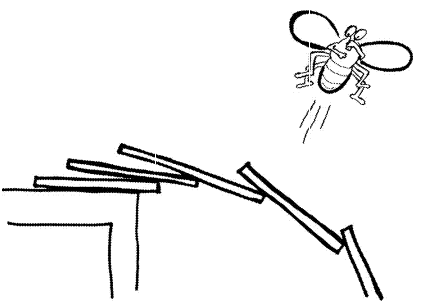
INFORMATIKA

Využití aplikace MS Excel pro výpočet těžiště soustavy kvádrů

ŠTĚPÁN HUBÁLOVSKÝ

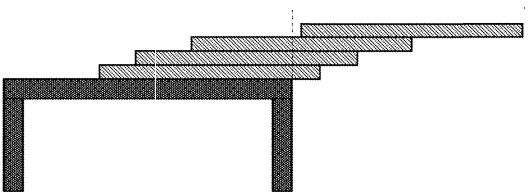
Pedagogická fakulta UHK, Hradec Králové



Článek se zabývá využitím počítače v modelování fyzikální úlohy z mechaniky tuhého tělesa. Řešení úlohy podané v tomto článku ukazuje jednu z možností, jak lze navzájem propojit výuku předmětů fyzika, matematika a informatika na střední škole.

Zadání úlohy

Lze postavit na stůl šikmou věž z identických homogenních kvádrů tak, aby vrchní kvádr spočíval celý mimo desku stolu? Pokud ano, určete, kolik kvádrů celkem potřebujeme. Situace je vyobrazena na obrázku 1.

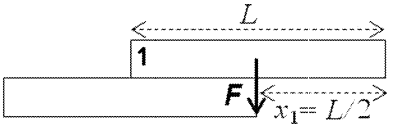


Obr. 1

Fyzikální pohled na řešení této úlohy lze najít např. v [1], matematický pohled je v článku [2] v MFI a v tomto článku kompletujeme řešení ještě přidáním modelování této úlohy za pomocí počítače.

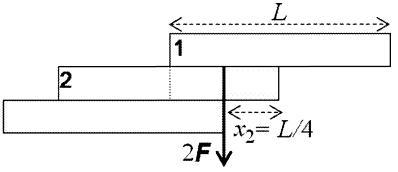
Fyzikální analýza problému

Vzhledem k tomu, že kvádry jsou homogenní, má každý z nich těžiště ve svém středu. Kvádr **1** (obr. 2) můžeme tedy maximálně vysunout tak, že jeho těžiště leží přesně nad hranou kvádru pod ním, tedy o $x_1 = \dfrac{L}{2}$ přes spodní kvádr.



Obr. 2

Obdobnou úvahu provedeme pro soustavu kvádrů **1** a **2**. Situace je zřejmá z obrázku 3. Těžiště dvojice kvádrů **1** a **2** leží nad hranou třetího kvádru.



Obr. 3

Využitím momentové věty (viz např. [3])

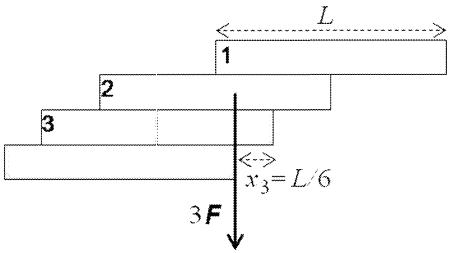
$$F\left(\frac{L}{2} - x_2\right) = Fx_2 \tag{1}$$

dostaneme řešení $x_2 = \dfrac{L}{4}$. Těžiště soustavy kvádrů **1** a **2** se nachází v 1/4 od pravé hrany kvádru **2**. Znamená to, že dvojici kvádrů **1** a **2** lze po kvádru **3** vysunout o další 1/4 délky L.

Dále postupujeme obdobně – je třeba zjistit, o kolik lze vysunout soustavu kvádrů **1**, **2** a **3** vzhledem ke kvádru **4**. Určíme opět polohu jejich společného těžiště a využijeme momentovou větu:

$$F\left(\frac{L}{2}-x_3\right)=F_{12}x_3=2Fx_3, \tag{2}$$

dostáváme polohu těžiště $x_3=\dfrac{L}{6}$ od pravé hrany kvádru **3**. Všechny tři kvádry dohromady lze tedy vysunout o další $1/6$ délky.



Obr. 4

Obdobnou úvahou zjistíme, že soustavu kvádrů **1**, **2**, **3** a **4** lze vysunout o další $1/8L$. Celkový přesah pravého okraje kvádru **1** přes okraj stolu pro soustavu 4 kvádrů je tedy roven částečnému součtu řady:

$$S=\left(\frac{1}{2}+\frac{1}{4}+\frac{1}{6}+\frac{1}{8}\right)L=\frac{25}{24}L.$$

Znamená to, že již při čtyřech kvádrech lze docílit stavu, kdy horní kvádr celý leží mimo hranu stolu.

