PRESEK
List za mlade matematike, fizike, astronome in računalnikarje

ISSN 0351-6652
Letnik **23** (1995/1996)
Številka 4
Strani 212–215

Martin Juvan:

PROBLEM TRDNIH ZAKONOV

Ključne besede: računalništvo, računalniško programiranje, urejanje seznamov.

Elektronska verzija: http://www.presek.si/23/1266-Juvan.pdf

PROBLEM TRDNIH ZAKONOV

Obstaja nekaj problemov, ki jih na začetku poskuša rešiti vsak zagnan programer. Kdo še ni poskušal poiskati prvih sto praštevil, postaviti kraljic na šahovsko desko tako, da se ne napadajo, izpisati vseh permutacij ali pa s skakačem obiskati vseh polj šahovnice?

Tudi naloga o trdnih zakonih je podobne vrste. Morda je nekoliko manj znana, čeprav jo najdemo v mnogih knjigah, ki obravnavajo programiranje. Opisana je tudi v prvem delu Wirthovega Računalniškega programiranja[1]. Naloga zahteva naslednje: Dani sta skupini n fantov in n deklet. Naša naloga je, da sestavimo n (zakonskih) parov, in sicer tako, da bodo sklenjeni zakoni trdni. Vsak fant in vsako dekle namreč sestavi svoj seznam naklonjenosti do oseb nasprotnega spola. Zakoni so trdni, če ne obstajata fant F in dekle D, ki med seboj nista poročena, a je fant F bolj naklonjen dekletu D kot svoji ženi, dekle D pa ima fanta F rajši od svojega moža.

Poglejmo primer: Fantje naj bodo Aleš, Igor in Miha, dekleta pa Breda, Karin in Neli. Seznami naklonjenosti so naslednji:

Aleš : Breda, Karin, Neli	Breda : Igor, Miha, Aleš
Igor : Neli, Breda, Karin	Karin : Aleš, Igor, Miha
Miha : Breda, Neli, Karin	Neli : Miha, Aleš, Igor

Pri tem je na začetku seznama oseba, ki jo imajo najrajši, z osebo na koncu pa bi se poročili le v skrajni sili. Zakoni Aleš–Karin, Igor–Breda in Miha–Neli naj bi bili trdni, medtem ko zveze Aleš-Breda, Igor–Karin in Miha–Neli ne, saj ima Igor rajši Bredo kot Karin, Breda pa je bolj naklonjena Igorju kot Alešu.

Če imamo n fantov in n deklet, jih lahko med seboj poročimo na $n(n-1)\cdot\ldots\cdot 1 = n!$ različnih načinov. Nekatere izbire bodo trdnejše od drugih. Ni pa takoj jasno, da pri poljubnih seznamih naklonjenosti vedno obstajajo trdni zakoni.

V nadaljevanju prispevka bomo opisali postopek, ki bo kot vhod dobil sezname naklonjenosti za fante in dekleta ter vrnil trdne zakonske zveze. Pri tem bomo tudi dokazali, da trdni zakoni vedno obstajajo.

Iskanje trdnih zvez poteka takole: Na začetku nobena oseba ni zaročena. Dokler niso vsi fantje zaročeni, ponavljamo. Izberemo nezaročenega fanta. Ta med dekleti, ki jih še ni zaprosil, zaprosi tisto, ki ji je najbolj

[1] N. Wirth, *Računalniško programiranje*, 1. del, DMFAS, Ljubljana, 1989.

naklonjen. Če dekle ni zaročeno ali pa ima prosilca raje od svojega trenutnega zaročenca, razdre morebitno zaroko in se zaroči s prosilcem. Ko so vsi fantje zaročeni, so pari sestavljeni, zato le še sklenemo zakone.

