* k *synov* } **then**
    Usporiadaj synov vrcholu v_i zľava-doprava tak, aby platilo $Z(v_{i1}) \leq \ldots \leq Z(v_{ik})$;
  end
  $Z(v_i) := 0 + Z(v_{i1}) + \ldots Z(v_{ik}) + 1$;
end

Ako vidíme zo zápisu algoritmu, najskôr pridelíme všetkým koncovým vrcholom značky 01, rovnako ako pri binárnych stromoch. Potom postupujeme od koncových vrcholov ku koreňu. Predtým,

než pridelíme značku nejakému vrcholu, musia mať už všetci jeho synovia pridelené značky a všetkých jeho synov musíme lineárne usporiadať zľava-doprava podľa ich značiek. Je potrebné si preto uvedomiť, že symbol $\leq$ v zápise algoritmu znamená nejaké lineárne usporiadanie[2] značiek vrcholov. Môžeme napríklad zvoliť také lineárne usporiadanie, pri ktorom značky usporiadame podľa počtu ich cifier a ak budú mať dve značky rovnaký počet cifier, tak ich usporiadame podľa veľkosti, ako čísla. Napríklad $01 < 11 < 0011 < 001011 < \ldots$ Týmto usporiadaním synov vrcholu je zabezpečené to, že dva izomorfné koreňové stromy budú mať rovnaký kód. Nemôže totiž nastať prípad aký vidíme na obrázku 11.4, kde synov a príslušné podstromy nejakého vrcholu ľubovoľne preusporiadavame. Postup si ilustrujeme na príklade prvého koreňového stromu z obrázku 11.4.

■ **Príklad 11.11** Ideme nájsť kód ľavého krajného koreňového stromu z obrázku 11.4. Postup je znázornený na obrázku 11.5 a bude nasledovný.

1. V prvom kroku dáme koncovým vrcholom značky 01.
2. V druhom kroku ideme na predposlednú úroveň stromu a pridelíme značky dvom vnútorným vrcholom.
3. V treťom kroku sme sa dostali ku koreňu, ktorý má 3 synov so značkami 0011, 001011 a 01. Pri vyššieuvedenom lineárnom usporiadaní, je ich poradie $01 < 0011 < 001011$, takže zodpovedajúcim spôsobom preusporiadame synov koreňa a príslušné podstromy. Koreňu následne pridelíme značku 00100110010111, ktorá je zároveň identifikačným kódom koreňového stromu.



Obr. 11.5: Konštrukcia kódu koreňového stromu

■

Z uvedeného postupu je zrejmé, že izomorfné koreňové stromy musia mať rovnaké identifikačné kódy. Ak by sme chceli z identifikačného kódu spätne nakresliť koreňový strom, tak postup bude v podstate rovnaký ako pri rekonštrukcii binárneho stromu z jeho kódu. Rozdiel bude len v dvoch bodoch. Po prvé, kódy koreňových stromov neobsahujú žiaden znak `X`, a preto všetky časti rekonštrukčného algoritmu, ktoré sa týkajú znaku `X`, môžeme vynechať. A po druhé, v koreňových stromoch sa synovia nerozlišujú na ľavého a pravého a vrchol môže mať aj viac než 2 synov. Preto, ak z nejakého vrcholu máme nakresliť šípku dole, tak túto šípku nakreslíme vpravo od poslednej šípky, ktorá už vedie z tohto vrcholu dole. Zvyšok algoritmu zostáva nezmenený.

11.2.3 Algoritmus overovania izomorfizmu stromov

Koreňové stromy sa od stromov líšia len tým, že majú jeden pevne vybraný vrchol, ktorý sa nazýva koreň. Takže ak chceme stromom prideľovať kódy, na základe ktorých vieme povedať, či dané dva stromy sú izomorfné, musíme si vhodným spôsobom nájsť koreň týchto stromov.

