Problem trdnih zakonov

Nika Jerman, Gimnazija Poljane
Nicola Pinzani, Licej F. Prešerna, Trst
Mihael Simonič, Gimnazija Bežigrad
Mentor: Nino Bašić UL FMF

Članom posadke je po koncu MaRSovskega potovanja nekoliko dolgčas, zato se odločijo, da se bodo pozabavali (tj. da si bodo krajšali čas) v parih. A glej ga zlomka, nikakor ne morejo uskladiti svojih želja (prikazane so v tabelici spodaj). Zato uporabijo poseben sistem ... vas morda zanima kakšnega?

1 Opis problema

Imamo skupino n deklet in n fantov, ki jih želimo razporediti v pare. Vsak fant si sestavi svoj seznam naklonjenosti, v katerega razporedi dekleta. Na prvo mesto postavi tisto dekle, s katero bi bil najrajši. Tudi dekleta si naredijo prav take sezname, le da rangirajo fante. Radi bi jih razporedili v pare, tako da ne bodo imeli nobenih možnosti za "prešuštvo". To pomeni, da ne bo takega fanta in takega dekleta, ki ne bi bila v paru, pa bi imel on njo raje kot svojo partnerko in tudi ona njega raje kot svojega partnerja.

1.1 Formalni opis in definicije

Da bomo lahko problem obravnavali z veliko mero matematične strogosti, bomo kar takoj definirali osnovne pojme, s katerimi bomo opisalovali naš problem.

Definicija 1. *Popolno prirejanje* med množico fantov $\mathcal{F}$ in množico deklet $\mathcal{D}$ je množica $\mathcal{P} \subseteq \mathcal{F} \times \mathcal{D}$, ki jo sestavlja n urejenih parov, tako da vsak fant in vsako dekle nastopita v natanko enem paru. (Z drugimi besedami, popolno prirejanje med $\mathcal{F}$ in $\mathcal{D}$ je bijektivna preslikava iz $\mathcal{F}$ v $\mathcal{D}$.)

Vsak fant in vsako dekle ima tudi svojo relacijo "imeti rajši", za katero bomo uporabili oznaki $\succ$ in $\prec$. Ta relacija ustreza zgoraj opisanemu seznamu naklonjenosti. To, da ima fant f dekle d_1 rajši od dekleta d_2 zapišemo takole: $d_1 \succ_f d_2$. Relacija "imeti rajši"je linearna urejenost.

Ovirajoči par v prirejanju sta tak fant f in tako dekle d, da:

- f in d nista v paru,
- f ima rajši dekle d kot svojo partnerko,
- d ima rajši fanta f kot svojega partnerja.

Formalneje to povemo takole:

Definicija 2. Naj bosta $(f, d) \in \mathcal{P}$ in $(\widetilde{f}, \widetilde{d}) \in \mathcal{P}$ različna para v popolnem prirejanju. Če velja $\widetilde{d} \succ_f d$ in $f \succ_{\widetilde{d}} \widetilde{f}$, pravimo, da je $(f, \widetilde{d})$ *ovirajoči par*.

Definicija 3. Prirejanje je *trdno*, če ne vsebuje nobenega ovirajočega para.

Zgled 1. Z mislimi se vrnimo k naši posadki. Njihove naklonjenosti so prikazane v tabelah 1 in 2. Zazrimo se za trenutek v tabeli.

Fant	Seznam naklonjenosti
Gašper	Maja, Nina, Aleksandra, Anja
Dejan	Nina, Aleksandra, Maja, Anja
David	Anja, Maja, Aleksandra, Nina
Uroš	Anja, Nina, Maja, Aleksandra

Tabela 1: Seznami naklonjenosti za fante

Dekle	Seznam naklonjenosti
Anja	Uroš, Dejan, David, Gašper
Maja	Gašper, Dejan, David, Uroš
Nina	Dejan, Uroš, Gašper, David
Aleksandra	Dejan, Uroš, David, Gašper

Tabela 2: Seznami naklonjenosti za dekleta

Gašperjeva relacija "imeti rajši"je takšna (glej tabelo 1):

$$\text{Maja} \succ \text{Nina} \succ \text{Aleksandra} \succ \text{Anja},$$

Anjina relacija "imeti rajši"pa takšna (preberemo jo iz tabele 2):

$$\text{Uroš} \succ \text{Dejan} \succ \text{David} \succ \text{Gašper}.$$

Primer popolnega prirejanja je

$$\{(\text{Uroš}, \text{Anja}), (\text{Gašper}, \text{Aleksandra}), (\text{David}, \text{Nina}), (\text{Dejan}, \text{Maja})\}.$$

Zgornje prirejanje ni trdno, saj obstajajo ovirajoči pari. Tak par je npr. (Dejan, Nina), saj ima Dejan rajši Nino kot Majo, s katero je trenutno žaročen". Nina pa ima Dejana rajši kot Davida.