Gornji postopek bomo zapisali v obliki podprograma v pascalu. Glavni program, ki poskrbi za vnos podatkov in privlačen izpis (ter tolažbo za nesrečno, čeprav trdno poročene), pa bomo izpustili. Predno se lotimo programiranja, se moramo domeniti, kako bomo v podprogramu predstavili podatke. Kot običajno, bomo fante in dekleta predstavili s števili od 1 do n. Podprogram `TrdniZakoni`, s katerim sestavljamo pare, bo imel štiri parametre: tri vhodne in enega izhodnega, v katerem bomo vrnili sestavljene pare. Vhodni podatki so število fantov oziroma deklet `n` ter tabeli `f` in `d`, s katerima podamo sezname naklonjenosti fantov in deklet. Tako bo `f[i][j]` število, ki predstavlja dekle, ki je v seznamu naklonjenosti fanta `i` na mestu `j`. Podobno bo `d[i][j]` fant, ki ga ima `i`-to dekle na `j`-tem mestu v svojem seznamu naklonjenosti. Pare bomo vrnili v tabeli `z`. Pri tem bo `z[i]` fant, s katerim poročimo `i`-to dekle.

```
const
  maxN = 50;                          { največje število fantov oziroma deklet }
type
  seznam = array[1..maxN] of integer;
  tabela = array[1..maxN] of integer;          { podatkovna struktura za }
                                                 { sezname naklonjenosti }

procedure TrdniZakoni(n: integer; vzr f,d: tabela; var z: seznam);
{ Sklene trdne zakone glede na sezname naklonjenosti f in d }
  var
    nasl: seznam;          { Fant i bo naslednjič zaprosil dekle f[i][nasl[i]]. }
    stp: integer;                    { število trenutno nezaročenih fantov }
    prosti: seznam;                  { seznam trenutno nezaročenih fantov }
    i,j,k: integer;                                  { pomožni števci }
begin
  stp := n;                          { Na začetku noben fant ni zaročen. }
  for i:=1 to n do begin
    prost[i] := 1;                   { Na začetku noben fant ni zaročen. }
    z[i] := 0;                         { Tudi nobeno dekle ni zaročeno. }
    nasl[i] := 1;      { Fantje najprej zaprosijo dekleta, ki jih imajo najraje. }
  end;
  while stp>0 do begin      { Dokler ne zaročimo vseh fantov, ponavljamo. }
    i := prosti[stp];                      { Izberemo nezaročenega fanta. }
    j := f[i][nasl[i]];                          { Fant i zaprosi dekle j. }
    k := 1;          { Ali ima dekle j rajši fanta i kot svojega zaročenca z[j]? }
    while (d[j][k]<>i) and (d[j][k]<>z[j]) do k := k+1;
                                          { Opomba: lahko je z[j]=0 }
```

```
    if d[j][k]=1 then begin
      if z[j]<>0 then                                  { razdremo zaroko }
        prost[stp] := z[j]          { Fant, ki je bil zaročen z j-tim dekletom, }
                                                       { ni več zaročen. }
      else
        stp := stp-1;                                  { še en srečen par več }
      z[j] := i;                                       { zaroka }
    end; { if }
    nasl[i] := nasl[i]+1;          { Naslednjič bo fant i zaprosil naslednje dekle }
                                                       { s seznama. }
  end; { while stp>0 }
end; { TrdniZakoni }
```

Gornji podprogram je kar kratek in tudi programersko ne prezahteven, še vedno pa nismo utemeljili, da se izteče, da se torej zunanja zanka `while` konča in da so sklenjeni zakoni res trdni.

Prepričajmo se najprej, da podprogram ne obtiči v neskončni zanki. Ves čas postopka je število nezaročenih fantov enako številu nezaročenih deklet, saj vsako dekle in vsakega fanta zaročimo kvečjemu z eno osebo nasprotnega spola. Nobeno zaročeno dekle tudi ne postane nezaročeno, morda le zamenja zaročenca z novim fantom, ki mu je bolj naklonjena. Ker pa vsak nezaročen fant po vrsti zaprosi vsa dekleta, se vsako dekle slej ko prej zaroči. Torej se zaročijo in končno tudi poročijo vsa dekleta in vsi fantje.