S vysunováním lze pokračovat i dále, úlohu můžeme zobecnit i na větší počet kvádrů. Momentová věta pro výpočet těžiště soustavy K kvádrů má tvar:

$$F\left(\frac{L}{2}-x_K\right)=(K-1)Fx_K, \tag{3}$$

odkud $x_K=\dfrac{L}{2K}$.

Matematická analýza problému

Na základě fyzikální analýzy problému je zřejmé, že úlohu lze transformovat na úlohu matematickou, v níž hledáme N-tý částečný součet řady

$\frac{1}{2}+\frac{1}{4}+\frac{1}{6}+\ldots$, který je větší než 1, přičemž N je nejmenší možná hodnota:

$$s_N = \sum_{k=1}^{N} \frac{1}{2k} = \frac{1}{2} \sum_{k=1}^{N} \frac{1}{k} \geq 1. \tag{4}$$

Vzhledem k tomu, že harmonická řada $\sum_{k=1}^{\infty} \frac{1}{k}$ je divergentní, je divergentní i řada $\frac{1}{2} \sum_{k=1}^{\infty} \frac{1}{k}$. Úlohu lze proto formulovat tak, že budeme hledat nejmenší přirozené N takové, aby N-tý částečný této řady byl větší, než předem stanovené kladné reálné číslo S_{MAX}:

$$s_N = \frac{1}{2} \sum_{k=1}^{N} \frac{1}{k} \geq S_{MAX}. \tag{5}$$

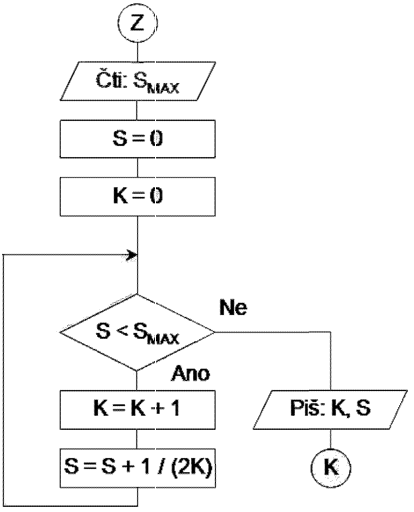
S ohledem na naši úlohu má N význam počtu použitých kvádrů a S_{MAX} vyjadřuje horizontální přesah pravého okraje horního kvádru vzhledem k desce stolu.

Poznámka 1. Libovolného přesahu lze ovšem dosáhnout jen teoreticky, prakticky u hmotných kvádrů narazíme na několik okolností, které způsobí, že by náš reálný model fungoval jen přibližně a pro nevelký počet kvádrů.

Algoritmizace úlohy

Úlohu lze numericky řešit za pomocí jednoduchého algoritmu, který pro zadanou hodnotu S_{MAX} najde nejmenší přirozené číslo N takové, aby byla splněna podmínka (5). Algoritmus výpočtu je znázorněn na vývojovém diagramu (obr. 5).

Výpočet se zde realizuje pomocí cyklu while, přičemž v K-tém kroku cyklu ($K = 1, 2, 3, \ldots$) je vypočtena hodnota K-tého členu řady, a tato hodnota se akumuluje do proměnné S. Příkaz *while* na začátku cyklu hlídá podmínku (5) a po jejím splnění se vypíše výsledek: výsledné K (počet použitých členů řady) a S (nejmenší částečný součet, který je větší než S_{MAX} = požadovaný horizontální přesah pravé hrany horního kvádru od hrany stolu).



Obr. 5

Programové zpracování úlohy

Algoritmus řešení úlohy můžeme pro počítač zpracovat v libovolném programovacím jazyce, zvolme Pascal:

```
program Teziste;
uses Crt;
var  Smax,  S:  Real;
     K:  Integer;
begin
  ClrScr;
  Write('Zadej presah horniho kvadru: ');
  ReadLn(Smax);
  S:= 0;
  K:= 0;
  while   S < Smax   do
  begin
     K := K + 1;
     S := S + 1/(2*K)
```

```
  end;
  WriteLn(K:10,S:10:3);
  ReadLn
end.
```

Zpracování úlohy v aplikaci MS Excel

Úlohu můžeme na počítači zpracovat i pomocí aplikace MS Excel. Ukážeme dvě varianty zpracování.