V definícii 3.3.5 (strana 56) bol definovaný pojem *centrum grafu*. Pripomíname, že centrum grafu je množina tých vrcholov daného grafu, ktoré majú minimálnu excentricitu. Voľne môžeme

[2] Pozri definíciu 7.1.3 na strane 138.

povedať, že centrum je tvorené vrcholmi, ktorých vzdialenosť od všetkých ostatných vrcholov grafu je najmenšia možná. Sú typy grafov, napríklad cykly, v ktorých je centrum tvorené celou ich vrcholovou množinou. Na druhej strane spektra sú stromy, ktorých centrum zahŕňa len minimálnu časť ich vrcholovej množiny. Platí nasledujúca veta.

Veta 11.2.1 — O centre stromu. Centrum stromu pozostáva buď z jedného vrcholu, alebo z dvoch susedných vrcholov.

Dôkaz: pre stromy s 1 alebo s 2 vrcholmi, tvrdenie triviálne platí. Vezmime si teraz strom T, ktorý má viac než 2 vrcholy. Veľmi ľahko sa nahliadne, že listy sú tie vrcholy stromu, ku ktorým má každý vrchol stromu najväčšiu vzdialenosť (pozri cvičenie 11.7). Z toho potom vyplýva, že ak stromu T odstránime všetky jeho pôvodné (len pôvodné, pretože odstránením listu, môže vzniknúť nový list) listy, tak excentricita každého zostávajúceho vrcholu sa zníži práve o 1. Okrem toho je zrejmé, že pokiaľ má strom viac než 2 vrcholy, tak list nemôže byť centrálny vrchol. Preto ak strom T' dostaneme zo stromu T odstránením všetkých pôvodných listov stromu T, tak platí $C(T) = C(T')$. Inak povedané, centrum stromu sa po aplikácii operácie odstránenia všetkých jeho pôvodných listov zachováva. Strom má konečný počet vrcholov, takže krok s odstraňovaním všetkých jeho listov budeme opakovať dovtedy, kým výsledný graf bude mať viac než 2 vrcholy. Po konečnom počte krokov musíme skončiť, pretože nám buď zostane len jeden vrchol, alebo nám zostanú dva vrcholy spojené hranou. V oboch prípadoch ide o centrum pôvodného stromu T. Q.E.D.

Veta 11.2.1 je síce veľmi jednoduchá, avšak pre algoritmus zisťovania izomorfizmu stromov je kľúčová. Vlastnosť „byť centrom grafu“ je totiž invariant (pozri cvičenie 11.8). Dokáže sa to veľmi jednoducho pomocou toho, že aj vlastnosť „vzdialenosť ľubovoľných dvoch vrcholov grafu“ je invariant. Vďaka vete 11.2.1 vieme každému stromu priradiť jednoznačný identifikačný kód tak, že dva stromy budú izomorfné práve vtedy, keď ich identifikačné kódy budú rovnaké. Postup konštrukcie identifikačného kódu pre daný strom je nasledovný.

1. Pre daný strom T nájdeme jeho centrum $C(T)$.
2. Ak je centrum tvorené len jedným vrcholom, tak tento vrchol zoberieme ako koreň a kód príslušného koreňového stromu bude kód stromu T.
3. Ak je centrum tvorené dvoma susednými vrcholmi $\{c_1, c_2\}$, tak zo stromu T odstránime hranu (c_1, c_2). Vynechaním hrany (c_1, c_2) sa strom T rozpadne na dva komponenty T_1 a T_2, pričom c_1 patrí do T_1 a c_2 patrí do T_2. Nájdeme kódy koreňových stromov (T_1, c_1) a (T_2, c_2). Označme kódy týchto dvoch koreňových stromov, v uvedenom poradí, ako K_1 a K_2. Potom
   - ak $K_1 \leq K_2$, kde symbol $\leq$ označuje lexikografické[3] usporiadanie, tak za koreň stromu T zvolíme vrchol c_1 a kód stromu T bude kód koreňového stromu (T, c_1).
   - ak $K_2 \leq K_1$, kde symbol $\leq$ označuje lexikografické usporiadanie, tak za koreň stromu T zvolíme vrchol c_2 a kód stromu T bude kód koreňového stromu (T, c_2).

