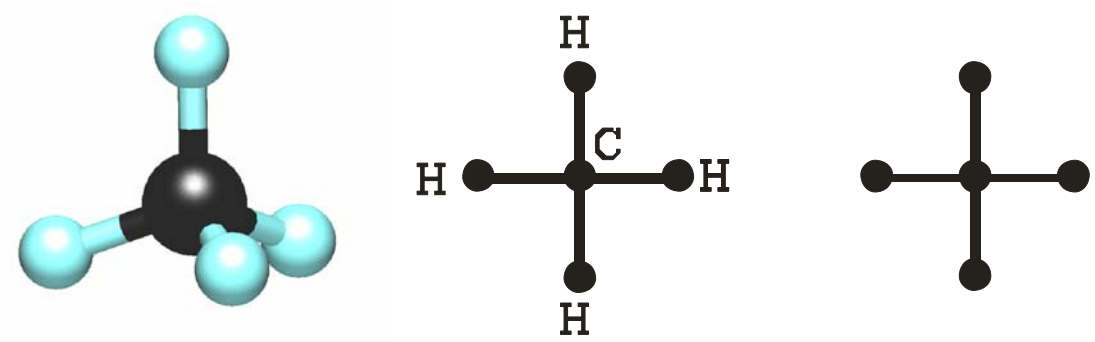
12. kapitola

Teória grafov III – stromy ako modely, vlastnosti stromov, binárne prehľadávanie, prefixové kódy, stromy algebraických výrazov, hry

12.1 Stromy ako modely a ich základné vlastnosti

Strom je súvislý graf bez kružníc. Prvý krát boli stromy použité už anglickým matematikom Arthurom Cayleym[1] r. 1857 na spočítanie druhov istého typu chemických zlúčenín – alkánov (pozri obr. 12.1). Alkány sa inak volajú nasýtené uhľovodíky a majú sumárnu formulu C_nH_{2n+2}. V grafovom modeli je každý atóm uhlíka reprezentovaný vrcholom stupňa 4 a každý vodíkový atóm vrcholom stupňa 1. V grafe reprezentujúcom alkány je teda je $3n+2$ vrcholov, a keďže počet hrán je polovicou sumy stupňov vrcholov, ich počet je $(4n+2n+2)/2=3n+1$ hrán. (Dôkaz faktu, že taký súvislý graf nemá kružnice, je ponechaný na cvičenie 12.5). Neizomorfné stromy o n vrcholoch stupňa 4 a o $2n+2$ vrcholoch stupňa 1 reprezentujú rozdielne izoméry C_nH_{2n+2} (pozri cvičenie 12.8).



priestorový model štruktúrny vzorec graf

Obrázok 12.1. Ukážka rôznych typov reprezentácie chemickej zlúčeniny metánu, ktorý patrí medzi alkány.

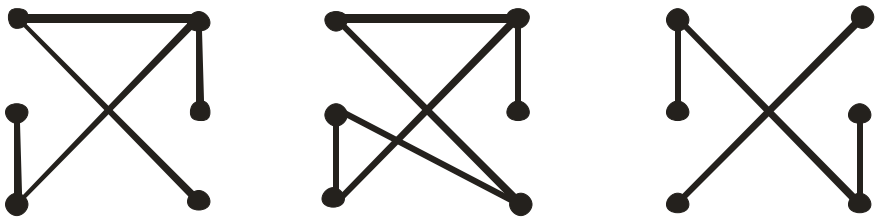
Od Cayleyho doby boli stromy použité na riešenie problémov v množstve disciplín. V bežnom živote sa s použitím stromov môžeme stretnúť od genealógie – stromu príbuznosti u rodokmeňa, klasifikačného stromu živočíchov a rastlín, stromu športového turnaja až po organizačnú štruktúru hierarchie podniku, kde vrcholy sú jednotlivé organizačné jednotky a hrany predstavujú vzťahy nadriadenosti-podriadenosti medzi nimi. Najobyčajnejším využitím stromov v počítači je štruktúra adresárov. Stromy sa ale nepoužívajú iba na opis štruktúr, ale predovšetkým ako pomôcka na manipulácie s informáciou. Je to napríklad:

- rozmiestňovanie prvkov v databázach
- efektívne kódy na ukladanie a prenos informácií
- rozhodovacie stromy
- modely hier na určenie vyhrávajúcej stratégie

[1] Arthur Cayley (1821-1895) vyštudoval v Cambridgi, ale pretože nebola voľná pozícia matematika, dal sa na právnickú kariéru, kde sa stal odborníkom, napriek tomu, že v priebehu 15 ročnej kariéry právnika napísal 300 matematických článkov. Potom, čo sa v Cambridgi uvoľnilo miesto matematika, Caley sa naň prihlásil, aj keď išiel so zárobkom podstatne dolu.

- nájdenie najlacnejšieho prepojenia uzlov komunikačnej siete
- prehľadávanie stromov (napríklad riešenie problému rozmiestnenia dám na šachovnici, aby sa neohrozovali).

Nesúvislý graf bez cyklov (skladá sa zo stromov) voláme les (pozri obr. 12.2).



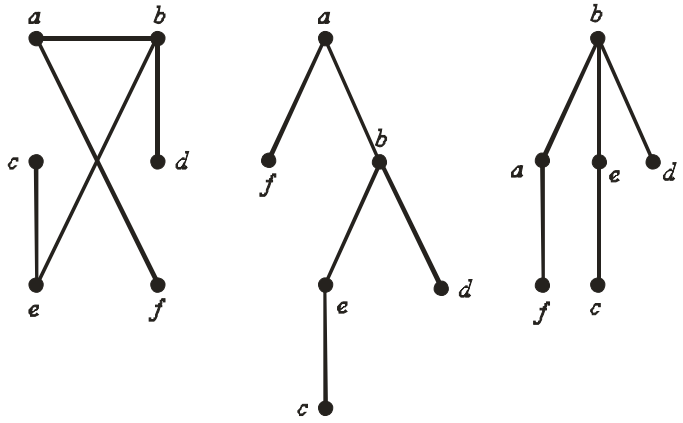
Obrázok 12.2. Ukážka stromu, grafu s cyklom, ktorý teda nie je strom, a lesu, skladajúceho sa z dvoch stromov. Aj keď zvyčajne sa stromy zakresľujú bez kríženia hrán, keďže všetky stromy sú planárne grafy, strom ostáva stromom bez ohľadu na grafickú reprezentáciu.

Veta 12.1: Neorientovaný graf je stromom iba vtedy, ak existuje práve jedna cesta medzi ľubovoľnou dvojicou jeho vrcholov.

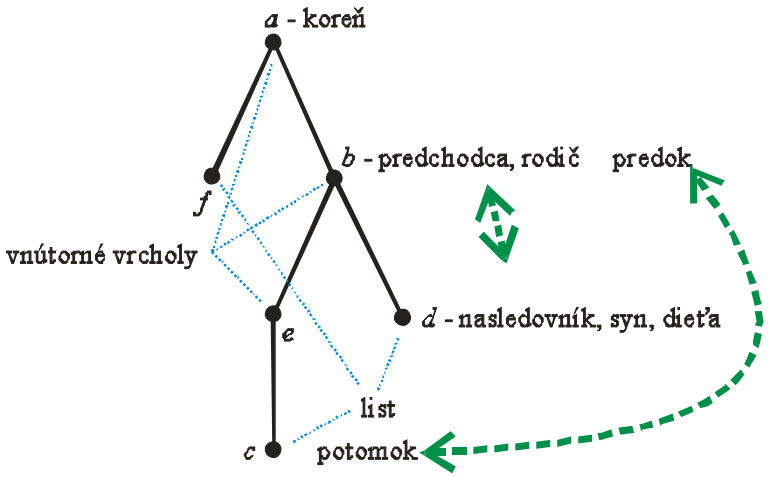
Dôkaz: Predpokladajme, že T je strom. Potom T je súvislý graf bez kružníc. Nech x a y sú dva vrcholy z T. Pretože T je súvislý, medzi x a y existuje cesta. Táto cesta musí byť jediná, pretože keby existovali 2 rôzne cesty, dala by sa zobrať cesta z x do y a naspäť z y do x, a z rozdielnych hrán týchto ciest by sa dala zostaviť kružnica. To je v spore s predpokladom, že T je súvislý graf bez kružníc.