První a jednodušší varianta využívá metodu výpočtu pomocí vzorců zapsaných do buněk Excelu. V buňkách sloupce A jsou pod sebou pořadová čísla kvádrů K. V buňkách sloupce B jsou vzorce, které pro danou buňku vypočítají odpovídající hodnotu členu řady. V buňkách sloupce C jsou vzorce počítající částečný součet $s_N = s(N)$ prvních N členů řady dle (5). Součástí vzorců je i podmínka z (5), která ukončí výpočet částečných součtů v případě, že tento součet překročí hodnotu S_{MAX} zadanou uživatelem do buňky C1. Do buňky C4 je tedy vloženo KDYŽ(C4<$C1; C3+B4). Vzorce stačí napsat do jednoho řádku a tažením za úchyt je zkopírovat do libovolného počtu dalších řádků v daném sloupci s využitím relativní adresace buněk ve vzorcích. Na obrázku 6 je kopie listu Excelu pro hodnotu $S_{MAX} = 1{,}5$.

Poznámka 2. Harmonická řada diverguje $k + \infty$ „velmi pomalu“, např. pro přesah $S_{MAX} = 1$ stačí 4 členy (4 kvádry), pro přesah 1,5 (viz obr. 6) je třeba použít již 11 kvádrů, pro přesah 2 by to bylo již přes 30 kvádrů a pro přesah např. 5 by bylo zapotřebí více než 12 360 kvádrů. Viz předchozí poznámku 1.

Ve druhé variantě je tu ukázáno zpracování této úlohy využitím programovacího jazyka Visual Basic for Application (*VBA*), který je součástí aplikace *MS Excel*. Tento implementovaný programovací jazyk se využívá především ke zjednodušení rutinních prací a k „automatickému“ zpracovávání dat. Přestože *VBA* patří mezi moderní objektově orientované a událostmi řízené programovací jazyky a můžeme s jeho pomocí vytvářet v Excelu i vlastní formuláře, lze ho použít také ve strukturovaném přístupu k programování a zpracovávat v něm jednoduché algoritmy, mezi něž patří i naše úloha. Níže je kopie zdrojového kódu *VBA*.

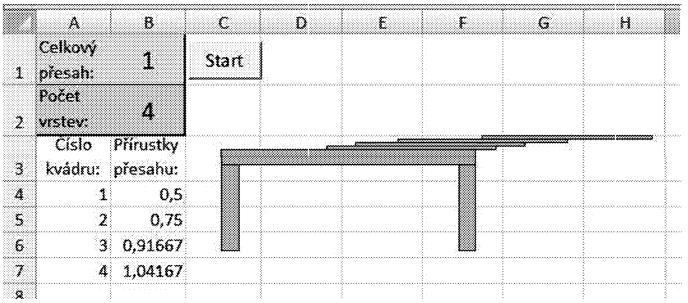
	A	B	C
1		S =	1,50
2	K	1 / (2*K)	s(N)
3			0,00
4	1	0,50000000	0,50000000
5	2	0,25000000	0,75000000
6	3	0,16666667	0,91666667
7	4	0,12500000	1,04166667
8	5	0,10000000	1,14166667
9	6	0,08333333	1,22500000
10	7	0,07142857	1,29642857
11	8	0,06250000	1,35892857
12	9	0,05555556	1,41448413
13	10	0,05000000	1,46448413
14	11	0,04545455	1,50993867
15	12	0,04166667	NEPRAVDA
16	13	0,03846154	NEPRAVDA
17	14	0,03571429	NEPRAVDA

Obr. 6

```
Sub Těžiště_soustavy_kvádrů()
  Call Vymazat_list 'Grafický podprogram
  Smax = InputBox("Zadej přesah horního kvádru")
  S=0
  K=0
  Call Kresli_Kvádr(K, S, Smax) 'Grafický podprogram
  Do While   S < Smax
    K = K + 1
    S = S + 1  / (2 * K)
    Cells(K  +  3,  1)  =  K
    Cells(K  +  3,  2)  =  S
    Call Kresli_Kvádr(K, S, Smax)  'Grafický   podprogram
  Loop
    Call Kresli_Stůl(K, S, Smax)  'Grafický podprogram
    Cells(1, 2) = Smax
    Cells(2, 2) = K
End Sub
```

Pro zadání vstupní hodnoty S_{MAX} používáme metodu *InputBox*. Hodnoty proměnné K (udávající počet kvádrů) program vypisuje do buněk ve

sloupci A příkazem `Cells(K + 3, 1) = K`, částečné součty řady S, udávající velikost celkového přesahu kvádrů, program vypisuje do sloupce B příkazem `Cells(K + 3, 2) = S`. Program je rozšířen i o grafické znázornění „poskládání“ kvádrů na sobě, každý obdélník kreslený na pracovní plochu listu zastupuje jeden kvádr. Na obrázku 7 je řešení pro $N = 1$. V tomto článku však není dost prostoru, kde bychom popsali syntaxi grafiky *VBA* pro Excel, a proto v hlavním programu jsou uvedeny pouze odkazy na podprogramy, které grafické znázornění provedou (*Call Vymazat_list*, *Call Kresli_Hranol(K, S, N)*, *Call Kresli_Stůl(K, S, N)*). Další informace o *VBA* lze získat například v [5].