11.3 Hra NIM

Hra NIM[4] údajne vznikla v Číne a v Európe pochádzajú prvé zmienky o nej zo 16. storočia. Táto hra bola v rôznych regiónoch sveta a v rôznych dobách známa pod rozličnými menami. Názov NIM pochádza od amerického matematika Boutona z Harvardskej univerzity, ktorý v roku 1901 vypracoval kompletnú teóriu tejto hry. Hra NIM bola jedna z prvých počítačových hier. Už v roku

[3] Lexikografické usporiadanie je lineárne usporiadanie.
[4] `https://en.wikipedia.org/wiki/Nim`

1940 postavili v New Yorku stroj *Nimatron*, na ktorom sa dala hrať hra NIM. V minulosti, v časoch pred príchodom PC, mobilov a smartfónov, bola hra NIM veľmi populárna aj medzi mládežou. My sa budeme zaoberať hrou NIM preto, lebo táto hra má víťaznú stratégiu a naviac sa táto víťazná stratégia dá opísať pomocou grafov.

11.3.1 Pravidlá hry NIM

Existuje množstvo variantov hry NIM. V zásade však ide vždy o to, že sa pridávajú, alebo odoberajú nejaké objekty a ten hráč, ktorý dosiahne určené množstvo, alebo zoberie posledný objekt, prehráva, alebo vyhráva – v závislosti od verzie hry. Hra NIM môže vyzerať napríklad takto

> *Hru NIM hrajú dvaja hráči. Na začiatku hry je na stole kôpka 100 zápaliek a hráči z nej striedavo odoberajú 1 až 10 zápaliek. To znamená, že každý hráč musí v každom ťahu odobrať z kôpky najmenej 1 zápalku a najviac 10 zápaliek. Hráč, ktorý zoberie poslednú zápalku, prehráva.*

Pozrieme sa teraz na víťaznú stratégiu hry NIM.

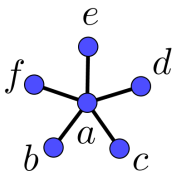
11.3.2 Hra NIM a grafy

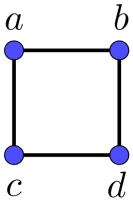
Najskôr si ukážeme, ako môžeme hru NIM reprezentovať pomocou orientovaných grafov. Počtu zápaliek v kôpke priradíme čísla 1 až 100 a každému číslu bude zodpovedať vrchol grafu. Dva vrcholy grafu, označme si ich napríklad u, v, budú spojené orientovanou hranou práve vtedy, ak sa v jednom ťahu vieme dostať z počtu u na počet v. V takom prípade pôjde orientovaná hrana (šípka) z vrcholu u do vrcholu v. Napríklad z vrcholu 100 pôjdu šípky do vrcholov 90, 91, 92, 93, 94, 95, 96, 97, 98 a 99. Predtým než budeme pokračovať v popise víťaznej stratégie hry, zadefinujeme tri nové pojmy a dokážeme jednoduchú vetu (to všetko pre neorientované grafy).

Definícia 11.3.1 — Stabilná množina vrcholov. Nech $G = (V,E)$ je graf. Potom $S \subseteq V$ sa nazýva stabilná množina vrcholov grafu G, ak každý vrchol z $V \setminus S$ je susedný s aspoň jedným vrcholom z S.

▪ Príklad 11.12

Napríklad v grafe na obrázku vpravo množina $S_1 = \{a\}$ **je** stabilná a množina $S_2 = \{f,c\}$ **nie je** stabilná.





V grafe na obrázku vľavo množina $S_1 = \{a,d\}$ **je** stabilná, množina $S_2 = \{a,b\}$ takisto **je** stabilná a množina $S_3 = \{a\}$ **nie je** stabilná.

▪

Definícia 11.3.2 — Nezávislá množina vrcholov. Nech $G = (V,E)$ je graf. Potom $N \subseteq V$ sa nazýva nezávislá množina vrcholov grafu G, ak žiadne dva vrcholy z množiny N nie sú susedné.