Teraz predpokladajme, že existuje práve jedna cesta medzi ľubovoľnými dvoma vrcholmi grafu. Potom je graf súvislý a neobsahuje kružnicu. Aby sme to ukázali, predpokladajme, že existuje kružnica obsahujúca body x a y. Potom by medzi x a y existovali dve cesty, pretože kružnica obsahujúca body x a y sa dá rozdeliť na cestu z x do y a na druhú cestu z y do x. Z toho vyplýva, že graf s práve jednou cestou medzi ľubovoľnou dvojicou jeho vrcholov je strom. ■

V mnohých aplikáciách sa využíva strom so špeciálne vyznačeným vrcholom volaným ***koreň***. Taký strom sa volá ***koreňový*** (alebo *zakorenený*) ***strom*** (rooted tree). Keď máme vybratý takýto koreň, môžeme priradiť každej hrane orientáciu smerom od koreňa. Výberom koreňa môžeme z jedného stromu vytvoriť dva rôzne koreňové stromy, pozri obr. 12.3. Koreňové stromy preberajú ďalšiu terminológiu tak z genealógie, ako z biológie, pozri obr. 12.4. Predpokladajme, že T je koreňový strom a v je jeho vrchol rôzny od koreňa. ***Predchodca*** (alebo rodič – parent) vrcholu v je práve jeden vrchol u, kde (u, v) je orientovaná hrana na ceste z koreňa do vrcholu v. Vrchol v sa potom volá ***nasledovník*** (syn, dieťa - child) vrcholu u. ***Predok*** (ancestor) vrcholu v je okrem koreňa tiež každý vrchol na ceste od koreňa k danému vrcholu v, s výnimkou samotného vrcholu v. ***Potomok*** (descendant) vrcholu v je ľubovoľný vrchol, ktorý má vrchol v ako predka. Vrchol stromu je volaný ***list***, keď nemá žiadne deti. Vrcholy, ktoré majú deti, sa volajú ***vnútorné vrcholy***. Keď b je vrchol stromu, ***podstrom*** s koreňom b je podgraf stromu zostavený z vrcholu b, všetkých jeho potomkov a všetkých hrán incidentných s potomkami, napr. na obr. 12.3 u druhého grafu by to bol podstrom určený vrcholovou množinou $\{b,e,c,d\}$.

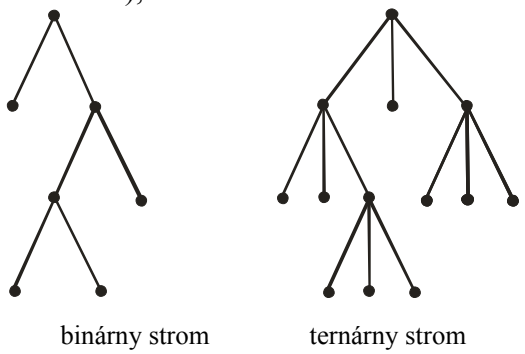


Obrázok 12.3. Ukážka stromu a rôznych koreňových stromov vytvorených z pôvodného stromu rôznym výberom koreňa.



Obrázok 12.4. Ukážky pre termíny u koreňových stromov.

Úroveň vrcholu je dĺžka cesty od vrcholu ku koreňu. ***Hĺbka stromu*** (height, teda výška) je maximálna úroveň vrcholov. Koreňový strom sa volá ***n-árny strom*** keď každý vrchol má maximálne *n* detí. Koreňový strom sa volá ***plne n-árny strom*****,** keď každý vnútorný vrchol má práve *n* detí. Pre *n*=2 sa *n*-árny strom volá ***binárny strom***, pre *n*=3 sa *n*-árny strom volá ***ternárny strom*** (pozri obr. 12.5),



Obrázok 12.5. Plne binárny a plne ternárny strom.

Usporiadaný koreňový strom je koreňový strom, kde deti každého vnútorného vrcholu sú usporiadané, kreslia sa poporiadku zľava doprava. V binárnom strome sa vrchol naľavo volá ***ľavý nasledovník***, vrchol napravo ***pravý nasledovník***. Napríklad u obrázku 12.4 je vrchol *e* ľavý nasledovník a vrchol *d* pravý nasledovník vrcholu *b*.

Veta 12.2: Strom o n vrcholoch má n-1 hrán.

Dôkaz: Použijeme matematickú indukciu. Keď n=1, strom o jednom vrcholu nemá hranu, veta platí. Induktívny krok: Predpokladajme, že veta platí pre k vrcholov, kde k je kladné celé číslo. Predpokladajme, že strom T má k+1 vrcholov, a že v je list a w je rodič vrcholu v. Odstránením vrcholu v a hrany $\{v,w\}$ z T vytvoríme strom T'. Podľa indukčnej hypotézy strom T' má k -1 hrán. Preto strom T musí mať k hrán, pridaním hrany $\{v,w\}$ zvýšime k -1 o 1. ■

Veta 12.3: Plne m-nárny strom s i vnútornými vrcholmi má $n=m\times i$ +1 vrcholov.

Dôkaz: Každý vrchol, okrem koreňa, je nasledovníkom vnútorného vrcholu. Pretože každý z i interných vrcholov má m nasledovníkov, existujú $m\times i$ vrcholov iných ako koreň. Preto strom obsahuje $n=m\times i$ +1 vrcholov. ■

Koreňový plne m-nárny strom hĺbky h je ***vyvážený*** (balanced), keď všetky jeho listy sú na úrovni h alebo h-1.

Veta 12.4: V m-nárnom strome hĺbky h je maximálne m^h listov.

Dôkaz: Je použitá matematická indukcia na hĺbku. Uvažujte m-árny strom hĺbky 1. Tieto stromy pozostávajú z koreňa s maximálne m nasledovníkmi, kde každý z nich je list. Preto v strome hĺbky 1 je maximálne $m^1=m$ listov.

Teraz predpokladajme, že veta platí pre všetky m-nárne stromy hĺbky menšej ako h. nech T je m-nárny strom hĺbky h. Listy T sú listy podstromov stromu T získaných odstránením hrán incidentných s koreňom. Každý z týchto podstromov má hĺbku maximálne h-1. Takže, podľa induktívnej hypotézy, má maximálne m^{h-1} listov. Pre m podstromov dostávame $m\times m^{h-1}=m^h$ listov. ■

Veta 12.5: Keď m-nárny strom hĺbky h má l listov, potom $h\geq\lceil \log_m l \rceil$. Rovnosť platí pre plne m-nárny vyvážený strom.

Dôkaz: Z vety 12.4 vieme, že $l \leq m^h$. Po zlogaritmovaní pri základe m dostávame $\log_m l \leq h$. Keďže h je celé číslo, platí $h\geq\lceil \log_m l \rceil$. Predpokladajme, že strom je vyvážený. Potom je každý strom v úrovni h alebo h-1, a pretože hĺbka je h, aspoň jeden list je na úrovni h. Potom musí existovať viac ako m^{h-1} listov. Pretože $l \leq m^h$, dostávame $m^{h-1}<l \leq m^h$, po zlogaritmovaní pri základe m dostávame h-1$<\log_m l \leq h$. Preto $h\geq\lceil \log_m l \rceil$.■

Príklad: Predpokladajme, že správu o novom víruse rozpošle každý príjemca o minútu po prijatí 10 kamarátom, a nikto nedostane správu dvakrát. Za ako dlho presiahne počet príjemcov množstvo ľudí na Slovensku? Pri použití vety 12.4 dostávame
$\lceil \mathrm{Log}_{10} 5000000 \rceil$=7 minút.

12.2 Binárne prehľadávacie stromy

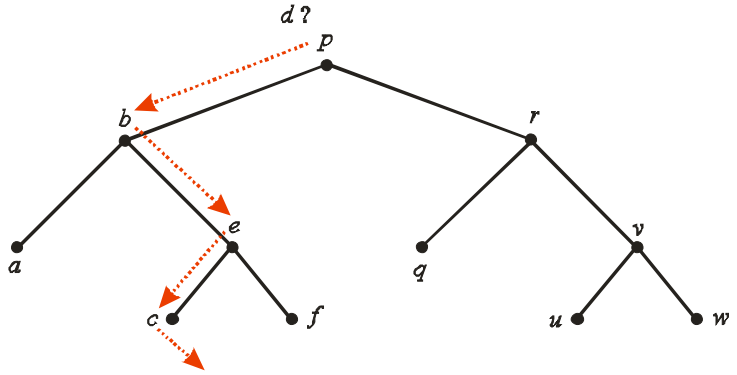
Binárne stromy sú s výhodou využívané na uchovávanie informácie tak, aby mohla byť ľahko nájdená. Vyhľadanie nejakého záznamu je snáď najčastejšie používaná operácia v informatike. Chceli by sme vytvoriť algoritmus, ktorý nám nájde záznam, keď máme záznamy jednoznačne usporiadané (indexované). To sa dá uskutočniť pomocou binárneho prehľadávacieho stromu. Binárny prehľadávací strom je binárny usporiadaný strom, kde každému vrcholu je priradený záznam. Záznam vrcholu musí mať väčší index ako sú indexy záznamov vrcholov v jeho ľavom podstrome a menší index ako majú indexy záznamov v jeho pravom podstrome.

Takýto strom môžeme vytvoriť z ľubovoľného neusporiadaného zoznamu záznamov nasledujúcim algoritmom. Zoberieme si prvý záznam, priradíme ho koreňu. Na pridanie ďalšieho záznamu porovnáme záznam s už umiestnenými záznamami v strome. Začíname od koreňa, ideme doľava keď záznam je menší ako vrcholu už priradený záznam, v opačnom prípade ideme doprava. Keď ideme doľava (doprava) a na danom mieste nie je už žiaden ľavý (pravý) nasledovník, vytvoríme ho a priradíme mu náš zaraďovaný záznam (pozri zaraďovanie záznamu *d* na obr. 12.6).