Obr. 7

Závěr

V článku jsme ukázali jednu z možností, jak vzájemně spojit výuku fyziky, matematiky, algoritmizace a programování. Uvedenou fyzikální úlohu lze také demonstrovat jednoduchým, efektním a ke zdárnému výsledku vedoucím experimentem (např. s kartami), což může ještě více přispět k motivaci studentů.

Literatura

[1] *Fenclová, J.*: Didaktické myšlení jednání učitele fyziky. SPN, Praha 1984.

[2] *Calda, E.*: Harmonická řada a nakloněná věž, která nespadne. MFI roč. 12 (2002/03), č. 3.

[3] *Svoboda, E.*: Přehled středoškolské fyziky. Prometheus, Praha 1992.

[4] *Odvárko, O.*: Matematika pro gymnázia – Posloupnosti a řady. Prometheus, Praha, 1995.

[5] *Walkenbach, J.*: Microsoft Excel, Programování ve VBA. Computer Press, Brno, 2004.

Nalezení všech nejkratších cest mezi dvěma vrcholy

EVA MILKOVÁ

Katedra informatiky a managementu, Univerzita Hradec Králové,
http://lide.uhk.cz/fim/ucitel/milkoev1/

Velmi častou úlohou, se kterou se v praxi setkáváme, je zjištění, jak se lze nejkratším možným způsobem dostat z jednoho místa na jiné. A v některých případech potřebujeme určit nejen jedno, ale všechna nejkratší spojení mezi zadanými místy.

V článku „Prohledávání grafů do šířky: hledání kružnic s danými vlastnostmi“ (viz [1]) jsme se zabývali prohledáváním souvislého neorientovaného grafu do šířky a

- definovali pojem **strom prohledávání do šířky**:

Definice: Nechť $G = (V, E)$ je souvislý graf, T kostra získaná algoritmem prohledávání grafu G do šířky se začátkem prohledávání ve vrcholu v. Kořenový strom (T, v) s kořenem ve vrcholu v nazýváme *strom prohledávání do šířky* příslušný prohledávání grafu G do šířky s počátkem prohledávání ve vrcholu v.

- formulovali **vlastnost stromu prohledávání do šířky**:

Věta (vlastnost stromu prohledáváni do šířky): Nechť $G = (V, E)$ je souvislý graf, T kostra získaná algoritmem prohledávání vrcholů grafu G do šířky se začátkem prohledávání ve vrcholu v a (T, v) příslušný strom prohledávání do šířky. Pak pro koncové vrcholy libovolné hrany grafu G, která neleží v (T, v), platí, že tyto vrcholy leží ve stejné vrstvě nebo v sousedních vrstvách stromu (T, v).

- a s touto vlastností související tvrzení, zde připomeňme pouze tvrzení vztahující se k **nalezení nejkratší cesty mezi dvěma vrcholy**:

Tvrzení 1: Nechť G je souvislý graf a (T, v) jeho kořenový strom prohledávání do šířky. Pak délka nejkratší cesty z vrcholu v do vrcholu y v grafu G

je rovna $h(y)$, kde $h(y)$ značí číslo vrstvy stromu (T, v), ve které leží vrchol y. (Pozn.: Nejkratší cestou z vrcholu v do vrcholu y rozumíme tu cestu mezi těmito vrcholy, která má minimální počet hran.)

V tomto článku ukážeme, jak můžeme postupovat, chceme-li najít všechny nejkratší cesty mezi dvěma vrcholy v zadaném grafu. Nejprve se zabýváme hledáním cest s minimálním počtem hran (tj. nejkratších cest v hranově neohodnoceném grafu), a pak získané poznatky aplikujeme na hledání nejkratších cest mezi dvěma vrcholy v grafu hranově ohodnoceném. V obou případech uvažujeme souvislý neorientovaný graf.

