Problém stabilných manželstiev

Našu exkurziu do sveta efektívnych algoritmov začneme tým, že si spolu vyriešime jednu úlohu. Na nej si skúsime ukázať, čo všetko nás u algoritmických úloh bude zaujímať, a čo naopak nie je podstatné. Spominanou prvou úlohou bude tzv. *problém stabilných manželstiev*. Najskôr si ho predstavíme vo veľmi zjednodušenej podobe. Bude nám chvíľu trvať, kým sa dostaneme k jej riešeniu, ale nebojte sa. Na konci zvíťazíme a navyše si ešte cestou ukážeme, že vôbec má zmysel takúto úlohu riešiť.

Teraz sa už ale pustime do sľubovaného zadania úlohy. V našej úlohe máme n mužov a n žien. Každý muž má svoje preferencie, t. j. usporiadané všetky ženy v poradí, v akom by s nimi chcel žiť. Aj každá žena má takéto poradie mužov. My ich teraz pôjdeme vhodným spôsobom popárovať do manželstiev.

Predstavme si, že sme už všetkých mužov popárovali so ženami, a teda máme n manželstiev. *Nestabilnou dvojicou* budeme volať dvojicu (muž M, žena Z), ktorí momentálne nie sú spolu v manželstve, ale muž M by bol radšej so Z ako so svojou aktuálnou partnerkou *a zároveň* by aj žena Z bola radšej s M ako so svojim aktuálnym partnerom. Sadu manželstiev voláme *stabilnou*, ak pre ňu neexistuje žiadna nestabilná dvojica.

A čo je teda našou úlohou? Navrhnúť algoritmus (resp. napísať program), ktorý bude tento problém riešiť. Na vstupe dostaneme vyššie popísané údaje: pre každého muža aj pre každú ženu zoznam ich preferencií. A pre tie má náš algoritmus zistiť, či vôbec nejaká sada stabilných manželstiev existuje, a ak áno, tak nejakú nájsť.

1.1 Kódovanie vstupu

Reálne dáta často prichádzajú v rôznych škaredých podobách: majú rôzne formáty, obsahujú texty v rôznych kódovaniach, a tak ďalej. Ako sme už spomínali, my si často budeme úlohy čo najviac zjednodušovať. V našom prípade nás teda vôbec nebude trápiť, že sa ľudia na vstupe nejako volajú, a už tobôž nebudeme riešiť to, ako rozoznať od seba ľudí, ktorí sa volajú rovnako. Namiesto toho budeme predpokladať, že tieto detaily už niekto vyriešil za nás a mužov aj ženy nám očísloval číslami od 0 po $n-1$.

Jeden možný vstup pre náš program by teda mohol vyzerať napr. nasledovne:

```
3
2 1 0
2 0 1
2 0 1
0 2 1
1 2 0
1 0 2
```

(V prvom riadku je číslo n, v druhom preferencie muža 0, v treťom preferencie muža 1, a tak ďalej. Teda napríklad žena 0 najviac chce muža 0, potom muža 2 a najmenej muža 1.)

1.2 Kontrola riešenia

Skôr, než sa pustíme do riešenia samotnej úlohy (teda hľadania stabilnej sady manželstiev), zamyslime sa nad výrazne jednoduchšou úlohou: *len skontrolovať*, či je konkrétna sada manželstiev stabilná. Na vstupe teda okrem preferencií dostaneme aj zoznam manželských dvojíc a máme zistiť, či existuje nejaká nestabilná dvojica. Ako by sa dala takáto kontrola naprogramovať?

Nejde o príliš ťažký problém. Na jeho vyriešenie nám stačí systematicky prezrieť a skontrolovať všetky dvojice, ktoré teoreticky môžu byť nestabilné.

Jeden možný algoritmus si môžeme v pseudokóde zapísať nasledovne:

```
pre každého muža M:
    pre každú ženu Z inú ako jeho manželka:
        # (ideme skontrolovať, či je dvojica (M,Z) nestabilná)
        1. zisti, či je Z v preferenciách muža M skôr ako jeho manželka
```

```
        2. zisti, či je M v preferenciách ženy Z skôr ako jej manžel
        ak bola v oboch prípadoch odpoveď áno, máme nestabilnú dvojicu, hotovo

ak sme sa dostali až sem a nič sme nenašli, sada manželstiev je stabilná
```

Čitateľ týchto skrípt by mal byť v praktickom programovaní dostatočne skúsený na to, aby dokázal takýto pseudokód „preložiť“ do programu vo svojom obľúbenom programovacom jazyku.

