Netvarkingas bibliotekininkas

Iš pažiūros algoritmas nėra sunkus: gan aišku, kaip atrasti vienokią ar kitokią klaidą ir kaip ją ištaisyti. Tačiau gali būti įvairiausių klaidų atvejų, jos persipina vienos su kitomis, algoritmas tampa ilgas, painus ir ... kai kuriais atvejais klaidingas. Autoriai nemažai vargę prie šio uždavinio nėra tikri, kad numatė absoliučiai visus atvejus...

Algoritmą suskaidysime į dalis ir kiekvieną dalį aiškinsime atskirai. Uždavinio sprendimas skyla į dvi dalis: pirma pagal duotas lenteles bandoma dėlioti knygas į lentynas įsimenant ką ir kaip pavyko išdėlioti, po to remiantis turimais duomenimis taisomos klaidos.

Pradėkime nuo tipų aprašų bei globalių kintamųjų:

```
const MAX_L = 100;   { maksimalus leidinių skaičius }
      MAX_P = 2000;  { maksimalus pozicijų skaičius }
type lentelė = array [1..MAX_L] of integer;
     lentyna = array [1..MAX_P] of integer;
     duomenys = array [1..MAX_L]
              of record
                   kiek,  { kiek leidinio tomų pavyko išdėlioti }
                   pask: integer;  { paskutinio padėto tomo vieta }
                 end;
{ žemiau surašyti globalieji kintamieji }
var n,   { leidinių skaičius }
    pozicijos: integer;  { pozicijų skaičius lentynoje }
    vieta, skaič: lentelė;  { pirmoji lentelė }
    perr2: lentyna;         { antroji lentelė }
```

Visi šio algoritmo pradiniai duomenys pateikti globaliais kintamaisiais. Jų reikšmės darbo metu nekeičiamos, be to jų yra nemažai ir jų reikia vos ne kiekvienai nedidelei procedūrai ar funkcijai, todėl pamanyta, kad bus patogiau jei nereikės kiekvienai procedūrai perdavinėti krūvos duomenų.

Atstatyti yra svarbiausia procedūra, sprendžianti uždavinį. Detalizuosime procedūrą ir joje esančius kreipinius į kitas procedūras.

Perrašinėjant lentelę gali būti padaryta įvairiausių klaidų, tačiau viena jų ypač nemaloni: tai nuoroda į neegzistuojančią poziciją. Pavyzdžiui, lentyna turi 15 vietų knygoms, o klaidingai perrašytame langelyje įrašytas skaičius 20. Klaida nemaloni dėl to, kad daugelyje situacijų tektų tikrinti ar neišeinama už lentelės ribų. Todėl pirmasis veiksmas – patikrinti, ar nėra nuorodos už lentelės ribų ir jei taip, šią klaidą pakeisti kita: vietoj netinkamo skaičiaus įrašyti nulį. Šios klaidos ieško funkcija *už_ribų.*

Atsižvelgus į šią klaidą dėliojame knygas. Dėliojimo rezultatas – dvi lentelės. Knygas dėliojame lentelėje *lent.* Imame po vieną leidinį ir dedame to leidinio tomus, kol jie pasibaigia (pasiekiame -1). Nustojame dėti leidinio tomus ir tada, kai randame klaidą (reikia dėti knygą į langelį, kuriame įrašytas 0 arba dėti knygą į vietą, kurioje jau padėta knyga).

$$lent[i] = \begin{cases} l, \text{ jei toje vietoje padėtas } l\text{-tojo leidinio tomas;} \\ 0, \text{ jei šioje vietoje antrojoje lentelėje buvo įrašytas 0;} \\ -2, \text{ jei šioje vietoje nepadėta knyga;} \end{cases}$$

Beje, dėdami knygas, netikriname, ar padėtų tomų skaičius lygus leidinio tomų skaičiui.

Sąlygoje pateiktam antram pavyzdžiui ši lentelė atrodytų taip:

Vieta lentynoje	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Leidinys	1	1	2	1	-2	2	2	3	2	3	4	0	0	0

Lentelėje *leid* įsimenami duomenys apie kiekvieną leidinį: kiek jo tomų pavyko išdėlioti lentynoje ir kokioje vietoje yra paskutinis sėkmingai padėtas tomas.