Na nájdenie záznamu používame prakticky rovnaký prístup. Nasledujúci algoritmus nám dáva návod, ako nájsť záznam v binárnom prehľadávacom strome alebo ho tam zaradiť, keď tam záznam ešte nie je. Na zistenie záznamu priradenému vrcholu *v* budeme používať funkciu *záznam*(*v*), samotná hodnota *v* odkazuje na umiestnenie vrcholu, teda smerník.

```
T:=binárny prehľadávací strom; x:=hľadaný záznam; v:=koreň T;
{vrcholy neexistujúce v T majú hodnotu umiestnenia null, keď
je strom prázdny, aj jeho koreň má hodnotu null}
while v≠null and záznam(v)≠x
begin
  if x<záznam(v) then
  if ľavý nasledovník(v)≠null then v=ľavý_nasledovník(v)
  else pridaj nový_vrchol ako ľavý nasledovník v a v:=null
  else
  if pravý nasledovník(v)≠null then v= pravý_nasledovník(v)
  else pridaj nový_vrchol ako pravý nasledovník v a v:=null
end
if koreň T=null
then pridaj nový_vrchol, t.j. koreň do T a priraď mu záznam x
else
if v=null or záznam(v)≠x
then priraď pre nový_vrchol záznam x a v=nový_vrchol
{v=umiestnenie záznamu x}
```

Aká je náročnosť vyššie uvedenej procedúry? Najväčší počet porovnaní potrebný na pridanie nového záznamu je dĺžka najdlhšej cesty v binárnom prehľadávacom strome z koreňa do listu. Pre *n* záznamov má taký strom *n* vrcholov, a takému stromu môžeme pridať list do $n+1$ pozícií (za porovnanie považujeme aj zistenie, či potomok vrcholu existuje). Použitím vety 12.5 dostávame, že výška stromu je $\lceil \log_2(n+1) \rceil$ pre plný vyvážený strom. Z toho vyplýva, že pre nájdenie alebo pridanie prvku do vyváženého binárneho stromu potrebujeme maximálne $\lceil \log_2(n+1) \rceil$ porovnaní. Keďže toto je najlepší možný výsledok pre binárne stromy, snažíme sa udržať záznamy vo forme vyváženého stromu, na čo existujú špecializované algoritmy.

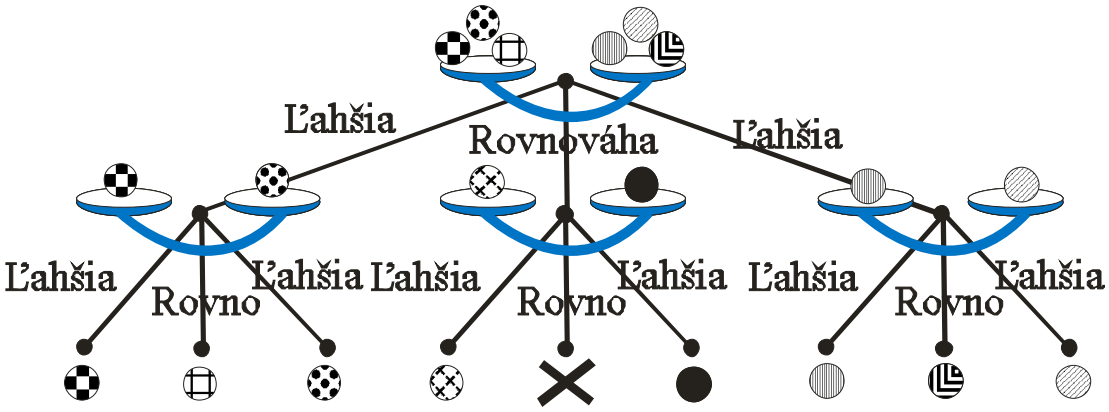


Obrázok 12.6. Priraďovanie záznamu *d* do binárneho prehľadávacieho stromu s vrcholmi ohodnotenými písmenami abecedy predstavujúcimi záznamy.

12.3 Rozhodovacie stromy

Koreňové stromy môžu byť použité na modelovanie problémov, v ktorých rad diskrétnych rozhodnutí vedie k riešeniu. Koreňový strom, v ktorom každý vnútorný vrchol odpovedá rozhodnutiu, s podstromami týchto vrcholov pre všetky možné výsledky tohto rozhodnutia, sa volá rozhodovací strom. Možné riešenia problému odpovedajú cestám k listom tohto koreňového stromu. To je ilustrované napr. na obr. 12.7.

Príklad: Máme 8 mincí, z ktorých je 1 falošná, ľahšia. Ako ju rozoznať čo najmenším počtom vážení rovnoramenných váh?



Obrázok 12.7. Rozhodovací strom pre rozpoznanie ľahšej falošnej mince spomedzi 8 mincí. Mince sú odlíšené farbou, a umiestnené na miskách rovnoramenných váh. Postupné označenie u hrán znamená, že ľavá miska váh je ľahšia, resp. že obidve misky sú v rovnováhe, popr. že pravá miska váh je ľahšia. Kotúčiky v dolnom riadku sú určené falošné mince.

Riešenie: Pre každé váženie na rovnoramenných váhach sú možné 3 výsledky: prvá miska bude ľahšia, obidve budú v rovnováhe, pravá miska bude ľahšia (pozri obr. 12.7). Rozhodovací strom pre sériu vážení je teda ternárnym stromom. Strom musí mať aspoň 8 listov, pretože je 8 možných riešení – každá z mincí môže byť ľahšia. Najväčší počet vážení potrebný na určenie falošnej mince je hĺbka rozhodovacieho stromu. Z vety 12.5 vyplýva, že táto hĺbka je najmenej $\lceil \log_3 8 \rceil = 2$. Sú teda potrebné aspoň dve váženia. Rozhodovací strom,

ktorý ukazuje, že sa problém fakticky dá riešiť dvoma váženiami, je na obr. 12.7. Ako na takýto rozhodovací strom dojsť je už iná otázka.

Pomocou modelu rozhodovacieho stromu sa dá riešiť aj zložitosť algoritmov na triedenie. Dá sa takto nájsť napríklad najmenší počet porovnaní pre najhorší prípad postupnosti na usporiadanie pre taký algoritmus.

Veta 12.6. Každý triediaci algoritmus vyžaduje na triedenie n záznamov v najhoršom prípade[2] aspoň $\lceil \log_2(n!) \rceil$ porovnaní dvoch prvkov, čo odpovedá zložitosti[3] $\Omega(n \log n)$.

Dôkaz: Algoritmus na triedenie záznamov zoberie vo všeobecnosti sekvenciu záznamov k_1, k_2, ... k_n, a vráti permutáciu ich indexov, ktorá odpovedá utriedenej sekvencii. Pre dané n si môžeme predstaviť optimálny algoritmus ako binárny rozhodovací strom: interné vrcholy odpovedajú porovnaniu dvoch záznamov, a listy odpovedajú korektným permutáciám. Triedenie danej sekvencie odpovedá ceste po strome od koreňa k listu. Počet porovnaní je prinajhoršom rovný hĺbke stromu.

Daná sekvencia má $n!$ permutácií, takže strom má $n!$ listov. Preto musí byť hĺbka stromu podľa vety 12.5 rovná najmenej $\lceil \log_2(n!) \rceil$. Podľa Stirlingovej formuly platí, že $n! \geq n^n e^{-n}$, čo je ekvivalentné k $\log_2 (n!) \geq n \log_2 n - n \log_2 e$. Takže pre najhorší prípad je zložitosti $\geq \lceil \log_2(n!) \rceil \geq n \log_2 n - n \log_2 e = \Omega (n \log n)$. ■

12.4 Prefixové kódovanie

Postavme si problém, ako zapísať viac informácií pomocou menšieho počtu bitov? Odpoveďou je priradiť znakom alebo vstupným informáciám, ktoré sa opakujú najčastejšie (pravdepodobnosť výskytu), čo najkratší kód. Taký princíp môžeme použiť tak pri stratovom kódovaní (ako je MP3, kedy znesieme malý šum výmenou za kratší súbor), tak i pri nestratovom kódovaní (ako je winzip či rar kód, kedy by sa nám nepáčilo, aby zmizli písmenká zo skomprimovanej diplomovej práce).

Kódovanie si ukážme na príklade nestratového kódovania, kedy máme pomocou reťazcov bitov zakódovať písmena anglickej abecedy a nebudeme rozlišovať medzi malými a veľkými písmenami. Normálny ASCII kód je 8 bitový, ale keďže sa budeme baviť iba o 26 znakoch, na ich rozlíšenie by nám stačilo 5 bitov, pretože $16=2^4<26<2^5=32$. Keď ale zoberieme do úvahy frekvenciu výskytu jednotlivých písmen, môžeme pomocou lepšieho zakódovania priemernú správu odvysielať menej bitmi, aj keď niektoré písmenká pritom môžu byť zakódované aj dlhším reťazcom bitov ako je 5 bitový. Najčastejšie písmenká budú ale zakódované najkratším reťazcom bitov.