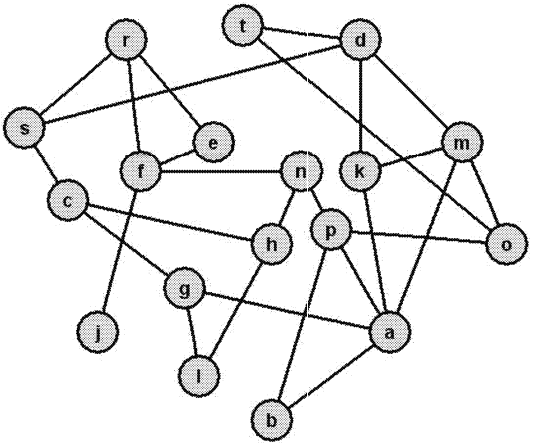
Hledání cest s minimálním počtem hran mezi vrcholy x a y v neorientovaném grafu

Předpokládejme, že graf G je hranově neohodnocený. Algoritmus, kterým najdeme v grafu G všechny cesty z vrcholu x do vrcholu y s minimálním počtem hran, stručně řečeno všechny nejkratší cesty z vrcholu x do vrcholu y, rozdělíme do dvou základních fází.

1) Pomocí algoritmu prohledávání grafu G do šířky s počátkem ve vrcholu x zaznamenáme u každého vrcholu v číslo $h(v)$, tj. číslo vrstvy stromu (T, x), ve které vrchol v leží. (Pozn.: Algoritmus můžeme ukončit ve chvíli, kdy určíme číslo $h(y)$, délku nejkratší cesty z x do y, a u zbylých neprohledaných vrcholů v položíme $h(v) = h(y)$.)
2) Pomocí modifikace algoritmu prohledávání grafu G do šířky (viz dále) vytvoříme strom všech nejkratší cest mezi vrcholy x a y (nazvěme jej *$x - y$ strom nejkratších cest*).

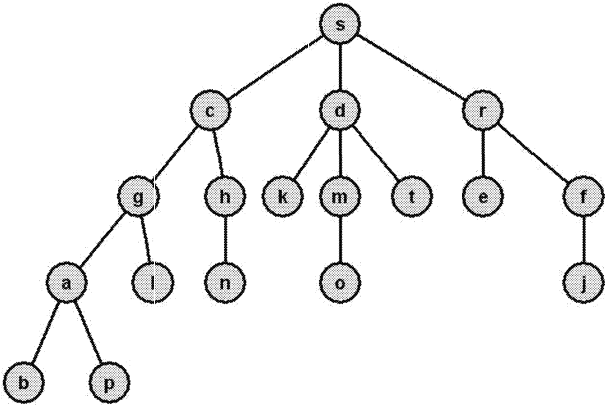
Definice: Nechť $G = (V, E)$ je souvislý graf a (T, x) jeho kořenový strom prohledávání do šířky. *$x - y$ stromem nejkratších cest* nazveme kořenový strom $T_{x,y}$ s kořenem ve vrcholu $y[1]$ a listy $x[1], \ldots, x[k]$, kde k je počet nejkratších cest z vrcholu x do vrcholu y existujících v grafu G, přičemž každá cesta ve stromu $T_{x,y}$ vedoucí z listu $x[i]$, $i = 1, \ldots, k$, ke kořeni $y[1]$ reprezentuje právě jednu nejkratší cestu z vrcholu x do vrcholu y v grafu G.

Pro lepší názornost, dříve než přistoupíme k nástinu algoritmu, pomocí něhož $x-y$ strom nejkratších cest vytvoříme, ilustrujme celý postup získání $s - p$ stromu nejkratších cest mezi vrcholy s a p v následujícím grafu G. (obr. 1)

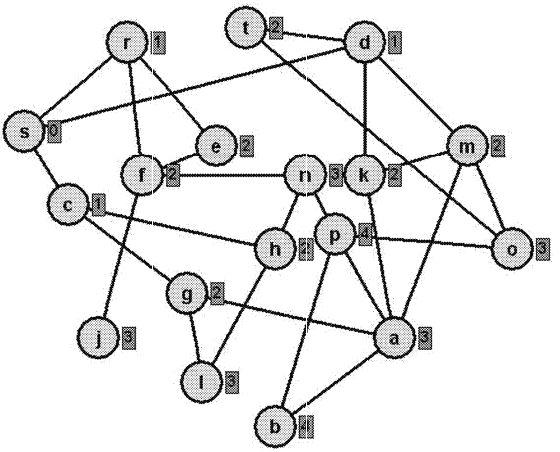


Obr. 1 Graf G

Provedením algoritmu prohledávání grafu G do šířky s počátkem ve vrcholu s získáme strom prohledávání do šířky (obr. 2), čímž získáme pro každý vrchol v číslo $h(v)$ – viz obr. 3.