Išdėliojus knygas, einama prie klaidų paieškos. Klaidų analizę pateiksime vėliau. Tada ir paaiškinsime, kodėl pakanka patikrinti dvi sąlygas norint sužinoti, ar lentelė perrašyta teisingai. Nustačius, kad yra klaida, kreipiamasi į procedūras *ieškoti_skaičiaus* ir *ieškoti_vietos*, kurios ieško klaidos pozicijos ir teisingos reikšmės toje pozicijoje.

```
procedure atstatyti (var yra: boolean;
                     var kl_vieta, kl_sk: integer);
{ patikrina, ar antroje lentelėje yra klaida ir jei galima, ją atstato }
  var l, poz: integer;
      leid: duomenys;
      lent: lentyna;
begin
  { jei yra nuorodą į neegzistuojančią poziciją, šią klaidą keičiame kita }
  poz := už_ribų;
  if poz > 0
     then perr2[poz] := 0;
  { išdėliojame knygas }
  dėlioti (leid, lent);
  yra := false;
  for l := 1 to n do
    if (leid[l].kiek <> skaič[l]) or
       (perr2[leid[l].pask] <> -1)
       then yra := true
       else begin
              ieškoti_skaičiaus (kl_sk);
              ieškoti_vietos (lent, leid, kl_vieta)
            end;
end; { atstatyti }
```

Funkcija *už_ribų* bei procedūra *dėlioti* yra akivaizdžios: jų nekomentuosime.

```
function už_ribų: integer;
{ tikrina, ar nėra nuorodos į neegzistuojančią poziciją }
  var i, poz: integer;
begin
  poz := 0;
  for i := 1 to pozicijos do
    if (perr2[i] < -1) or (perr2[i] > pozicijos)
        then poz := i;
  už_ribų := poz
end; { už_ribų }


procedure dėlioti (var leid: duomenys;
                   var lent: lentyna);
{ bando dėlioti knygas lentynoje }
{ leid – kiekvieno leidinio išdėliotų tomų skaičius ir paskutinio tomo vieta, }
{ lent – rodo kurioje pozicijoje kurio leidinio tomai yra }
  var tomas, buvo, l, i, poz: integer;
```

```
begin
  { inicializuojame masyvus leid ir lent }
  for l := 1 to n do
    begin
      leid[l].kiek := 0;
      leid[l].pask := 0;
    end;
  for i := 1 to pozicijos do
    if perr2[i] = 0
       then lent[i] := 0
       else lent[i] := -2;     { nei viena pozicija lentynoje neužimta }
  { dėliosime kiekvieno leidinio tomus }
  for l := 1 to n do
    begin
      buvo := 0;              { buvusi pozicija }
      poz := vieta[l];        { pirmojo tomo vieta }
      tomas := 0;             { kiek tomų sudėliota }
      while (poz <> -1) and (poz <> 0) do
        { kol nepriėjome paskutinio tomo ar nuorodos į nulinę poziciją }
        if (poz > 0)
        then if lent[poz] <> -2  { priėjome užimtą vietą }
             then poz := 0
             else begin
                    lent[poz] := l;  { šioje vietoje yra leidinys l }
                    tomas := tomas + 1;
                    buvo := poz;
                    poz := perr2[poz];
                  end;
      { įsimename duomenis apie l-tąjį leidinį }
      leid[l].kiek := tomas;
      leid[l].pask := buvo;
    end
end;  { dėlioti }
```

Čia baigiasi pirmoji algoritmo dalis: turėdami reikiamus duomenis pereiname prie klaidų analizės.

Procedūra *ieškoti_skaičiaus* randa koks skaičius turėjo būti įrašytas klaidingai perrašytame langelyje.

Perrašant lentelę buvo padaryta klaida: vienas skaičius pakeistas kitu skaičiumi. Tarkime, kad klaidingai perrašytoje pozicijoje buvo įrašytas skaičius sk_1. Perrašant suklysta ir jis pakeistas kitu skaičiumi.

Iš sąlygos žinome, kad *nulis* negalėjo būti pakeistas kitu skaičiumi, t. y. $sk_1 \neq 0$. Jei $sk_1 = -1$, tai perrašytoje lentelėje *minus vienetų* skaičius nebelygus leidinių skaičiui.

Liko atvejis, kai $sk_1 > 0$. Visi natūralieji skaičiai antroje lentelėje reiškia lentynos poziciją. Į kiekvieną lentynos poziciją (išskyrus tas kurios yra tuščios) turi būti nuoroda, t. y. tos pozicijos numeris turi būti įrašytas arba pirmoje arba antroje lentelėje. Jei sk_1 pakeičiamas kitu skaičiumi, tuomet į poziciją sk_1 nebėra nuorodos, t. y. šios pozicijos numeris nėra įrašytas nei pirmoje, nei perrašytoje lentelėje.