Aké značky máme priradiť jednotlivým znakom? Treba dať pozor. Predstavme si správu AAABBC. Čo keby sme priradili značky zle? Napríklad znaku A by sme dali reťazec 0 a znaku B reťazec 1, a znaku C reťazec 01. Postupnosť bitov 0001101 braná zaradom by

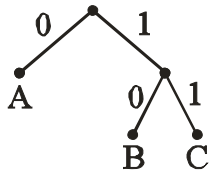
[2] Najhorším prípadom sa mieni také usporiadanie vstupných dát, na ktorom algoritmus zle funguje. Najlepší prípad by bola permutácia odpovedajúca už usporiadaným záznamom.

[3] Veľké grécke písmeno omega sa používa ako mierka zložitosti algoritmu, zvyčajne v závislosti na potrebnom čase alebo pamäti, pre veľkosť problému n, čo je zvyčajne počet prvkov. Zložitosť algoritmu často závisí nie len na veľkosti, alebo množstve dát, ale priamo na ich hodnotách. Používa sa teda viac značení: O(f(n)) – Omikron notácia – horný odhad alebo horšie už to nebude, časové nároky algoritmu nikdy nebudú rásť rýchlejšie ako $f(n)$; $\Theta(f(n))$ – Theta notácia – priemerný odhad; $\Omega(f(n))$ – Omega notácia – dolný odhad alebo lepšie už to nebude. Neformálne, platnosť $f(n) \in \Omega(g(n))$ znamená, že hodnota f (počet porovnaní) je väčšia ako $g(n)$ vynásobené nejakou konštantou. Formálne, $f(n) \in \Omega (g(n))$ znamená, že existujú kladné konštanty c a k, také že $0 \leq cg(n) \leq f(n)$ pre všetky $n \geq k$. Hodnoty c a k musia byť konštantné pre funkciu f a nesmú závisieť od n.

bola preložená ako AAABBAB, ale my sme tak zakódovali AAABBC. Dá sa v tomto prípade vôbec zakódovať AAABBC?

Jedným zo spôsobov, ako zaručiť, že žiaden binárny reťazec nezodpovedá viac ako jednej sekvencii písmen je, že žiaden reťazec odpovedajúci jednému písmenu sa neobjaví ako počiatočná časť reťazca odpovedajúceho inému písmenu. Kód s takouto vlastnosťou sa volá prefixový kód.

Prefixový kód môže byť reprezentovaný binárnym stromom, kde písmená sú listy stromu. Hrany stromu sú označené tak, že hrana idúca k ľavému následníkovi je označená 0 a hrana idúca k pravému následníkovi je označená 1. Binárny reťazec odpovedajúci písmenu je sekvencia označení hrán na ceste od koreňa k danému písmenu. Strom reprezentujúci kód môže byť použitý na dekódovanie binárneho reťazca. Napríklad pre strom na obr. 12.8 dostávame pre správu AAABBC binárny reťazec 000101011.



Obrázok 12.8. Binárny strom odpovedajúci prefixovému kódu pre prvé tri písmená abecedy.

Pri dekódovaní postupujeme nasledovne. Začneme v koreni, zoberieme prvý bit, je to 0, ideme teda doľava, je to koniec tejto vetvy ? Áno je, žiadne ďalšie z nej už nevedú. Teda napíšeme si písmeno A. Znovu začneme v koreni, zoberieme bit, je to 0, teda opäť si zapíšeme A. Znovu ideme do koreňa, zoberieme ďalší bit, je to 0 teda opäť si zapíšeme A. Znovu ideme do koreňa, zoberieme ďalší bit, je to 1, ideme z koreňa doprava, je tam niečo uložené ? Nie, nie je, ďalej sa to vetví, tak zoberieme ďalší bit, aby sme videli, ako sa to vetví. Ďalší bit je 0, teda ideme doľava, tam narazíme na B. Znova zopakujeme od koreňa. A takto postupujeme, kým máme nejaké bity na rozpakovanie.

Prefixový kód môžeme zostrojiť pre akýkoľvek binárny strom s hranami idúcimi doľava označenými jednou binárnou hodnotou, s hranami idúcimi doprava komplementárnou binárnou hodnotou a s listami stotožnenými so znakmi.

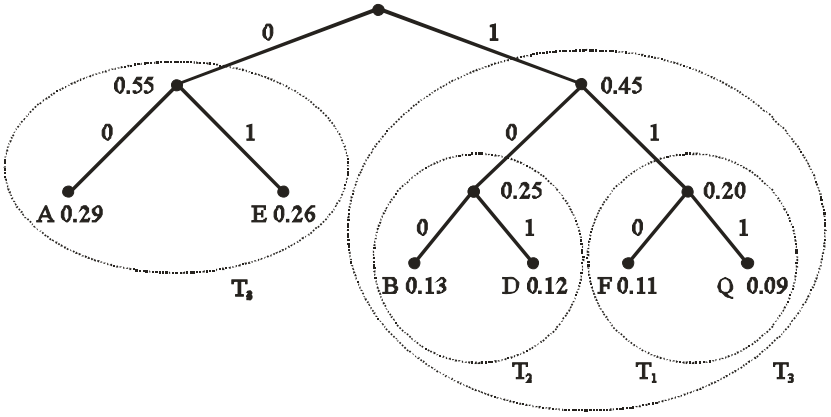
David Huffman (1925-1999) ako študent na MIT r. 1951 opísal v semestrálnej práci algoritmus, známy od tej doby ako ***Huffmanove kódovanie***. Tento algoritmus má ako vstup frekvencie (pravdepodobnosť výskytu) znakov v správe a vytvára prefixový kód, ktorý kóduje správu najmenším možným počtom bitov spomedzi všetkých možných binárnych prefixových kódov pre danú sadu symbolov.

Algoritmus Huffmanovho kódovania má za cieľ vytvoriť binárny koreňový strom so znakmi ako s listami. Algoritmus začína s lesom jednovrcholových stromov označených písmenami abecedy a s váhami stromov rovnými frekvencii priradeného písmena. Pri každom kroku sa kombinujú dva stromy s najmenšou celkovou váhou do jedného stromu pridaním nového koreňa a umiestnením stromu s väčšou váhou ako ľavého podstromu a stromu s menšou váhou ako pravého podstromu. Takto vytvorenému stromu priradíme váhu rovnú súčtu váh obidvoch stromov. Algoritmus končí, keď je pôvodný les pospájaný do jedného stromu.

```
F:=les z n koreňových jednovrcholových stromov s priradenými
symbolmi abecedy a_i a váhami w_i odpovedajúcimi frekvenciám;
while F nie je strom
begin
  nahraď koreňové stromy T a T' s najnižšími váhami z F s
```

```
  w(T)≥w(T′) stromom s novým koreňom a s T a T′ ako ľavým,
  resp. pravým podstromom. Hranu k T označ 0, hranu k  T′
  označ 1.
  Priraď novému stromu váhu w(T)+w(T′)
```
end

Príklad: Majme nasledujúce symboly s frekvenciami A: 0.29, E: 0.26, B: 0.13, D: 0.12, F: 0.11, Q: 0.09. Najprv spojíme F a Q do stromu T_1 s váhou (u koreňa) 0.20, potom B a D do stromu T_2 s váhou 0.25, potom stromy T_1 a T_2 do stromu T_3 s váhou 0.45, potom A a E do stromu T_4 s váhou 0.55, a nakoniec stromy T_3 a T_4 do výsledného stromu (pozri obr. 12.9).

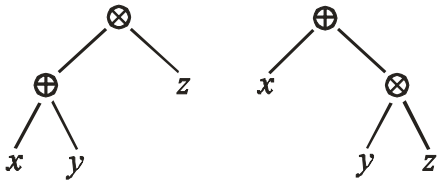


Obrázok 12.9. Binárny koreňový strom pre Huffmanov kód

Pre Huffmanove kódovanie existuje veľa variant, napríklad namiesto jednotlivých symbolov môžeme kódovať dvojice symbolov, alebo obecne *n*-tice symbolov.

12.5 Koreňové stromy reprezentujúce algebraické výrazy

Majme množinu *S*, na ktorej sú definované dve binárne operácie, $\oplus$ a $\otimes$. Bez toho, aby sme vedeli, ktorá z nich má prednosť, tak môžeme napríklad výraz $x\oplus y\otimes z$ chápať ako $(x\oplus y)\otimes z$ alebo $x\oplus(y\otimes z)$. Také výrazy môžeme jednoznačne reprezentovať binárnym stromom. Výraz môže byť rekurzívne definovaný ako $X\alpha Y$, kde α je symbol pre binárnu operáciu a X a Y sú buď prvky z množiny *S* alebo výraz. Taký výraz môže byť reprezentovaný binárnym stromom s koreňom α, ľavým podstromom X a pravým podstromom Y. Hore uvedené výrazy potom môžu byť reprezentované stromami ako na obr. 12.10.



Obrázok 12.10. Binárne stromy pre výrazy $(x\oplus y)\otimes z$ a $x\oplus(y\otimes z)$.