Efektívnejšia kontrola riešenia

FIXME inverzná permutácia.

1.3 Skúšanie všetkých možností

Keď teda už vieme efektívne skontrolovať, či je nejaká sada manželstiev riešením našej úlohy,

Ako u mnohých algoritmických úloh, tak aj u tejto úlohy je jedným z možných spôsobov riešenia použitie *hrubej sily*. V našom prípade môžeme postupne skúšať všetky možné popárovania mužov a žien, a každé z nich skontrolovať.

Aký má takéto riešenie problém? Taký, že je použiteľné len pre veľmi malé hodnoty n. Všetkých možných spôsobov utvorenia manželstiev je totiž veľmi veľa: presne $n!$ (n faktoriál).

Teda napríklad keby sme mali 20 mužov a 20 žien, existovalo by 2 432 902 008 176 640 000 rôznych popárovaní, ktoré by náš program musel vyskúšať. Neskôr si ukážeme, ako približne odhadnúť, ako dlho daný program pobeží. Pre tento program by nám pre $n = 20$ vyšlo, že na súčasnom počítači pobeží rádovo *stotisíce rokov*.

Mimochodom, všimnite si ten obrovský rozdiel: riešenie pre $n = 20$ vieme *skontrolovať* v okamihu, ale *nájsť* nejaké trvá nepredstaviteľne dlho.

V našej úlohe sa ukáže, že problém je v algoritme, ktorým riešenie hľadáme – časom nájdeme iný algoritmus, ktorý riešenie vždy nájde efektívne. Je tomu tak ale vždy, aj u iných problémov? Presnejšie: ak vieme v nejakej úlohe efektívne *overiť*, či je niečo jej riešením, musí vždy existovať aj efektívny algoritmus, ktorý danú úlohu *vie vyriešiť*? Odpoveď na práve položenú otázku zatiaľ nevieme. Úzko súvisí s jedným z najvýznamnejších otvorených problémov súčasnej teoretickej informatiky: otázkou, či $P = NP$.

1.4 Algoritmus „opravujúci chyby“

Ak teda chceme rozumne rýchlo vyriešiť vstup obsahujúci viac ako po 20 mužov a žien, nemôžeme si dovoliť skúšať (ani len zďaleka!) všetky možnosti. Bude treba vymyslieť efektívnejší postup. Skôr, než budete čítať ďalej, skúste sa aj vy nad nejakým zamyslieť.

Ak ste sa naozaj nad úlohou zamysleli, je veľmi pravdepodobné, že vám okrem iného prišla na rozum aj nejaká verzia algoritmu, ktorý môžeme dobre vystihnúť frázou „necháme veciam voľný priebeh“.

Tento algoritmus začneme tým, že ľubovoľne popárujeme mužov so ženami. Teraz spustíme kontrolu, či už máme korektné riešenie. Ak kontrola nenájde žiadnu nestabilnú dvojicu, sme hotoví. A čo ak kontrola nejakú nestabilnú dvojicu (M, Z) nájde? Keď tak veľmi chcú byť spolu, nech si teda sú. Danému mužovi M dáme teda za ženu Z. Tým nám samozrejme ostal jeden muž a jedna žena – doterajší partneri M a Z – ktorí teraz nikoho nemajú. To vyriešime ľahko: dáme aj ich dokopy.

(Na celú takúto operáciu sa môžeme dívať aj tak, že si vlastne dvaja muži vymenili manželky. Alebo naopak, dve ženy vymenili manželov, je to predsa symetrické.)

No a to je všetko – len dokola opakujeme vyššie popísaný postup, až kým nedostaneme situáciu, v ktorej sú už všetci spokojní, teda neexistuje žiadna nestabilná dvojica. To vyzerá dobre, nie?

1.4.1 Nepríjemné otázky

Na prvý pohľad to možno vyzerá, že sme našu úlohu celkom šikovne vyriešili. Je to ale skutočne tak? Skôr, než zaspíme na vavrínoch, položme si niekoľko nepríjemných otázok.

Rieši tento nový algoritmus naozaj zadanú úlohu? Jedno je jasné: *AK* tento algoritmus skončil, *TAK* naozaj práve našiel platné riešenie. Ako dlho mu ale bude trvať, kým skončí? Koľko výmien bude treba spraviť? Zaručí nám niekto, že to tomuto algoritmu nebude trvať tak isto dlho ako riešeniu, ktoré skúšalo všetky možnosti? Alebo ešte dlhšie?

A pri týchto otázkach musíme bohužiaľ smutne sklopiť uši a pripustiť, že odpovede na ne (zatiaľ) nevieme.