Pomocou predchádzajúcich dvoch pojmov môžeme definovať pojem *jadro grafu*.

Definícia 11.3.3 — Jadro grafu. Nech $G = (V,E)$ je graf. Potom $J \subseteq V$ sa nazýva jadro grafu G, ak je množina J stabilná a zároveň nezávislá.

■ **Príklad 11.13** V prípade prvého grafu (hviezda) z príkladu 11.12 množina $S_1 = \{a\}$ **je** jadrom tohto grafu.
V prípade druhého grafu (cyklus) z príkladu 11.12 množina $S_1 = \{a,d\}$ **je** jadrom daného grafu ale množina $S_2 = \{a,b\}$ **nie je** jadrom daného grafu, pretože nie je nezávislá. ■

O jadre grafu platí nasledujúca veta.

Veta 11.3.1 — O existencii jadra. Každý konečný súvislý graf bez slučiek má jadro.

Dôkaz: spravíme popisom algoritmu konštrukcie jadra grafu. Budeme používať nasledovné označenie. Ak $G = (V,E)$ je daný graf a $M \subseteq V$, tak zápisom $o(M)$ budeme označovať množinu všetkých vrcholov grafu G, ktoré susedia s niektorým vrcholom z množiny M, t. j. $o(M) \subseteq V$.
Algoritmus konštrukcie jadra daného grafu $G = (V,E)$ je nasledovný:

1. Zoberme si ľubovoľný $v_1 \in V$ a položme $J = \{v_1\}$.
2. Ak $o(J) = V \setminus J$, tak J je jadro grafu G a algoritmus skončí.
3. Zvoľme si ľubovoľný $v_i \in V \setminus o(J)$, položme $J := J \cup \{v_i\}$ a ideme na krok 2.

Keďže daný graf je konečný, po konečnom počte krokov dostaneme jadro grafu G.

V prípade orientovaných grafov musíme upraviť definíciu stabilnej množiny. Ak z vrcholu u ide orientovaná hrana (šípka) do vrcholu v, tak budeme hovoriť, že vrchol v je *dosiahnuteľný* z u. Stabilná množina vrcholov v orientovanom grafe bude taká množina $S \subseteq V$, pre ktorú z každého vrcholu z množiny $V \setminus S$ je dosiahnuteľný aspoň jeden vrchol z S. Definície nezávislej množiny a jadra potom aj pre orientované grafy zostávajú rovnaké ako pre neorientované grafy. Výrok *„žiadne dva vrcholy z množiny N nie sú susedné“* znamená, že žiadny vrchol z N nie je dosiahnuteľný zo žiadneho vrcholu z N.

11.3.3 Víťazná stratégia hry NIM

Skúmajme hru NIM, s pravidlami uvedenými v časti 11.3.1. Je zrejmé, že prehráva ten hráč, ktorý je na ťahu v momente, keď na stole zostala už len 1 zápalka. Ako môže jeho súper dosiahnuť tento stav? Vyhrávajúci hráč musí mať v momente pred svojím posledným ťahom na stole 2 až 11 zápaliek. To sú všetky počty zápaliek, z ktorých sa odobratím 1 až 10 zápaliek vie dostať na konečný počet 1. Pri grafovej reprezentácii hry NIM opísanej v časti 11.3.2 to znamená, že vrchol 1 je dosiahnuteľný z vrcholov 2 až 11. Ak by sme v tejto úvahe postupovali ďalej, tak dostaneme vrcholy orientovaného grafu, po ktorých sa hráč musí pohybovať, ak chce hru vyhrať.

Hra NIM, s pravidlami uvedenými v časti 11.3.1, má preto vyhrávajúcu stratégiu pre 2. hráča. Tomuto hráčovi stačí, ak bude v každom ťahu odoberať z kôpky toľko zápaliek, aby skončil vo vrcholoch grafu, ktoré sú v nasledujúcom zozname zobrazené čiernou farbou.