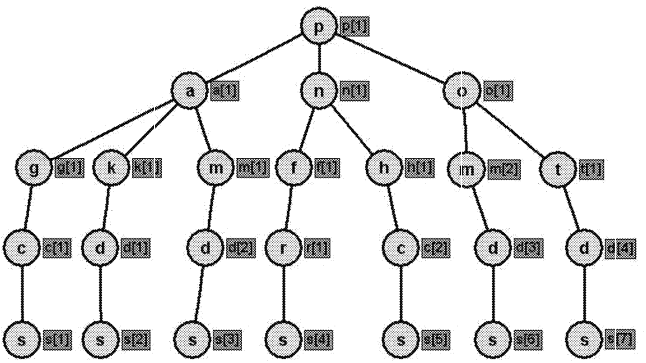


Obr. 2 Strom prohledávání do šířky (T, s).



Obr. 3 Graf G doplněný u každého vrcholu v o číslo $h(v)$.

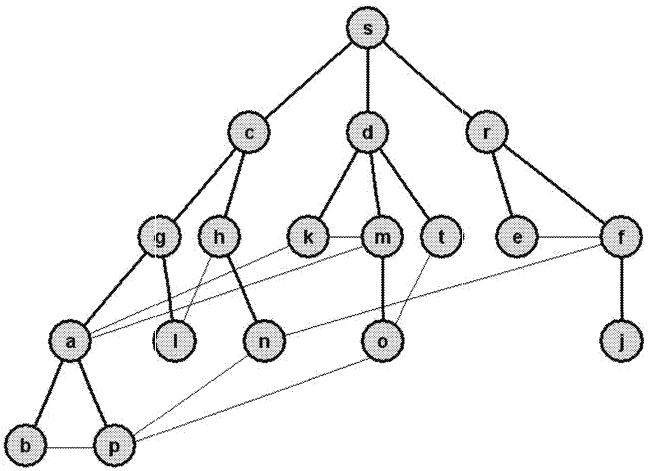
Výsledný $s-p$ strom nejkratších cest je zobrazen na obr. 4, přičemž jeho množinu vrcholů tvoří vrcholy $a[1]$, $c[1]$, $c[2]$, $d[1]$, $d[2]$, $d[3]$, $d[4]$, $f[1]$, $g[1]$, $h[1]$, $k[1]$, $m[1]$, $m[2]$, $n[1]$, $o[1]$, $p[1]$, $r[1]$, $s[1]$, $s[2]$, $s[3]$, $s[4]$, $s[5]$, $s[6]$, $s[7]$, $t[1]$. Pro větší názornost jsou v jednotlivých vrcholech uvedena jména příslušných reprezentovaných vrcholů grafu G.



Obr. 4 $s-p$ strom nejkratších cest

Z obrázku $s-p$ stromu nejkratších cest je všech 7 nejkratších cest mezi vrcholy s a p zcela zřejmých.

Pro kontrolu správnosti nalezených cest uveďme i obrázek stromu (T, s) doplněného o hrany grafu G, které ve stromu (T, s) neleží (viz obr. 5). Protože se jedná o malý graf, lze z něj poměrně snadno vyčíst všechny nejkratší cesty z vrcholu s do vrcholu p existující v grafu G, a vidíme, že tyto cesty se shodují s cestami zobrazenými v $s-p$ stromu nejkratších cest na obr. 4.



Obr. 5 Strom prohledávání do šířky (T, s) doplněný o zbylé hrany grafu G

Konstrukce $x - y$ stromu nejkratších cest

Pomocí algoritmu prohledávání grafu G do šířky s počátkem ve vrcholu x určíme u každého vrcholu v číslo $h(v)$. Nechť $q = h(y)$ značí délku nejkratší cesty z x do y.

- Kořen stromu $T_{x,y}$ tvoří vrchol $y[1]$ reprezentující vrchol y grafu G.
- První vrstvu stromu $T_{x,y}$, tj. následníky kořene $y[1]$, tvoří vrcholy $v[i]$ reprezentující sousední vrcholy v vrcholu y v grafu G, jejichž číslo $h(v)$ je rovno $h(y) - 1$, přičemž index i u vrcholu v označuje i-tý výskyt vrcholu v ve stromu $T_{x,y}$.
- Druhou vrstvu stromu $T_{x,y}$ tvoří vrcholy $w[i]$ reprezentující, postupně pro každý vrchol $v[j]$ z předchozí vrstvy, sousední vrcholy w vrcholu v v grafu G, jejichž číslo $h(w)$ je rovno $h(v) - 1$ (tj. $h(w) = h(y) - 2$), přičemž index i u vrcholu w označuje i-tý výskyt vrcholu w ve stromu $T_{x,y}$.
- Analogicky jsou tvořeny další vrstvy stromu $T_{x,y}$, třetí až q-tá vrstva.

Poznámka: Je zřejmé, že vrcholy ležící v první vrstvě stromu $T_{x,y}$ mají vždy pouze index 1.