2 "Naivni"algoritem

Trdno prirejanje lahko poskusimo poiskati tako, da najprej razporedimo vse fante in dekleta v naključne pare. Nato iščemo ovirajoče pare in jih odpravljamo. To naredimo tako, da sestavimo dva nova para. V prvi novi par združimo fanta in dekle, ki sta bila ovirajoči par; v drugi par pa združimo njuna nekdanja partnerja. Postopek se konča, ko ni nobenega ovirajočega para več. Takrat prirejanje postane trdno. Postopek bomo imenovali "naivni"algoritem.

Zgled 2. Prikazali bomo delovanje "naivnega"algoritma na podatkih iz zgleda 1. Najprej razporedimo fante in dekleta v naključne pare, na primer takole:

$$\{(\text{Uroš}, \text{Anja}), (\text{Gašper}, \text{Aleksandra}), (\text{David}, \text{Nina}), (\text{Dejan}, \text{Maja})\}.$$

S pomočjo obeh tabel, ki prikazujejo naklonjenosti, lahko sistematično preverimo, če je prirejanje trdno. Izkaže se, da prirejanje ni trdno, saj ovirajoči pari obstajajo. Prvi tak par je $(\text{Gašper}, \text{Maja})$, ker ima Gašper raje Majo kot Aleksandro, s katero je trenutno v zvezi; Maja pa ima prav tako raje Gašperja kot Dejana, s katerim je sedaj. Ko izvedemo zamenjavo, dobimo novo prirejanje, ki zgleda takole:

$$\{(\text{Uroš}, \text{Anja}), (\text{Dejan}, \text{Aleksandra}), (\text{David}, \text{Nina}), (\text{Gašper}, \text{Maja})\}.$$

Spet preverimo, če je prirejanje že trdno, in ugotovimo, da tudi tokrat obstaja ovirajoči par, tj. $(\text{Dejan}, \text{Nina})$. Po zamenjavi dobimo novo prirejanje:

$$\{(\text{Uroš}, \text{Anja}), (\text{Dejan}, \text{Nina}), (\text{David}, \text{Aleksandra}), (\text{Gašper}, \text{Maja})\}.$$

Tokrat ugotovimo, da je prirejanje trdno.

2.1 Protiprimer za "naivni"algoritem

Pričakovali bi, da se bo med izvajanjem tega algoritma število ovirajočih parov zmanjševalo. Vendar ni vedno tako. Če preizkušamo naivni algoritem na različnih podatkih, kmalu ugotovimo, da ob tem, ko odpravimo en ovirajoči par, lahko nastanejo novi ovirajoči pari. Možno pa si je izmisliti tudi take primere, ko s tem algoritmom ne pridemo do popolnega prirejanja. Algoritem se namreč ne konča, ker pade v tako imenovano *neskončno zanko*. To prikazuje zgled 3.

Zgled 3. Pri teh podatkih, ki so jih prikazujeta tabeli 3 in 4, sicer obstajajo trdna prirejanja, lahko pa se zgodi, da postopek ločitev in ponovnih zarok poteka tako, da se znajdemo v neskončni zanki.

Začetno prirejanje parov naj bo

$$\{(\text{Jaka}, \text{Neža}), (\text{Luka}, \text{Jana}), (\text{Nik}, \text{Anja})\}.$$

Fant	Seznam pref.
Jaka	Jana, Neža, Anja
Luka	Jana, Neža, Anja
Nik	Neža, Jana, Anja

Tabela 3: Seznam preferenc za fante

Dekle	Seznam pref.
Neža	Jaka, Nik, Luka
Jana	Nik, Jaka, Luka
Anja	Nik, Jaka, Luka

Tabela 4: Seznam preferenc za dekleta

Opazimo, da je ovirajoči par (Jaka, Jana). Po zamenjavi dobimo novo prirejanje:

$$\{(\text{Jaka}, \text{Jana}), (\text{Luka}, \text{Neža}), (\text{Nik}, \text{Anja})\}.$$

Naslednji trije koraki so:

- ov. par: (Nik, Jana); novo prirejanje: $\{(\text{Jaka}, \text{Anja}), (\text{Luka}, \text{Neža}), (\text{Nik}, \text{Jana})\}$
- ov. par: (Nik, Neža); novo prirejanje: $\{(\text{Jaka}, \text{Anja}), (\text{Luka}, \text{Jana}), (\text{Nik}, \text{Neža})\}$
- ov. par: (Jaka, Neža); novo prirejanje: $\{(\text{Nik}, \text{Anja}), (\text{Luka}, \text{Jana}), (\text{Jaka}, \text{Neža})\}$

Glej ga zlomka! Po štirih korakih smo spet dobili začetno prirejanje, zato se pri takih izbirah postopek nikoli ne zaključi.