100, 99, . . . , 90, 89, 88, . . . , 79, 78, 77, ... , 24, 23, 22, . . . , 13, 12, 11, . . . , 2, 1

Množina vrcholov $J = \{1,12,23,34,45,56,67,78,89,100\}$ tvorí jadro orientovaného grafu hry NIM. Tieto vrcholy sú nezávislé a z každého vrcholu hry NIM nepatriaceho do J, vieme dosiahnuť niektorý vrchol množiny J. Preto ak 2. hráč skončí svoj ťah v niektorom vrchole množiny J, tak 1. hráč sa nemôže dostať do iného vrcholu množiny J, ale po jeho ťahu sa 2. hráč vždy vie dostať do nasledujúceho vrcholu množiny J. Pre hru s pravidlami uvedenými v časti 11.3.1 spočíva víťazná stratégia pre 2. hráča v pohybovaní sa po vrcholoch takého jadra grafu hry, ktoré obsahuje vrchol 1. A to je práve množina J uvedená vyššie.

Všetky varianty hry NIM sa dajú reprezentovať pomocou orientovaných grafov a ich víťazná stratégia vždy súvisí s jadrom grafu hry. Hra NIM pritom nie je jediná hra s grafovou reprezentáciou. My sme túto hru použili len ako jednoduchú ukážku takejto kategórie hier na ilustráciu možností aplikácií teórie grafov.

11.4 Cvičenia

Cvičenie 11.1 Zostrojte binárny prehľadávací strom pre množinu Ω pri uvedenom poradí jej prvkov

(a) $\Omega = \{G, V, M, Z, L, A, I, K, F, N, R\}$
(b) $\Omega = \{K, V, C, T, S, M, I, B, N, G, A\}$
(c) $\Omega = \{M, G, S, R, V, C, I, B, O, T\}$
(d) $\Omega = \{H, Y, T, A, L, S, R, K, O, E, X\}$

■

Cvičenie 11.2 Je nasledovné tvrdenie pravdivé? Svoju odpoveď riadne zdôvodnite.

> *Nech B je binárny strom. Ak pre každý jeho vrchol v je dátová položka vložená do v väčšia, než dátová položka vložená v ľavom synovi vrcholu v a zároveň menšia, než dátová položka vložená v pravom synovi vrcholu v, tak B je prehľadávací binárny strom.*

■

Cvičenie 11.3 Pre všetky binárne stromy z cvičenia 5.19 (strana 118) zostavte ich kódy a pomocou nich overte, či sú tieto binárne stromy izomorfné. ■

Cvičenie 11.4 Pre všetky koreňové stromy z cvičenia 5.18 (strana 117) zostavte ich kódy a pomocou nich overte, či sú tieto koreňové stromy izomorfné. ■

Cvičenie 11.5 Pre všetky stromy z cvičenia 5.17 (strana 117) zostavte ich kódy a pomocou nich overte, či sú tieto stromy izomorfné. ■

Cvičenie 11.6 Dokážte, že dva stromy (bez prívlastkov) sú izomorfné práve vtedy, ak majú rovnaký kód skonštruovaný postupom opísaným v časti 11.2.3. ■

Cvičenie 11.7 Dokážte, že ak G je strom, tak jeho vrchol s maximálnou excentricitou je list. ■

Cvičenie 11.8 Dokážte, že vlastnosť „byť centrom grafu“ je invariant. Inak povedané, ak máme dva izomorfné grafy G a G', tak $C(G)$ sa izomorfizmom musí zobraziť na $C(G')$. ■

Cvičenie 11.9 Dokážte, že existuje maximálne 4^n navzájom neizomorfných stromov na n vrcholoch. ■