Bežný zápis výrazu, napr. $x\oplus y\otimes z$ sa volá ***infixová notácia***. Výraz môžeme rovnako dobre rekurzívne definovať v tzv. ***prefixovej*** (alebo ***poľskej***) ***notácii*** ako αXY čo by namiesto výrazu $(x\oplus y)\otimes z$ dalo výraz $\otimes\oplus xyz$ a namiesto výrazu $x\oplus(y\otimes z)$ výraz $\oplus x\otimes yz$. Tento zápis môže vyzerať mätúco, ale je jednoznačnejší v tom zmysle, že na rozdiel od bežného infixového zápisu nepotrebujeme poznať precedenciu operátorov alebo ju mať určenú zátvorkami. Okrem tejto prefixovej notácie existuje aj ***postfixová (reverzná poľská) notácia***,

kedy je symbol pre operáciu písaný za prvkami, ktoré spája, ako u $XY\alpha$ čo by namiesto výrazu $(x \oplus y) \otimes z$ dalo výraz $xy \oplus z \otimes$ a namiesto výrazu $x \oplus (y \otimes z)$ výraz $xyz \otimes \oplus$. Voľne povedané, výraz v prefixovej notácii sa z odpovedajúceho binárneho stromu urobí tak, že keď ideme po vrcholoch grafu, ako by sme ho prehľadávali do hĺbky, zapíšeme operátor vždy prvý krát, keď vrcholom prechádzame, rovnako ako pre premennú pri liste, u infixovej notácie to urobíme pri druhom prechode vrcholom, u postfixovej notácie pri treťom priechode vrcholom.

Koreňové stromy vyjadrujúce algebraické výrazy sa používajú napríklad pri symbolickej regresii u genetického programovania.

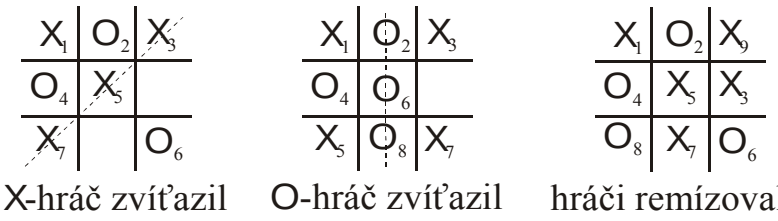
12.6 Koreňový strom ako model hry

Stromy môžu byť použité na analýzu určitého typu hier ako sú piškvorky (angl. tic-tac-toe), dáma, šach alebo odoberanie zápaliek (anglický ekvivalent hra nim).V týchto hrách sa dvaja hráči postupne striedajú v ťahoch. Každý hráč vie, aký ťah urobil jeho protihráč a do hry nevstupuje prvok náhodnosti. Také hry sa dajú modelovať pomocou stromu hry. Vrcholy takého stromu reprezentujú momentálne pozície hry a hrany reprezentujú legálne ťahy medzi pozíciami. Vrcholy reprezentujúce rovnaké pozície sa v takom strome môžu nachádzať viackrát, pokiaľ k nim vedú rozdielne postupnosti ťahov. Koreň reprezentuje štartovnú pozíciu. Listy reprezentujú koncové pozície hry. Listy môžeme ohodnotiť 1, keď odpovedajú výhre prvého hráča, 0 keď nastala remíza, a -1, keď odpovedajú prehre prvého hráča.

Tu si uvedieme hru volanú ***piškvorky***. Táto hra vyžaduje dvoch hráčov, prvý hráč je označený symbolom X a druhý hráč symbolom O. Snahou každého hráča je umiestniť na štvorcovej 3×3 hracej doske svoje symboly tak, aby tvorili buď riadok, stĺpec, alebo uhlopriečku (obr. 12.11).

Hra je zahájená hráčom X, potom nasleduje ťah hráča O. Toto striedanie hráčov sa opakuje tak dlho, až jeden z hráčov vyhral, alebo je na hracej doske umiestnených deväť symbolov.

Priebehy hier môžeme reprezentovať pomocou „stromu riešení“, kde je explicitne ukázané, ktoré ťahy boli použité (obr. 12.12).



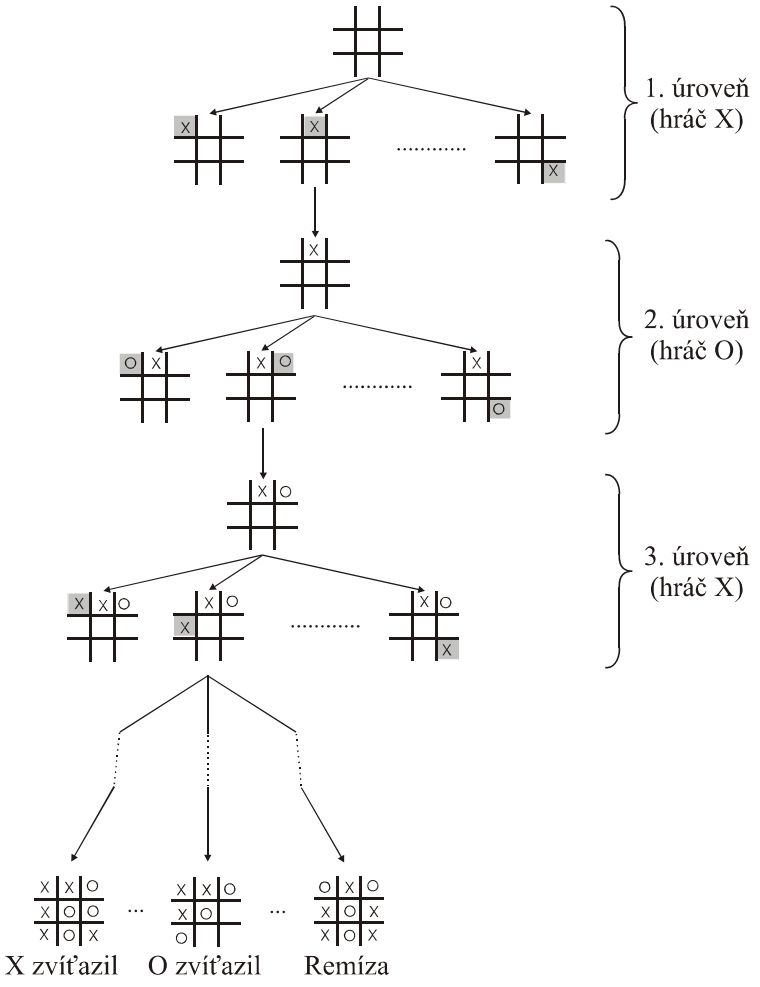
Obrázok 12.11. Znázornenie 3 partií hry piškvorky. Indexy pri jednotlivých symboloch X a/alebo O znamenajú poradie ťahu. V prvých dvoch partiách bolo dosiahnuté víťazstvo, hráč umiestnil svoje symboly do riadku, stĺpca alebo diagonály, čo je naznačené prerušovanou čiarou. V tretej partii sa žiadnemu hráčovi takto nepodarilo umiestniť svoje symboly, po 9 ťahoch, keď sú obsadené všetky pozície hracej dosky, hra končí remízou.

Hra piškvorky patrí medzi tzv. symetrické hry, z pohľadu druhého hráča je hra identická s hrou prvého hráča (hráči sú rovnocenní, odlišujú sa len v tom, že jeden z nich zahajuje hru). Obaja hráči riešia rovnaký strategický cieľ (maximalizujú svoj zisk) a súčasne zabrániť v tejto akcii súperovi (minimalizujú súperov zisk).

Vo všeobecnosti existujú dva diametrálne odlišné prístupy k riešeniu problému, ako vyhrať:

Prvý prístup je založený na existencii modelu hry, ktorý obsahuje ***hierarchicky usporiadané pravidlá***. Keď tieto pravidlá budeme dodržiavať, mali by sme dospieť ak nie víťaznej pozícii, tak aspoň k remíze. Žiaľ, tento model je zostrojiteľný len pre jednoduché hry, pre zložitejšie hry (napr. šach) už nie sme schopní ho zostrojiť s dostatočnou presnosťou a efektivitou. Obvykle sme v zložitých prípadoch schopní formulovať len rôzne všeobecné pravidlá (heuristiky), ktorých dodržiavanie v priebehu hry by malo viesť k víťazstvu.

Druhý prístup je založený na rozsiahlom ***prehľadávaní stromu riešení***, pomocou ktorého, aspoň teoreticky, sme schopní nájsť optimálny ťah v každej etape hry. Aj keď je tento prístup koncepčne veľmi jednoduchý, jeho numerická realizácia v počítači naráža na vážne problémy s enormnou veľkosťou stromu riešení. Stromy hier môžu byť obrovské, napr. pre šach je odhadovaný na 10^{100} vrcholov. K numerickému zvládnutiu tohto prístupu musíme zaviesť niektoré podstatné zjednodušenia týkajúce sa hĺbky prehľadávania stromu riešení (tak napr. pri prehľadávaní stromu riešení ideme do maximálnej hĺbky 3 alebo 4). Toto zjednodušenie prehľadávania stromu riešení je obvykle kombinované s rôznymi heuristikami, ktoré ohodnocujú „koncové“ stavy hry.