Důkaz správnosti konstrukce $x-y$ stromu nejkratších cest

Z vlastnosti stromu (T,x) víme, že pro každé dva sousední vrcholy v, w grafu G platí právě jedna ze tří možností: $h(v)=h(w)-1$, $h(v)=h(w)$, $h(v)=h(w)+1$.

Z konstrukce $T_{x,y}$ vidíme, že pro každý vrchol v grafu G uvažujeme pro zařazení do stromu $T_{x,y}$ pouze ty sousední vrcholy vrcholu v, které leží ve vrstvě $h(v)-1$, tudíž je zřejmé, že poslední q-tou vrstvu tvoří vrcholy $x[1]$, ... , $x[k]$ reprezentující vrchol x, protože $h(x)=0$ (pozn.: $h(x)=h(y)-q$), přičemž k značí počet výskytů vrcholu x ve stromu $T_{x,y}$. Odsud plyne, že každá cesta z vrcholu $x[i]$, $i=1,\ldots,k$, ke kořeni $y[1]$ ve stromu $T_{x,y}$ reprezentuje cestu z vrcholu x do vrcholu y v grafu G délky q.

Zbývá dokázat, že každá cesta délky q z x do y (tj. každá nejkratší cesta z x do y) vyskytující se v grafu G je reprezentována ve stromu $T_{x,y}$. Nechť $P_q=(x=v_0,v_1,\ldots,v_{q-1},v_q=y)$ je cesta délky q v grafu G. Protože cesta P_q je cestou nejkratší délky, musí pro každý její vrchol v_t platit, že délka nejkratší cesty z x do vrcholu v_t je rovna t, tj. $h(h_t)=t$, $t=1,\ldots,q$. Pro každou z q dvojic sousedních vrcholů v_{t-1}, v_t cesty P_q tudíž platí $h(v_{t-1})=h(v_t)-1$, což znamená, že cesta P_q je reprezentována některou z cest konstruovaných ve stromu $T_{x,y}$.

K sestavení $x-y$ stromu nejkratších cest můžeme využít jednoduchou modifikaci algoritmu prohledávání grafu do šířky, která spočívá v tom, že do fronty nejprve uložíme vrchol y a zaznamenáme u něj index 1 a číslo $h(y)$, tj. $y[1](h(y))$. Dále pracujeme s frontou standardně, tj. pro vrchol $v[i]$ ležící na začátku fronty ukládáme do fronty postupně sousední (v zadaném grafu) vrcholy w vrcholu v, jejichž číslo $h(w)=h(v)-1$, a u každého vrcholu w zaznamenáme jednak index odpovídající pořadí jeho výskytu ve frontě, a jednak číslo $h(w)$. Pokud žádné sousední vrcholy vrcholu v neexistují, vrchol $v[i]$ z fronty odebereme.

Vrcholy (s indexem) postupně ukládané do fronty ukládáme současně do stromu $T_{x,y}$ spolu s informací o jejich předchůdci, přičemž v zájmu formální jednoduchosti položíme $y[1](y[1])$. Ze získaných informací pak lehce určíme všechny nejkratší cesty mezi vrcholy x a y tak, že postupujeme ve vytvořeném stromu $T_{x,y}$ od jednotlivých listů (vrcholů $x[1],\ldots,x[k]$) ke kořeni $y[1]$. U výsledných cest pochopitelně zapisujeme vrcholy již bez indexů.

Ilustrujme nastíněný algoritmus pro hledání všech nejkratších cest existujících v grafu G z obr. 1. Předpokládejme, že jsme již pomocí prohledávání do šířky určili u každého vrcholu v jeho číslo $h(v)$ ve stromu (T, s) – viz obr. 3.

Do fronty zařazujeme postupně vrcholy v pořadí:

$p[1](4)$, $a[1](3)$, $n[1](3)$, $o[1](3)$, $g[1](2)$, $k[1](2)$, $m[1](2)$, $f[1](2)$, $h[1](2)$, $m[2](2)$, $t[1](2)$, $c[1](1)$, $d[1](1)$, $d[2](1)$, $r[1](1)$, $c[2](1)$, $d[3](1)$, $d[4](1)$, $s[1](0)$, $s[2](0)$, $s[3](0)$, $s[4](0)$, $s[5](0)$, $s[6](0)$, $s[7](0)$

A současně vytváříme $s-p$ strom nejkratších cest:

$p[1](p[1])$, $a[1](p[1])$, $n[1](p[1])$, $o[1](p[1])$, $g[1](a[1])$, $k[1](a[1])$, $m[1](a[1])$, $f[1](n[1])$, $h[1](n[1])$, $m[2](o[1])$, $t[1](o[1])$, $c[1](g[1])$, $d[1](k[1])$, $d[2](m[1])$, $r[1](f[1])$, $c[2](h[1])$, $d[3](m[2])$, $d[4](t[1])$, $s[1](c[1])$, $s[2](d[1])$, $s[3](d[2])$, $s[4](r[1])$, $s[5](c[2])$, $s[6](d[3])$, $s[7](d[4])$