My teraz budeme opäť kvôli dramaturgii trochu predbiehať a povieme si, ako to teda je: dosť nanič. Existujú dokonca vstupy, pre ktoré sa vyššie popísaný algoritmus *zacyklí* – po niekoľkých postupných „opravách" nestabilných dvojíc dostane presne tú istú sadu manželstiev ako mal na začiatku. Pre tieto vstupy je teda náš nový algoritmus (nekonečnekrát) horší od algoritmu, ktorý skúša úplne všetky možnosti,

1.5 Praktická aplikácia

Priraďovanie rezidentov na miesta v nemocniciach v USA (a pár iných krajinách): National Residen Matching Program, http://www.nrmp.org/.

„The 2010 Main Residency Match was the largest in NRMP history, encompassing more than 37,000 applicants, 4,100 graduate medical education programs, and 25,500 residency training positions."

Pre takto veľké vstupné dáta už len jediné prezretie všetkých dvojíc (muž, žena) trvá dlho – naivný algoritmus, aj keby fungoval, je zjavne nepoužiteľný.

1.6 Efektívny algoritmus

Inicializácia: všetci muži sú nezadaní, všetky ženy sú nezadané.

Prvé kolo algortmu: Každý muž požiada o ruku ženu, ktorú má prvú v zozname. Ak má niektorá žena viac ako jedného pytača, tomu, ktorého ona chce najviac, povie „možno", ostatných rovno odmietne. Ak žena muža odmietne, ten si ju škrtne zo zoznamu.

Každé ďalšie kolo vyzerá podobne. Aktuálne zadaní muži nerobia nič. Nezadaný muž zájde za prvou neškrtnutou ženou na jeho zozname a požiada ju o ruku. (Môže sa stať, že tá žena je už zadaná. Vtedy sa udeje nasledovné: Ak ona chce nového pytača viac, starého odmietne a novému povie „možno". A naopak, ak viac chce starého pytača, nového odmietne.)

Poznámka: Tento algoritmus navrhli Gale a Shapley už v roku 1962.

1.7 Štandardné otázky

... ktoré teraz treba zodpovedať: Funguje ten algoritmus? A ak áno, ako je efektívny?

V prvom rade dokážeme, že keď algoritmus skončí (t. j. dostane sa do situácie, kedy sa už nič nové nemôže udiať), sú všetci muži popárovaní so ženami.

Sporom. Nech to tak nie je, potom existuje muž M ktorý nikoho nemá, a teda nutne existuje aj žena Z, ktorá tiež nikoho nemá. To, že Z nikoho nemá, znamená, že ju nikdy nikto o ruku nežiadal. A to je hľadaný spor – keďže M nikoho nemá, musel sa už (neúspešne) uchádzať o všetky ženy, vrátane Z.

Tým sme dokázali, že náš algoritmus naozaj nájde nejaké párovanie. Bude ale toto párovanie stabilné?

Opäť sporom. Nech existuje dvojica (M, Z), ktorá chce byť spolu viac ako s partnermi, ktorých im našiel náš algoritmus. Čo nám to o nich hovorí? M skončil s partnerkou, ktorú chce menej ako Z. To ale znamená, že niekedy počas behu algoritmu musel M žiadať Z o ruku. No a keďže nie sú spolu, Z ho musela odmietnuť kvôli niekomu, koho chce ona viac ako M – a to je spor s tým, že Z chce M viac ako svojho súčasného partnera.

Tým sme teda už dokázali, že náš algoritmus naozaj rieši zadanú úlohu. A aký je rýchly? Na to si stačí všimnúť, že každý muž požiada nejakú ženu o ruku nanajvýš N-krát, dokopy sa teda odohrá nanajvýš N^2 akcií.

1.8 Demo na webe

Na http://mathsite.math.berkeley.edu/smp/smp.html je krásne spracovaná ukážka, oplatí sa preklikať si ju.

1.9 Asymetria algoritmu Galea-Shapleyho

Stabilných párovaní môže byť veľa. Náš algoritmus nájde to z nich, ktoré najviac vyhovuje mužom.

Príklad: Muži M_1, M_2, M_3, ženy Z_1, Z_2, Z_3. Ak pre každé i muž M_i najviac chce ženu Z_i, vyrobí náš algoritmus manželstvá $M_i - Z_i$ bez ohľadu na to, čo chcú ženy. Ich muži dokonca môžu byť ich posledné voľby.

V prípade NRMP hrajú rolu „mužov“ nemocnice.