Obrázok 12.12. Znázornenie stromu riešení, ktorého vrchol (koreň) je prázdna hracia doska. Po prvom ťahu, ktorý hral hráč X, je obsadená jedna pozícia (zdôraznená vytieňovaným) symbolom X. V druhej úrovni, hranej hráčom O, je obsadená pozícia (vytieňovaná) symbolom O. Vetva stromu riešení končí vtedy, ak jeden hráč zvíťazil alebo sú obsadené všetky pozície na hracej doske - hra skončila remízou.

Jeden zo základných princípov umelej inteligencie je návrh modelu študovaného problému. Môžeme napríklad pomocou súboru pravidiel byť schopný hrať piškvorky na

dobrej úrovni. Uvedieme jednoduchý model[4], ktorý sa zakladá na tom, že ťah hráča je určený nasledujúcimi šiestimi pravidlami s klesajúcou prioritou (obr. 12.13-12.15):

Pravidlo 1. *Hráč vykoná ťah, ktorý vedie k jeho víťazstvu.*

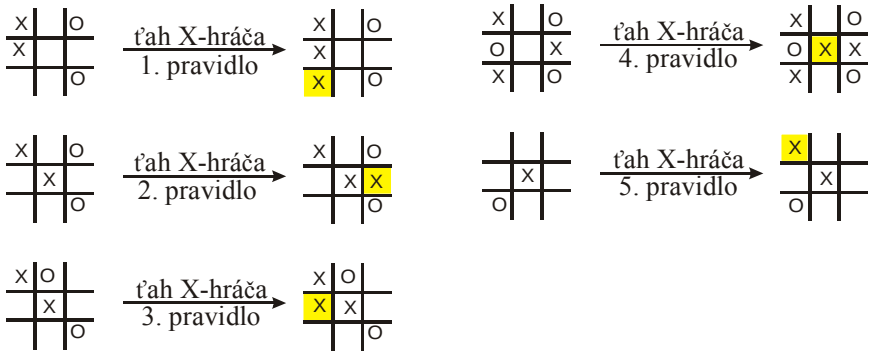
Pravidlo 2. *Hráč vykoná ťah, ktorý zabráni víťazstvu oponenta v nasledujúcom ťahu.*

Pravidlo 3. *Hráč vykoná ťah, ktorý pripraví možnosť dvojnásobného použitia 1. pravidla v nasledujúcom ťahu (tzv. vidlička, obr. 12.14).*

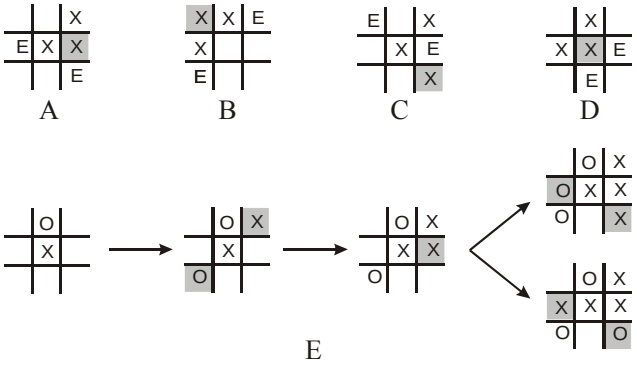
Pravidlo 4. *Hráč vykoná ťah, ktorým obsadí stredné pole.*

Pravidlo 5. *Hráč vykoná ťah, ktorým obsadí rohové pole.*

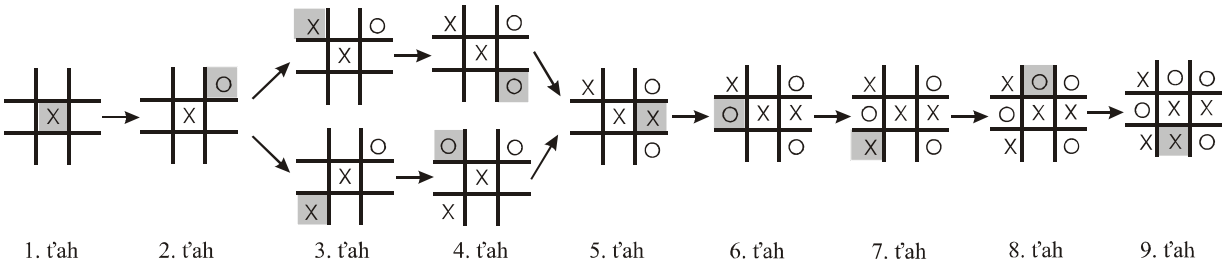
Pravidlo 6. *Hráč vykoná náhodný ťah.*



Obrázok 12.13. Diagramy ilustrujú jednotlivé pravidlá modelu hry. Poznamenajme, že v dôsledku 4. pravidla je prvým ťahom vždy obsadenie prostredného poľa.



Obrázok 12.14. Diagramy A-D znázorňujú základné typy vidličkových pozícií, ktoré sú aplikovateľné použitím pravidla 3. Symboly E znázorňujú príklady dvojice pozícií, ktoré musia byť prázdne, aby sa dala „vidlička“ v ďalšom ťahu doplniť na víťaznú trojicu. Diagram E ukazuje pozíciu, ktorá je prehraná pre hráča O už po jeho druhom ťahu. Pravidlá nášho modelu hry používa prvý hráč X.



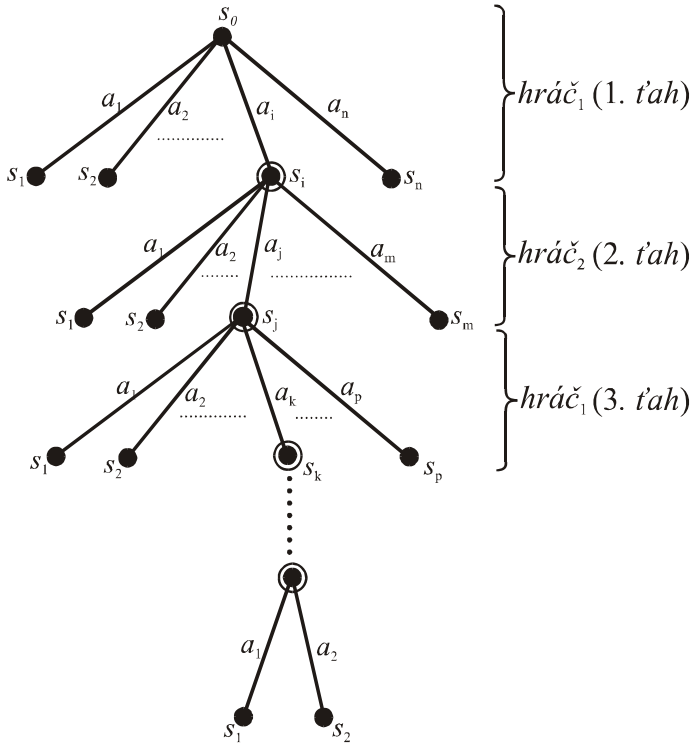
[4] Tomuto modelu chýba jedno pravidlo, aby vždy vyhral alebo aspoň remizoval.

Obrázok 12.15. Znázornenie priebehu hry pomocou modelu, hra skončila remízou. Spodná „vetva“ ukazuje, že rôznymi postupnosťami ťahov sa môžeme dostať na rovnaký stav, prípadne na stav ekvivalentný z hľadiska symetrie. Diagram, kde sa vetvi spájajú, ale už neodpovedá stromu.

Úvah o priebehu hry piškvorky môžeme formalizovať nasledovne: Rozloženie znakov X a O na hracej doske sa nazýva ***stav hry***.

Ťah hráča, t.j. pridanie znaku X alebo O na hraciu dosky do konkrétnej volnej polohy, sa nazýva ***akcia***. Pre daný stav s má hráč k dispozícii ***množinu prípustných akcií*** $\mathcal{A}(s)=\{a_1, a_2,..\}$. Hovoríme, že akcia a ***transformuje*** stav s na nový stav s' , alebo $s' = a(s)$. Množina všetkých možných stavov $\mathcal{S}=\{s_1, s_2, \ldots\}$ sa nazýva ***stavový priestor*** hry. Pomocou akcií sa môžeme ***pohybovať v stavovom priestore*** (obr. 12.16). Nech s_0 je počiatočný stav odpovedajúci prázdnej hracej doske, prvý hráč na vybranú pozíciu priloží znak X, dostaneme stav s_1, potom druhý hráč na stav s_1 vykoná akciu (priloží na voľnú pozíciu znak O), takto dostaneme stav s_2, atď. Priebeh hry je popísaný sekvenciou stavov

$$s_0 \xrightarrow{a_1} s_1 \xrightarrow{a_2} s_2 \ldots\ldots \xrightarrow{a_k} s_k \Rightarrow (s_0 s_1 s_2 \ldots s_k)$$



Obrázok 12.16. Prvý hráč vytvorí z počiatočnej pozície (vrchol stromu) všetky možné nasledujúce (1-ťahové) pozície s_1, s_2,…. Na základe určitých (racionálnych) úvah vyberie pozíciu s_i. Druhý hráč z pozície s_i vytvorí nové (2-ťahové) pozície, z nich vyberie pozíciu s_j ako výsledok svojho ťahu – akcie. Obaja hráči tento postup opakujú, končí sa vtedy, ak niektorý hráč vyhral alebo obaja hráči remizovali.