Primer, ki smo ga pravkar videli, nam vzbudi skrb. Ali vedno obstaja trdno prirejanje? Ali je tako prirejanje eno samo, ali jih je morda več? Obstaja kakšen hiter (tj. polinomski) algoritem za iskanje trdnega prirejanja, ki vedno deluje? V naslednjem razdelku, si bomo ogledali algoritem, ki sta ga iznašla D. Gale in L. S. Shapley v sedemdesetih letih 20. stoletja. Pokazali bomo, da vedno deluje in ima polinomsko časovno zahtevnost.

3 Gale-Shapleyev algoritem

Opišimo delovanje tega algoritma. Na začetku so vse osebe nezaročene. Nato fantje v določenem vrstnem redu (ki ga lahko na začetku poljubno izberemo) drug za drugim dvorijo dekletom, in sicer vsak fant tistemu dekletu, ki je trenutno prvo na njegovem seznamu naklonjensti. Dekle ponudbo sprejme, če je še nezaročeno ali če je snubcu bolj naklonjena kot svojemu trenutnemu zaročencu (v tem primeru

prejšnji zaročenec dobi košarico in to dekle zbriše iz svojega seznama). Dekle snubca zavrne, če ima trenutnega zaročenca raje. V tem primeru jo snubec zbriše iz svojega seznama. Po tem je na vrsti naslednji nezaročen fant. Algoritem izvajamo toliko časa, dokler vsak fant ne najde dekleta, ki se je z njim pripravljeno poročiti, ali pa konča s praznim seznamom.

Pokazali bomo, da se algoritem vedno konča in da so na koncu zaročeni vsi fantje in vsa dekleta.

Psevdo-koda Gale-Shapleyevega algoritma (zaradi enostavnosti so vsa dekleta na začetku že zaročena z namišljenim fantom, ki je deležen absolutno najmanjše naklonjenosti):

```
Vhod: seznam deklet, seznam fantov, seznami naklonjenosti
Izhod: trdno prirejanje

for d in dekleta: d.zarocenec = namisljeni fant
while obstaja nezaroceni fant:
  f = nezaroceni fant
  d = f.seznamPreferenc[1] # prvo dekle na seznamu
  if d ima rajsi f kot d.zarocenec:
    del d.zarocenec.seznamPreferenc[1]
    d.zarocenec = f
  else:
    del f.seznamPreferenc[1] # brisanje dekleta s seznama
```

3.1 Implementacija algoritma

Spodaj je implementacija v programskem jeziku Python:

```
def beriPreference(file, n):
    p = {}
    for i in range(n):
        ime, seznam = [x.strip() for x in file.readline().split(":")]
        p[ime] = [x.strip() for x in seznam.split(",")]
    return p

def drugaSmer(p):
    q = {}
    for f in p:
        s = {}
        for g, i in zip(p[f], range(1, n+1)):
            s[g] = i
        q[f] = s
    return q
```

```
def stabilnoPrirejanje(pFant, pDekle):
    qFant = drugaSmer(pFant)
    qDekle = drugaSmer(pDekle)
    pari = {}
    naslDekle = {}
    for f in pFant: naslDekle[f] = 0
    prosti = pFant.keys()
    while len(pari) < n:
        naslKrog = []
        for f in prosti:
            d = pFant[f][naslDekle[f]]
            naslDekle[f] += 1
            if d not in pari:
                pari[d] = f
            elif qDekle[d][f] < qDekle[d][pari[d]]:
                naslKrog.append(pari[d])
                pari[d] = f
            else:
                naslKrog.append(f)
        prosti = naslKrog
    return [(k, pari[k]) for k in pari.keys()]

podatki = open("primer.txt", "r")
n = int(podatki.readline())
pFant = beriPreference(podatki, n) # preberemo preference fantov
pDekle = beriPreference(podatki, n) # preberemo preference deklet

sp = stabilnoPrirejanje(pFant, pDekle)
print("Stabilno prirejanje: ")
for par in sp: print(par)

print("\nZamenjamo vlogo fantov in deklet!")
sp2 = stabilnoPrirejanje(pDekle, pFant)
print("Stabilno prirejanje: ")
for par in sp2: print(par)
```

3.2 Analiza algoritma

Na podlagi analize lastnosti algoritma bomo dokazali, da algoritem deluje pravilno in da v končnem številu korakov pare priredi trdno.