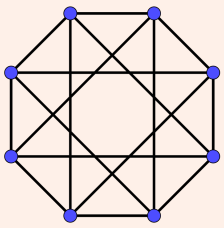
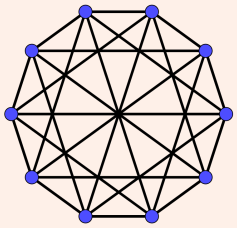
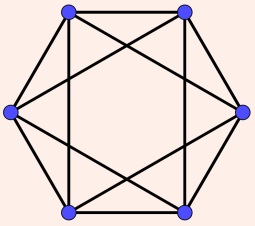
Cvičenie 11.10 V Pythone alebo v C napíšte program, ktorý pre dané dva binárne stromy nájde ich identifikačné kódy a pomocou nich overí, či sú izomorfné. ■

Cvičenie 11.11 V Pythone alebo v C napíšte program, ktorý pre dané dva koreňové stromy nájde ich identifikačné kódy a pomocou nich overí, či sú izomorfné. ■

Cvičenie 11.12 V Pythone alebo v C napíšte program, ktorý pre dané dva stromy nájde ich identifikačné kódy a pomocou nich overí, či sú izomorfné. ■

Cvičenie 11.13 Pre ktorého hráča by existovala a ako by vyzerala víťazná stratégia hry NIM (podľa pravidiel z časti 11.3.1), ak by na začiatku hry bolo na stole 150 zápaliek? ■

Cvičenie 11.14 Nájdite nejaké stabilné a nejaké nezávislé množiny vrcholov nasledujúcich grafov tak, aby tieto množiny boli / neboli jadrá daných grafov

■

Cvičenie 11.15 Dokážte, že každý konečný graf bez slučiek má jadro. ■

Cvičenie 11.16 Dvaja hráči hrajú nasledovnú hru

> *Prvý hráč povie číslo od 1 do 10 a potom k tomuto číslu hráči striedavo v každom ťahu pripočítavajú číslo od 1 do 10. Hru vyhráva hráč, ktorý ako prvý dosiahne číslo 100.*

Má táto hra víťaznú stratégiu? Ak áno, pre ktorého hráča a ako vyzerá táto stratégia? ■

Cvičenie 11.17 Zovšeobecnite víťaznú stratégiu hry z cvičenia 11.16, ak hráči v každom ťahu pripočítavajú číslo od 1 po n, kde $n \in \mathbb{N}$ a vyhráva ten hráč, ktorý ako prvý dosiahne hodnotu x, kde $x \in \mathbb{N}$. ■

Cvičenie 11.18 Dokážte, že každá nezávislá množina vrcholov daného grafu G je obsiahnutá v nejakom jeho jadre. ■

Cvičenie 11.19 V Pythone alebo v C napíšte program, ktorý bude proti ľudskému súperovi hrať hru NIM s pravidlami uvedenými v časti 11.3.1. Pritom

- počiatočný počet zápaliek v kôpke bude X, kde $X \geq 1$ je prirodzené číslo,
- hráči budú v každom ťahu hry odoberať z kôpky 1 až N zápaliek, kde $N \geq 1$ je prirodzené číslo,
- ľudský hráč si pred začiatkom hry môže zvoliť, či chce hrať ako 1., alebo ako 2. hráč,
- program bude hrať „inteligentne“, čiže bude sa snažiť hru vyhrať.

■

Cvičenie 11.20 V Pythone alebo v C napíšte program, ktorý bude proti ľudskému súperovi hrať hru opísanú v cvičení 11.16 a zovšeobecnenú v cvičení 11.17. Pritom

- hráči budú v každom ťahu hry pripočítavať číslo 1 až n, kde $n \in \mathbb{N}$,
- vyhráva ten hráč, ktorý ako prvý dosiahne číslo x, kde $x \in \mathbb{N}$,
- ľudský hráč si pred začiatkom hry môže zvoliť, či chce hrať ako 1., alebo ako 2. hráč,
- program bude hrať „inteligentne“, čiže bude sa snažiť hru vyhrať.

■

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.