Nekoliko težje se je prepričati, da so sklenjeni zakoni res trdni. Običajno za vsako pomembno zanko v programu poiščemo invarianto, ki se pri ponovitvah zanke ohranja. To je lastnost ali skupina lastnosti, ki nam pomaga, da pokažemo pravilnost delovanja. V našem primeru je invarianta zunanje zanke `while` lastnost, da so ves čas izvajanja zanke sklenjene zaroke trdne pri pogoju, da nezaročenih oseb ne upoštevamo. Na začetku to gotovo drži, saj nimamo nobenega zaročenega para. Preveriti moramo še, da se v vsaki ponovitvi zanke ta lastnost ohrani. Recimo, da v zanki spremenimo zaročene pare. Naj bosta fant F in dekle D novo zaročeni par. Ker so vsi ostali zaročeni pari ostali nespremenjeni, mora biti med morebitnima nesrečnima osebama, ki porušita trdnost zarok, bodisi fant F bodisi dekle D. Ločimo dve možnosti:

a) Recimo, da ima fant F rajši dekle D_1, ki je poročena s fantom F_1, kot svojo zaročenko D. Potem je fant F nekoč že zaprosil dekle D_1 in ta ga je zavrnila. Torej je takrat imela zaročenca, ki ga je imela rajši od fanta F. Ker dekleta zamenjajo zaročence le s fanti, ki jih imajo še rajši, ima dekle D_1 tudi svojega trenutnega zaročenca F_1 rajši od fanta F. Fant F torej ne pokvari trdnosti zarok.

b) Naj ima dekle D rajši fanta F_1, ki je zaročen z dekletom D_1, kot pa svojega zaročenca F. Potem fant F_1 še ni zaprosil dekleta D. Ker pa je že zaprosil dekle D_1, saj je z njo zaročen, jo ima rajši kot dekle D. Torej tudi dekle D ne more porušiti trdnosti zarok.

Vse zaroke so torej trdne tudi po koncu zunanje zanke `while`. Potem pa so trdni tudi zakoni, ki nastanejo iz njih.

Če vas zgornji argumenti niso prepričali, lahko trdnost sklenjenih zakonov preverite tudi s spodnjim funkcijskim podprogramom. Seveda bi bilo treba tudi tokrat pokazati, da podprogram deluje pravilno, a tako pikolovski vendarle ne bomo.

```
function Preveri(n: integer; var f,d: tabela; var z: seznam): boolean;
{ Preveri, ali so zakoni, podani s tabelo z, trdni. Če so, vrne TRUE, }
{ sicer pa FALSE. }
  var
    trdni: boolean;
    i,j,k,l: integer;
begin
  trdni := true;
  i := 1;          { Za vsakega fanta z[i] preverimo, ali bi zapustil svojo ženo i. }
  while trdni and (i<=n) do begin
    j := 1;          { Za vsa dekleta, ki jih ima fant z[i] rajši kot svojo ženo, }
    while trdni and (f[z[i]][j]<>i) do begin   { preverimo, ali bi tudi dekle }
      l := f[z[i]][j];                  { zapustilo svojega moža zaradi fanta z[i]. }
      k := 1;          { Ali ima dekle l rajši fanta z[i] kot svojega moža z[l]? }
      while (d[l][k]<>z[i]) and (d[l][k]<>z[l]) do k := k+1;
      trdni := d[l][k]=z[l];
      j := j+1;
    end;
    i := i+1;
  end;
  Preveri := trdni;
end; { Preveri }
```

Zgodbe pa tu še ni konec. Podprogram `TrdniZakoni` lahko uporabimo tudi za drugačne naloge in ne le za sestavljanje trdnih zakonov. Tako lahko na primer z njim sestavimo pare za ples, športno ekipo, pri čemer igralci navedejo svoja priljubljena igralna mesta, hkrati pa poznamo njihovo uspešnost na posameznih položajih, opravimo razdelitev daril (ali pa domačih nalog) in še bi lahko naštevali.

Seveda pa ima reševanje z opisanim postopkom tudi svojo slabo stran. Opisani pristop namreč zahteva, da seznamov naklonjenosti potem, ko jih določimo, ne smemo več spreminjati. To pa je pri nalogah iz vsakdanjega življenja zelo redko.

Martin Juvan