Fant ne more biti istočasno zaročen z dvema različnima dekletoma, saj lahko dvori le, če ni zaročen. Prav tako nobeno dekle ni istočasno v razmerju z dvema fantoma, saj lahko sklene novo zaroko samo, če odslovi svojega trenutnega zaročenca.

Položaj dekleta se v poteku algoritma ne poslabša, saj (če je v razmerju) razdre zaroko le v primeru, da je fantu, ki ji dvori, bolj naklonjena, kot njenemu trenutnemu zaročenecu. Fantu se nasprotno položaj lahko poslabša, saj gre v primeru,

da je zavrjen, do naslednjega dekleta na svojem seznamu želja. V primeru, da je nekemu dekletu bolj naklonjen kot svojemu trenutnemu (to dekle je višje na njegovem seznamu kot njegovo trenutno dekle), ga je to dekle enkrat že zavrnilo, saj fant lahko vsakemu dekletu dvori zgolj enkrat. Kljub temu pa ga algoritem razvrsti tako, da je njegov položaj glede na želje celotne skupine najboljši.

V primeru, da imamo enako število deklet in fantov, ob zaključku algoritma nihče ne ostane brez para, saj se seznami fantov nikoli ne spraznijo. To lahko dokažemo s protislovjem: če obstaja fant s praznim seznamom, so ga vsa dekleta bodisi zavrnila, bodisi zapustila. Vemo, da se položaj deklet skozi algoritem ne poslabša, iz cesar sledi, da so vsa dekleta zaročena. Obenem vemo tudi, da noben izmed fantov ne more biti zaročen z dvema dekletoma hkrati. Iz obojega sledi, da bi potem poleg fanta s praznim seznamom moralo obstajati več fantov kot deklet, kar pa vemo, da ni res. Podobno lahko dokažemo, tudi da se algoritem zaključi v koncno mnogih korakih. Že zaročeno dekle ne more ostati brez zaročenca. Zamenja ga lahko zgolj s fantom, ki mu je bolj naklonjena. Ker pa vsak nezaročen fant po vrsti zaprosi vsa dekleta, se tudi on slej ko prej zaroči. Ker imamo enako število deklet in fantov, se tako končno poročijo vsi.

Vsi nastali pari so takšni, da je prirejanje trdno. Predpostavimo, da ima fant f rajši dekle d kot svojo zaročenko. Potem jo je fant f nekoč že zaprosil in ga je to (kot smo že dokazali) bodisi zavrnilo bodisi kasneje zapustilo. Torej je takrat imelo ali ji je dvoril fant, ki ga je imela rajši od fanta f. Torej tak fant f ne pokvari trdnosti zarok. Podobno velja za dekle d_1, ki ima rajši fanta f_1 kot pa trenutnega zaročenca.

Če zamenjamo vloge fantov in deklet tako, da dekleta izbirajo prva, ugotovimo, da lahko (odvisno od primera) dobimo drugačne pare. Sledeč primer bo razjasnil, da je na boljšem tisti, ki prvi izbira:

Fantje	seznam	Dekleta	seznam
A	C, D	C	B, A
B	D, C	D	A, B

Vidimo, da v primeru, ko imajo prednost pri izbiranju fantje, dobimo pare AC in BD, ko izbirajo dekleta, pa dobimo pare AD in BC.

4 Zaključek

Gale in Shapley sta bila prva, ki sta ta algoritem matematično analizirala, vendar je bil Gale-Shapleyev algoritem v uporabi že prej. V Ameriki so ga uporabili za razporejanje podiplomskih študentov medicine v bolnišnice, kjer so študentje opravljali specializacijo. Vsak študent je sestavil preferenčni seznam bolnišnic in tudi bolnišnice so na podlagi ocen in intervjujev s študenti sestavile preferencne sezname študentov.

Obstaja še mnogo podobnih problemov, npr. *problem cimrov*, kjer moramo $2n$ ljudi razporediti v n dvoposteljnih sob. Tukaj oseb ne razdelimo na fante in dekleta. V isti sobi sta lahko katerikoli dve osebi.

Literatura

[1] Harry Mairson: *The Stable Marriage Problem*. The Brrandeis Review, Vol. 12, No. 1, 1992.

[2] M. Juvan: *Problem trdnih zakonov*. Presek, letnik 23, št. 4, 1995-1996.

[3] Sodelavci Wikipedije: *Stable marriage problem* (`http://en.wikipedia.org/wiki/Stable_marriage_problem`). Citirano 17.8.2010.