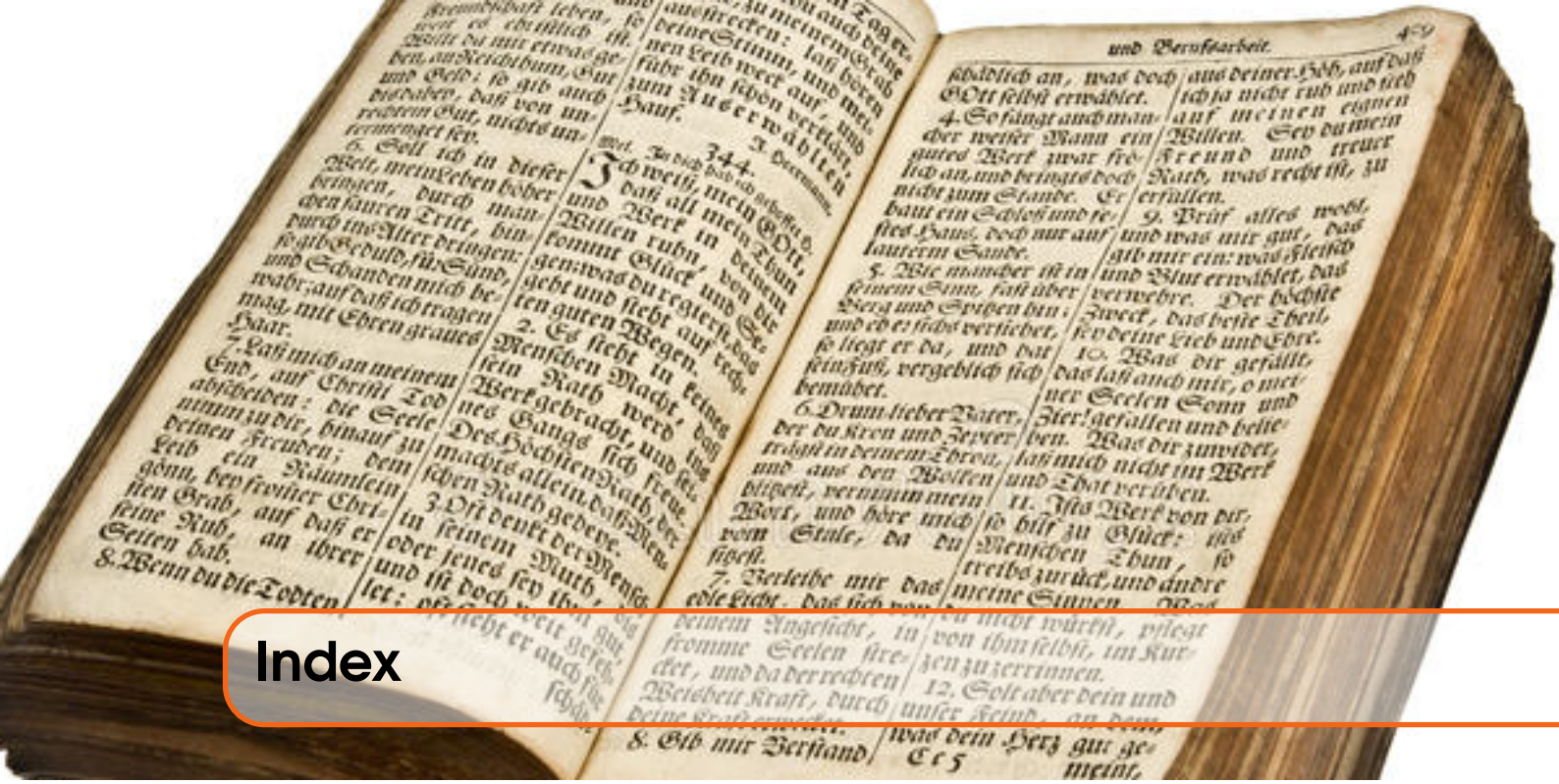
[30] Herbert S. Wilf: **generatingfunctionology**, Academic Press, Inc. 1994. `https://www.math.upenn.edu/~wilf/DownldGF.html`

[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

M

N

P

R

S

T

Ť

U

Ú

V

W

Z