Vlastnické vztahy (Úlohy z MO kategorie P, 47. část)

PAVEL TÖPFER

Matematicko-fyzikální fakulta UK, Praha

V dnešním pokračování se seznámíme s jednou soutěžní úlohou z domácího kola 44. ročníku Matematické olympiády kategorie P (programování). Tento ročník olympiády probíhal ve školním roce 1994/95, tedy před téměř třiceti lety. Zadání úlohy je poměrně krátké a uvedeme ho sice v původní podobě, ale s drobnými formulačními úpravami, které přispějí k vyjasnění a upřesnění řešené úlohy.

* * * * * * * * * * * *

Občas se stává, že některé společnosti jsou *částečnými vlastníky* jiných společností, neboť získaly část jejich akcií. Jedna společnost může vlastnit například 12 % akcií jiné společnosti. Řekneme, že společnost A *kontroluje* společnost B, pokud

- buď společnost A vlastní více než 50 % akcií společnosti B,
- nebo společnost A vlastní p procent společnosti B a zároveň kontroluje k ($k \geq 1$) společností C_1, …, C_k takových, že C_i vlastní x_i procent společnosti B (pro všechna i, $1 \leq i \leq k$) a přitom platí $p + x_1 + \ldots + x_k > 50$.

Je zadán celkový počet všech společností n, jednotlivé společnosti si označíme čísly do 1 do n. Dále je dán seznam trojic celých čísel (i, j, p), které znamenají, že společnost i vlastní p % společnosti j. Platí $1 \leq i \leq n$, $1 \leq j \leq n$, $1 \leq p \leq 100$.

Napište program, který v co nejkratším čase určí všechny dvojice (i, j) takové, že společnost i kontroluje společnost j. Výsledek ve formě seznamu dvojic (i, j) uspořádejte podle rostoucí hodnoty i.

* * * * * * * * * * * *

Vlastnickou strukturu společností si můžeme znázornit pomocí orientovaného grafu s ohodnocenými hranami. Vrcholy tohoto grafu budou představovat společnosti, hrany grafu reprezentují vlastnické vztahy. Orientovaná hrana vedoucí v grafu z vrcholu i do vrcholu j s ohodnocením p znamená, že společnost i vlastní p procent společnosti j. Seznam trojic čísel zadaný na vstupu programu je tedy přesně výčtem hran uvažovaného grafu.

Popsaný graf můžeme reprezentovat v našem programu některým ze standardních způsobů uložení grafu – například pomocí matice ohodnocení hran nebo pomocí seznamů následníků jednotlivých vrcholů. Pokud zvolíme matici ohodnocení hran, pak budeme pracovat se čtvercovou maticí m, v níž hodnota $m[i, j] = p$ představuje orientovanou hranu z vrcholu i do vrcholu j s ohodnocením p. Znamená tedy, že společnost i vlastní p procent společnosti j.

Dříve, než přistoupíme k vlastnímu řešení úlohy, uvědomíme si několik důležitých skutečností:

1. Nejjednodušším způsobem, jak může společnost A kontrolovat společnost B, je přímo vlastnit více než 50 % akcií společnosti B. Tomu odpovídá hrana v grafu s ohodnocením větším než 50.

2. Pokud cizí společnosti vlastní dohromady nejvýše 50 % akcií společnosti B, pak zcela jistě nikdo nemůže kontrolovat společnost B. Tato situace odpovídá tomu, že součet ohodnocení všech hran vedoucích do vrcholu B je roven nejvýše 50.

3. Společnost B nemusí nikdo kontrolovat, ani když jiné společnosti vlastní třeba i všechny její akcie – to nastane, pokud například společnosti C_1, C_2 vlastní každá 50 % akcií společnosti B a přitom akcie společností C_1, C_2 nikdo jiný nevlastní.

4. Ke kontrolování nějaké společnosti je zapotřebí ovládat (přímo nebo nepřímo) více než polovinu jejích akcií a to může vždy nejvýše jedna společnost.

5. V orientovaném grafu znázorňujícím vlastnické vztahy společností mohou být obsaženy cykly – například společnost A může být částečným vlastníkem společnosti B, společnost B částečným vlastníkem společnosti C a společnost C částečným vlastníkem společnosti A.

Pojem „kontrolovat společnost“ je definován rekurzivně, takže se přirozeně nabízí využít rekurzi i při návrhu řešení. První varianta řešení bude proto velmi jednoduchá a názorná.