Veľkosť stavového priestoru (t.j. počtom rôznych stavov - pozícií hry) je určená hornou hranicou 3^9 = 19,683. (sú tri stavy každej pozície dosky a deväť pozícií). Tento počet ale zahŕňa veľa nepovolených stavov, ako samé krížiky a žiadne krúžky, alebo pozíciu, kde má jeden hráč riadok alebo stĺpec troch krížikov a druhý hráč krúžkov. Odstránenie nepovolených stavov redukuje priestor na 5478 stavov a keď započítame stavy symetrické operáciou rotácie a zrkadlenia ako jeden stav, existuje len 765 odlišných stavov.

Použitím metódy spätného prehľadávania môžeme zostrojiť kompletný strom riešení hry piškvorky. Zistili sme, že má nasledujúci počet koncových pozícií, ktoré sú charakterizované takto

Počet	Typ
131184	víťazstvo hráča X
77904	víťazstvo hráča O
46080	remíza hráčov X a O
255168	celkový počet

Ďalší dôležitý problém pred ktorým teraz stojíme (predpokladáme, že už poznáme úplný strom riešení hry piškvorky) je zistiť víťaznú stratégiu pre prvého hráča. Existujú tieto tri možnosti:

(1) ***existuje víťazná stratégia pre prvého hráča***, to znamená, že pri použití tejto stratégie prvým hráčom musí druhý hráč prehrať,

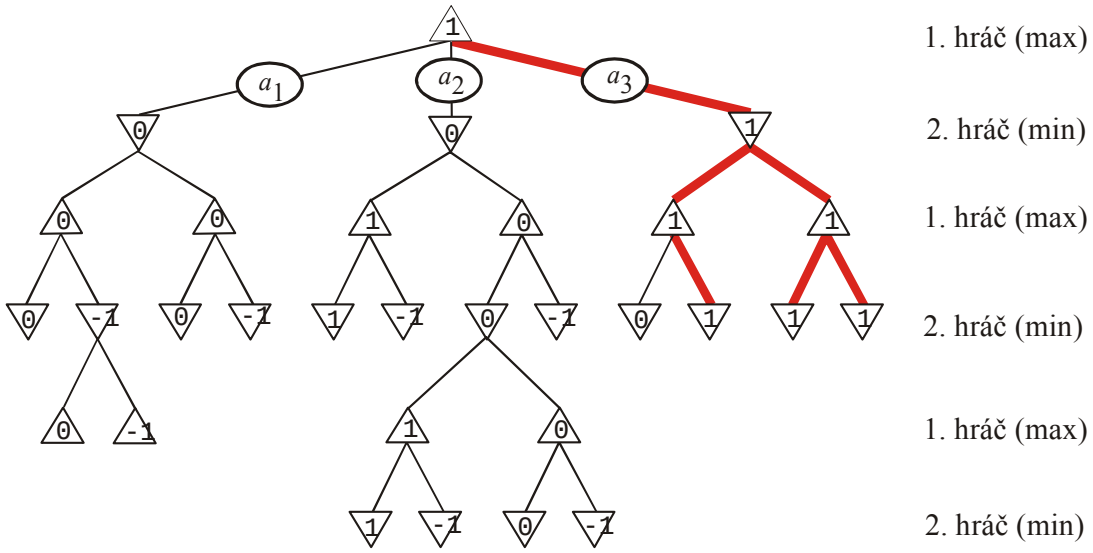
(2) ***existuje víťazná stratégia pre druhého hráča***, to znamená, že pri použití tejto stratégie druhým hráčom musí prvý hráč prehrať,

(3) ***neexistujú víťazné stratégie***, optimálna stratégia poskytuje obom hráčom len remízu.

Jednoduchá modifikácia metódy spätného hľadania pre konštrukciu stromu riešení nám umožňuje zistiť optimálnu stratégiu pre oboch hráčov. Najprv si ohodnotíme koncové vrcholy – listy tak, že pokiaľ vyhral prvý hráč, sú ohodnotené 1, pokiaľ vyhral druhý hráč, sú ohodnotené -1 a pokiaľ hra skončila remízou, vrchol je ohodnotený 0. Základný princíp pre získanie optimálnej stratégie je veľmi jednoduchý, vychádza zo skutočnosti, že každý hráč volí taký ťah, aby maximalizoval minimálnu hodnotu pozície vychádzajúcu z oponentových ďalších možných ťahov. Hráč maximalizuje svoj zisk (=ohodnotenie svojej nasledujúcej pozície smerom od koreňa k listom, teda ľavého či pravého nasledovníka) a tým minimalizoval zisk súpera, preto ho nazývame „*minimax princíp*“ (obr. 12.17).

Prvý hráč (označený max) chce vyhrať, to znamená, že bude vyberať také ťahy, aby maximalizoval svoj zisk a minimalizoval zisk súpera (druhého hráča, označeného min).

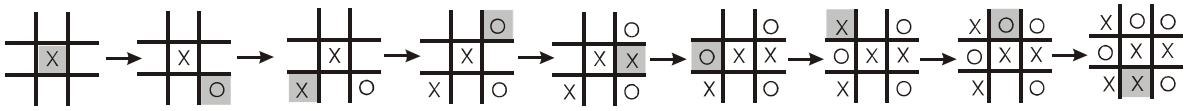
Druhý hráč sa bude správať podobne, ako prvý hráč, chce maximalizovať svoju zisk, čo je ekvivalentné tomu, že minimalizuje zisk prvého hráča (preto je druhý hráč označený min). Pri ohodnotení všetkých zvyšných vrcholov stromu riešení postupujeme po úrovniach od listov ku koreňu a postupne ohodnocujeme vnútorné vrcholy. Vnútorný vrchol prehľadávaného grafu je vždy označený maximom (pre 1. hráča) resp. minimom (pre 2. hráča) hodnôt jeho ľavého a pravého nasledovníka. Hráči sa pritom striedajú po úrovniach, definovaných grafovou vzdialenosťou od koreňa stromu. Výsledok na obr. 12.17 ukazuje, že pre danú hru existuje víťazná stratégia pre prvého hráča, pretože koreň má hodnotu 1. Zvýraznené hrany označujú ťahy vedúce k výhre.



Obrázok 12.17. Strom riešení a minimax princíp

Pomocou tohto jednoduchého postupu založeného na minimaxovom princípe môžeme riešiť veľkú triedu hier s dvoma hráčmi, ktorí striedavo vykonávajú ťahy, pričom hlavným strategickým zámerom každého hráča je maximalizovať svoj zisk (t.j. minimalizovať zisk súpera). Hlavný problém s použitím tohto algoritmu spočíva v tom, že strom riešení pre zložitejšie hry (napr. šach) má enormnú veľkosť, takže systematické prechádzanie po všetkých jeho vrcholoch- stavoch je nerealizovateľné.

Ak aplikujeme metódu spätného hľadania kombinovanú s minimax princípom na piškvorky, potom zistíme, že keď obidvaja hráči hrajú zo svojho hľadiska optimálne, môžu dosiahnuť len remízu. To znamená, že vrchol stromu riešení je ohodnotený na záver 0, jedna z týchto optimálnych partií je postupnosť ťahov reprezentovaná „permutáciou“ (597364128).



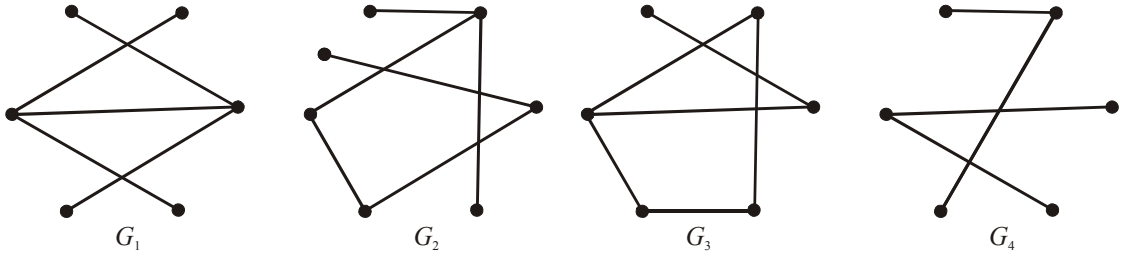
Obrázok 12.18. Znázornenie postupnosti (597364128) pomocou jednotlivých ťahov hry piškvorky. Jednoducho sa môže skontrolovať, že táto postupnosť ťahov vyhovuje modelu hry piškvorky, ktorý bol diskutovaný v prvej časti tejto kapitoly.

Z výsledkov získaných metódou spätného hľadania spolu s „minimax“ princípom vyplýva záver, že optimálna stratégia hry piškvorky vedie k remíze, neexistuje taká stratégia, pomocou ktorej by jeden hráč vyhral a druhý prehral. Tým, že metóda prekontrolovala celý strom riešení, môžeme tento výsledok pokladať za konečný a nemenný.