Výsledné cesty:

(s, d, t, o, p); (s, d, m, o, p); (s, c, h, n, p); (s, r, f, n, p); (s, d, m, a, p); (s, d, k, a, p); (s, c, g, a, p)

Hledání nejkratších cest mezi vrcholy x a y v hranově ohodnoceném neorientovaném grafu

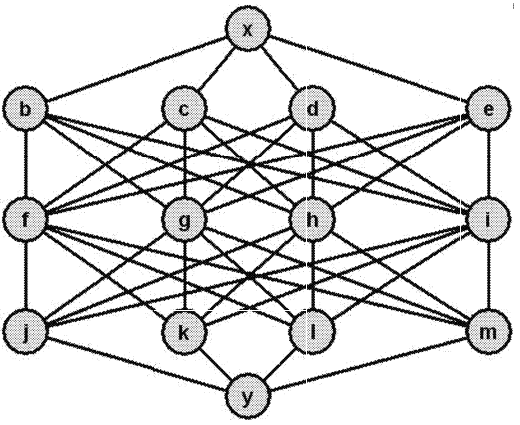
Při hledání všech nejkratších cest z vrcholu x do vrcholu y v hranově ohodnoceném neorientovaném grafu G postupujeme obdobně, jak bylo popsáno výše. Nejprve pro každý vrchol v určíme pomocí vhodného algoritmu (např. Dijkstrova algoritmu v případě nezáporně hranově ohodnoceného grafu – viz [2]) délku d_v nejkratší cesty z vrcholu x do vrcholu v v grafu G, a pak konstruujeme $x-y$ strom nejkratších cest $T^*_{x,y}$ pro hranově ohodnocený graf. Konstrukce stromu $T^*_{x,y}$ pro hranově ohodnocený graf probíhá analogicky jako konstrukce stromu $T_{x,y}$ pro hranově neohodnocený graf, rozdíl je pouze v tom, že místo s čísly $h(v)$ pracujeme s čísly d_v a pro zařazení vrcholu do stromu $T_{x,y}$ uvažujeme ty sousední vrcholy w vrcholu v v grafu G, pro které platí $d_w = d_v - e\{w, v\}$, kde $e\{w, v\}$ je ohodnocení hrany $\{w, v\}$.

Poznámka: Uvědomme si, že délkou cesty P v hranově ohodnoceném grafu máme na mysli součet ohodnocení hran cesty P nikoli počet hran cesty P. Proto v případě stromu $T^*_{x,y}$ nemusí jeho listy $x[1], \ldots, x[k]$, kde k je počet nejkratších cest z vrcholu x do vrcholu y existujících v hranově ohodnoceném grafu G, ležet v téže vrstvě.

Závěr

Uvedené algoritmy lze aplikovat analogicky na orientované grafy. V implementacích algoritmů je však zapotřebí zohlednit orientaci probíraných hran. Z důvodu korektnosti je též vhodné na tomto místě uvést, že vlastnost stromu prohledávání do šířky připomenutá na začátku tohoto článku neplatí pro orientované grafy. Avšak tvrzení uvedené pod ní, které je pro výše uvedenou konstrukci stromu $T_{x,y}$ podstatné, platí.

Časová složitost první fáze uvedených algoritmů je polynomiální (v případě prohledávání do šířky a v případě hledání nejkratších cest v acyklických grafech je $O(n+m)$, v případě Dijkstrova algoritmu $O(n^2)$), avšak konstrukce stromů $T_{x,y}$ a $T^*_{x,y}$ závisí na počtu existujících nejkratších cest v daném grafu a jejich počet může s rostoucím počtem vrcholů a hran grafu růst exponenciálně (stačí si představit grafy typu grafu na obr. 6 se zvyšováním počtu „částí“, kde vrcholy jedné jsou propojeny se všemi vrcholy druhé, a zvyšování počtu vrcholů v těchto „částech“).



Obr. 6 Graf obsahující 64 různých nejkratších cest délky 4 z vrcholu x do vrcholu y

Literatura

[1] *Milková, E.*: Prohledávání grafů do šířky: hledání kružnic s danými vlastnostmi. MFI 19 (2009/10), č. 7, s. 434 – 441.

[2] *Milková, E.*: Modifikace grafových algoritmů (od Jarníka k Dijkstrovi). MFI 11 (2001–02), č. 3, s. 178 – 182.