Sestrojíme rekurzivní funkci *kontroluje*(i, j), která bude přesně podle definice určovat, zda společnost i kontroluje společnost j. Nejprve snadno ověříme platnost podmínek podle bodu 1 a 2 z předchozího odstavce. Kontrola první podmínky je přímočará: jednoduše otestujeme, zda platí $m[i, j] > 50$. Jestliže ano, funkce rovnou vrátí hodnotu `True`. Také otestování druhé podmínky je jednoduché: pokud součet ohodnocení všech hran, které vstupují do vrcholu j, nebude větší než 50, pak je zjevně správnou odpovědí `False`. Jestliže naopak součet ohodnocení všech hran vstupujících do vrcholu j převyšuje hodnotu 50, pak musíme ověřit, zda je mezi nimi taková skupina hran se součtem ohodnocení větším než 50, které vycházejí z vrcholů kontrolovaných vrcholem i (včetně případné hrany z vrcholu i samotného). To zjistíme opakovaným rekurzivním voláním *kontroluje*(i, k) pro všechny vrcholy k, z nichž vede hrana do j.

Rekurzivní funkce na určení, zda společnost i kontroluje společnost j, může vypadat třeba takto:

```
def kontroluje(i, j): # ověření, zda "i" kontroluje "j"
    if m[i][j] > 50:
        return True   # přímé vlastnictví (hrana > 50)
    suma = 0
    for k in range(1, n+1):
        suma += m[k][j]
    if suma <= 50:
        return False  # součet všech vstupních hran <= 50
    suma = 0
    for k in range(1, n+1):
        if k == i or (m[k][j] > 0 and kontroluje(i, k)):
            suma += m[k][j]
    if suma > 50:
        return True
    return False
```

Právě popsanou funkci *kontroluje*(i, j) zavoláme postupně pro všechny dvojice společností i, j. Dvojice s kladnou odpovědí uložíme do výsledného seznamu nebo je ve správném pořadí budeme přímo vypisovat:

```
n = int(input("Počet společností: "))
m = [[0] * (n+1) for _ in range(n+1)]
                          # matice ohodnocení hran
v = int(input("Počet vlastnických vztahů: "))
print("Vztahy: KDO vlastní KOHO a KOLIK procent")
```

```
for h in range(v):
    i, j, p = [int(c) for c in input().split()]
    m[i][j] = p

for i in range(1, n+1):
    for j in range(1, n+1):
        if kontroluje(i, j):
            print(i, j)   # výsledky: kdo koho kontroluje
```

Dvourozměrné pole m zde představuje matici ohodnocení jednotlivých hran a slouží tedy jako datová reprezentace grafu, který zobrazuje vlastnické vztahy mezi společnostmi. Nulová hodnota $m[i][j]$ znamená, že společnost i nevlastní žádný podíl ve společnosti j, neboli v grafu neexistuje orientovaná hrana z vrcholu i do vrcholu j. Všimněte si, že ačkoliv máme n společností, matice m je v programu definována o rozměrech $(n+1)\times(n+1)$. To je ale jenom technická záležitost způsobená skutečností, že všechny seznamy se v Pythonu (i v mnoha jiných programovacích jazycích) indexují od 0, zatímco společnosti jsou podle zadání úlohy očíslovány od 1 do n. Prvky pole m s indexem 0 k ničemu nevyužíváme.

Výše uvedené řešení vypadá na první pohled správně a celkem elegantně, ale má jednu dost podstatnou vadu: pro některá vstupní data nefunguje správně. Program dává správné výsledky tehdy, pokud ve vlastnických vztazích společností nejsou žádné cyklické závislosti, neboli pokud uvažovaný graf neobsahuje žádné orientované cykly. V případě existence cyklů v grafu se ovšem může stát, že se výpočet programu dostane do nekonečné smyčky rekurzivních volání. Můžete se o tom sami přesvědčit na takto jednoduchých vstupních datech: máme 4 společnosti a mezi nimi 8 vlastnických vztahů (1 2 30), (2 1 30), (1 4 30), (4 1 30), (3 2 40), (2 3 40), (3 4 40), (4 3 40). Pro výpočet hodnoty *kontroluje*$(1,2)$ potřebujeme znát hodnotu *kontroluje*$(1,3)$, k jejímu určení se ovšem zavolá *kontroluje*$(1,2)$, a tak stále dokola.

Abychom naše řešení opravili, doplníme funkci *kontroluje*(i,j) o evidenci, s jakými hodnotami parametrů (i,j) jsme ji v aktuálním rekurzivním řetězci volání již zavolali. Díky tomu ji se stejnou dvojicí hodnot nikdy nebudeme volat opakovaně, a proto nemůže nastat nekonečná rekurze. K tomu nám poslouží globální seznam z, který bude představovat obsah volacího zásobníku v průběhu rekurzivních volání funkce *kontroluje*(i,j). Na začátku funkce vložíme do zásobníku dvojici hodnot parametrů (i,j) a před ukončením funkce tuto dvojici ze zásobníku opět odstraníme. Ja-

kékoliv rekurzivní volání funkce provedeme pouze s takovými parametry, které se momentálně nenacházejí v seznamu z. Tím vyloučíme možnost nekonečné rekurze, ale zároveň se tím ani nepřipravíme o žádné řešení úlohy.

```
z = []                    # volací zásobník

def kontroluje(i, j):     # ověření, zda "i" kontroluje "j"
    if m[i][j] > 50:
        return True       # přímé vlastnictví (hrana > 50)
    suma = 0
    for k in range(1, n+1):
        suma += m[k][j]
    if suma <= 50:
        return False      # součet všech vstupních hran <= 50
    suma = 0
    z.append((i, j))
    for k in range(1, n+1):
        if k == i or (m[k][j] > 0 and
                      (i, k) not in z and kontroluje(i, k)):
            suma += m[k][j]
    z.pop()
    if suma > 50:
        return True
    return False
```