Rovnako si môžeme pre dokonalejší model hry určiť počítačovými simulačnými výpočtami, že optimálna stratégia hry piškvorky sa pre prvého hráča riadi pomocou modelu hry s hierarchickými pravidlami. Skontrolovali sme optimálne postupnosti ťahov dĺžky 9, zistili sme, že vo všetkých prípadoch model hry je splnený.

Cvičenia

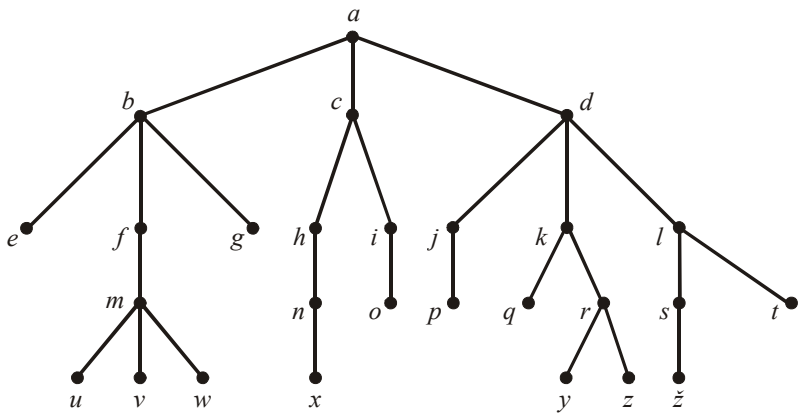
Cvičenie 12.1. Ktoré z nasledujúcich grafov na obr. 12.C1 nie sú stromy a prečo?



Obrázok 12.C1. Ktoré sú stromy?

Cvičenie 12.2. Odpovedzte pre graf na obr. 12.C2 nasledujúce otázky:

a) Ktorý z vrcholov je koreň?
b) Ktoré vrcholy sú vnútorné?
c) Ktoré vrcholy sú listy?
d) Ktoré vrcholy sú nasledovníci (synovia) vrcholu k?
e) Ktoré vrcholy sú rodičia vrcholu k?
f) Ktoré vrcholy sú predkovia k?
g) Ktoré vrcholy sú potomkovia vrcholu k?



Obrázok 12.C2. Koreňový strom.

Cvičenie 12.3. Koľko neizomorfných podstromov do 5 vrcholov obsahuje graf na obr. 12.C2?

Cvičenie 12.4. Majme n prirodzených čísel s_1, s_2, s_3, ..., s_n, kde $n \geq 2$. Nutná a postačujúca podmienka, aby existoval strom na n uzloch taký, že s_1, s_2, s_3, ..., s_n, sú po poriadku stupne jeho vrcholov, je

$$\sum_{i=1}^{n} s_i = 2n - 2$$

Dokážte.

Cvičenie 12.5. Nech G je jednoduchý graf o n vrcholoch. Ukážte, že G je strom vtedy a len vtedy, keď je súvislý a má $n - 1$ hrán.

Cvičenie 12.6. Predpokladajme, že 1024 ľudí sa účastní šachového turnaja. Použite koreňový strom ako model turnaja na určenie, koľko hier musí byť odohraných, aby sa určil víťaz, pokiaľ je hráč eliminovaný po jednej prehre a turnaj pokračuje, dokiaľ iba jeden účastník neprehral. Predpokladáme, že nebudú žiadne remízy.

Cvičenie 12.7. Reťazový list začína človekom posielajúcim list desiatim ďalším ľuďom. Každý príjemca je požiadaný, aby poslal list ďalším desiatim, a každý list obsahuje zoznam predchádzajúcich šiestich ľudí v reťazci. Pokiaľ zoznam neobsahuje menej ako šesť mien, každý príjemca pošle dvadsať korún prvému človeku v zozname, odstráni jeho meno zo zoznamu, a pridá svoje vlastné meno na koniec zoznamu. Keď všetci takto odpovedia na list a nikto nedostane viac ako jeden list, koľko peňazí človek zapojený do reťazca nakoniec dostane?

Cvičenie 12.8. Koľko rôznych izomérov majú nasledujúce nasýtené uhľovodíky?

(a) C_3H_8
(b) C_5H_{12}
(c) C_6H_{14}

Cvičenie 12.9. Ukážte, ako môže byť 16 čísel sčítaných pomocou 15 procesorov v priebehu 4 časových krokov potrebných na sčítanie dvojice čísel (vstup a prenos informácie neuvažujeme za časovo náročné kroky a ich čas zanedbávame v porovnaní so sčítaním).

Cvičenie 12.10. Nech n je mocnina dvoch. Ukážte, že n čísel môže byť sčítané v $\log_2 n$ krokoch za použitia siete so stromovou štruktúrou o n -1 procesoroch.

Cvičenie 12.11. Koľko vážení na rovnoramenných váhach je potrebné na nájdenie ľahšej falošnej mince spomedzi štyroch mincí? Popíšte algoritmus na nájdenie tejto ľahšej mince za použitia tohto počtu vážení.

Cvičenie 12.12. Koľko vážení na rovnoramenných váhach je potrebné na nájdenie falošnej mince spomedzi štyroch mincí, ktorá môže byť ľahšia alebo ťažšia ako ostatné tri?

Cvičenie 12.13. Koľko vážení na rovnoramenných váhach je potrebné na nájdenie spomedzi 12 mincí falošnej mince, ktorá je ľahšia ako ostatné?

Cvičenie 12.14. Ktorý z nasledujúcich kódov je prefixový kód?

(a) *a*: 11, *e*: 00, *t*: 10, *s*: 01
(b) *a*: 0, *e*: 1, *t*: 01, *s*: 001
(c) *a*: 101, *e*: 11, *t*: 001, *s*: 011, *n*: 010
(d) *a*: 010, *e*: 11, *t*: 011, *s*: 1011, *n*: 1001, *p*: 10101

Cvičenie 12.15. Skonštruujte binárny strom s prefixovými kódmi reprezentujúcimi tieto kódové schémy:

(a) *a*: 11, *e*: 0, *t*: 101, *s*: 100
(b) *a*: 1, *e*: 01, *t*: 001, *s*: 0001, *n*: 00001
(c) *a*: 1010, *e*: 0, *t*: 11, *s*: 1011, *n*: 1001, *p*: 10001

Cvičenie 12.16. Skonštruujte Huffmanove kódovanie pre nasledujúce symboly s frekvenciami: a: 0.2, b: 0.1, c: 0.15, d: 0.25, e: 0.3

Cvičenie 12.17. Reprezentujte nasledujúce výrazy ako binárne stromy

(d) $(r \otimes s) \oplus ((x \oplus y) \otimes z)$

(e) $r \otimes (s \oplus ((x \oplus y) \otimes z))$

(f) $(((r \otimes s) \oplus x) \oplus y) \otimes z$

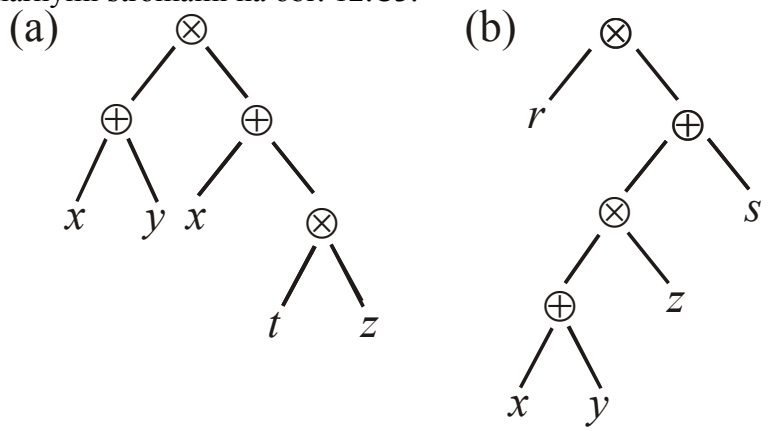
Cvičenie 12.18. Koľko rozdielnych možných interpretácií má každý z nasledujúcich výrazov, keď predpokladáme asociatívnosť operácie $\otimes$ a keď ju nepredpokladáme?

(g) $x \otimes y \otimes z$

(h) $t \oplus x \otimes y \otimes z$

(i) $t \otimes x \oplus y \otimes z$

Cvičenie 12.19. Zostrojte infixovú, prefixovú a postfixovú formu výrazov reprezentovaných nasledujúcimi binárnymi stromami na obr. 12.C3.



Obrázok 12.C3. Zostrojte infixovú, prefixovú a postfixovú formu stromov

Cvičenie 12.20. Zostrojte strom riešení pre hru odoberania zápaliek, kedy máte na začiatku hry 5 zápaliek, každý hráč môže odobrať alebo jednu, alebo 2 zápalky, a kto odoberie poslednú zápalku, tak prehral. Vrcholy z jednotlivých vrstiev stromu ohodnoťte pomocou minimax princípu.