Použití volacího zásobníku z vyřešilo naše problémy s nekonečným výpočtem rekurzivních volání, takto opravený program již dává správné výsledky pro všechna vstupní data. Nevyřešilo ale neefektivitu výpočtu, pro rozsáhlá a komplikovaná vstupní data bude výpočet našeho programu velmi pomalý. Během výpočtu totiž může být funkce *kontroluje*(i, j) postupně zavolána mnohokrát se stejnými vstupními parametry i, j, takže se budou zbytečně opakovaně počítat hodnoty, které jsme již někdy dříve spočítali. Snadno se o tom můžete přesvědčit, když si na úplný začátek funkce *kontroluje*(i, j) doplníte příkaz

```
print(">>>", i, j)
```

Ten způsobí, že se při každém zavolání funkce vypíše na výstup jeden řádek s informací, s jakými hodnotami parametrů jsme funkci právě zavolali. Zkuste si sami spustit program s takto upravenou funkcí třeba se vstupními daty, která jsme před chvílí použili při řešení problému s nekonečnou rekurzí.

Ještě názorněji uvidíte rozsah problému, když použijete vstupní data s 6 společnostmi a 30 vlastnickými vztahy, kde každá společnost vlastní 20 % akcií v každé ze zbývajících pěti společností.

Zmíněnou neefektivitu odstraníme nejlépe tak, že do programu doplníme globální dvojrozměrnou tabulku t, do které si budeme ukládat všechny již spočítané výsledné návratové hodnoty funkce *kontroluje*(). Tabulku na začátku výpočtu inicializujeme hodnotami `None`. Jakmile funkce *kontroluje*(i, j) zjistí, zda společnost i kontroluje společnost j, tento výsledek `True`/`False` nejen vrátí jako svoji návratovou hodnotu, ale zároveň ho uloží do tabulky jako hodnotu $t[i][j]$. Kdykoliv by funkce *kontroluje*() chtěla provést rekurzivní volání s nějakou dvojicí parametrů (i, k), nejprve se podívá do tabulky t, zda potřebnou hodnotu již nemá spočítanou z dřívějška. Pokud ano, rekurzivní volání nebude provádět, ale použije uloženou hodnotu $t[i][k]$.

Obdobnou úpravu provedeme také v hlavním programu, rovněž tam místo volání funkce *kontroluje*() použijeme hodnotu z tabulky t, pokud ji už známe z předchozího výpočtu.

```
def kontroluje(i, j):  # ověření, zda "i" kontroluje "j"
    if m[i][j] > 50:
        t[i][j] = True
        return True    # přímé vlastnictví (hrana > 50)

    suma = 0
    for k in range(1, n+1):
        suma += m[k][j]
    if suma <= 50:
        t[i][j] = False
        return False   # součet všech vstupních hran <= 50

    suma = 0
    z.append((i, j))
    for k in range(1, n+1):
        if k == i:
            suma += m[i][j]
            continue
        if m[k][j] == 0 or t[i][k] == False:
            continue
        if t[i][k] == True:
            suma += m[k][j]
            continue
```

```
            if (i, k) not in z and kontroluje(i, k):
                suma += m[k][j]
        z.pop()

        if suma > 50:
            t[i][j] = True
            return True
        t[i][j] = False
        return False

n = int(input("Počet společností: "))
m = [[0] * (n+1) for _ in range(n+1)]
                                # matice ohodnocení hran
z = []                          # volací zásobník
t = [[None] * (n+1) for _ in range(n+1)]
                                # tabulka známých hodnot

v = int(input("Počet vlastnických vztahů: "))
print("Vztahy: KDO vlastní KOHO a KOLIK procent")
for h in range(v):
    i, j, p = [int(c) for c in input().split()]
    m[i][j] = p

for i in range(1, n+1):
    for j in range(1, n+1):
        if t[i][j] or (t[i][j] == None and
                                    kontroluje(i, j)):
            print(i, j)    # výsledky: kdo koho kontroluje
```

Popsaná úprava přináší naprosto zásadní úsporu času. Zatímco předchozí řešení úlohy mělo v nejhorším případě exponenciální časovou složitost vzhledem k počtu společností n, nyní bude funkce *kontroluje*() zavolána pro každou dvojici parametrů pouze jednou, takže časová složitost celého řešení je $O(n^2)$.

Doplnění rekurzivního řešení úlohy o pole sloužící k uložení již spočítaných hodnot je známá programovací technika, která se pro zrychlení výpočtu běžně používá a můžete se s ní setkat i při řešení mnoha jiných úloh.