Remdamiesi šiais samprotavimais rašome procedūrą *ieškoti_skaičiaus*. Joje yra kreipiniai į funkciją *kiek* ir procedūrą *nuoroda*. *Kiek*(*x*) suskaičiuoja kiek kartų skaičius *x* pasikartoja perrašytoje lentelėje. Procedūra *nuoroda* skaičiuoja kiek kartų pirmoje ir antroje (perrašytoje) lentelėse sutinkama nurodyta pozicija.

```
procedure ieškoti_skaičiaus (var kl_sk: integer);
{ ieško teisingos reikšmės }
  var i, sk, n1, n2: integer;
begin
  if kiek(-1) < n
     then kl_sk := -1
     else begin
            for i := 1 to pozicijos do
              begin
                if perr2[i] <> 0
                   then nuoroda (i, sk, n1, n2);
                if sk = 0 then kl_sk := i
              end;
          end;
end;  { ieškoti_skaičiaus }
```

Pateikiame funkciją *kiek*.

```
function kiek (x: integer): integer;
{ kiek kartų skaičius x pasikartoja perrašytoje lentelėje }
  var i, sk: integer;
begin
  sk := 0;
  for i := 1 to pozicijos do
    if perr2[i] = x
       then sk := sk + 1;
  kiek := sk
end;  { kiek }
```

Keletas komentarų apie procedūrą *nuoroda*. Ji randa ne tik nuorodų skaičių į duotą poziciją *x*, bet ir pačias nuorodas, t. y. antrosios lentelės langelių numerius, kuriuose įrašytas *x*. Padarius vieną klaidą antroje lentelėje tokių nuorodų gali būti ne daugiau kaip dvi.

Beje, jei į poziciją *x* rodoma iš pirmosios lentelės, tuomet ta nuoroda žymima neigiamu skaičiumi ir reiškia leidinio numerį, kurio pirmas tomas turi būti pozicijoje *x*. Kadangi pirmoji lentelė yra teisinga, iš jos gali būti tik viena nuoroda į *x*.

```
procedure nuoroda (x: integer;
                   var sk,  { langelių skaičius }
                   n1,n2: integer { iš kurių langelių rodoma į x });
{ ieško kuriame langelyje įrašytas skaičius x }
{ nuorodą iš pirmosios lentelės žymėsime neigiamu skaičiumi }
  var poz, i, l: integer;
begin
  n1 := 0; n2 := 0;
  { tikrina, ar nėra nuorodos iš pirmosios lentelės }
  for l := 1 to n do
    if vieta[l] = x
       then n1 := -l;  { nuorodą iš pirmosios lentelės }
                 { susitarkime žymėti neigiamu skaičiumi }
  { tikrina, ar nėra tokios nuorodos antroje lentelėje }
  for i := 1 to pozicijos do
    if (perr2[i] = x)
       then if n1 = 0 then n1 := i
                      else n2 := i;
  sk := ord (n1 <> 0) + ord (n2 <> 0)
end;  { nuoroda }
```

Radę teisingą reikšmę, bandysime ieškoti klaidos vietos. Pradžioje išanalizuosime galimas klaidas bei jų pasekmes.

Imkime sąlygoje pateikto pavyzdžio pirmuosius du leidinius. Pagal antrąją lentelę galima sudaryti tokias grandines:

1 leidinys: $1 \rightarrow 4 \rightarrow 2 \rightarrow 5 \rightarrow -1$

2 leidinys: $3 \rightarrow 6 \rightarrow 7 \rightarrow 9 \rightarrow -1$

Sudarysime galimų klaidų bei jų pasekmių lentelę. Daugtaškiu žymėsime nutrūkusią grandinę.

Klaidos apibūdinimas	Nr	Pavyzdžiai	Pasekmė
Skaičius pakeičiamas 0	1	1 → 4 → 0 ....→ 5 → -1	Ketvirtame langelyje įrašius 0, nulių skaičius tampa vienetu didesnis, nei turi būti, atsiranda nuoroda į nulį (pirmasis langelis nukreipia į ketvirtą, kuriame įrašytas 0), t. y. grandinė baigiasi nekorektiškai (ne -1, pirmojo leidinio sėkmingai išdėliojami ne visi tomai, lieka nutrūkęs grandinės galas;
	2	0 → 4 → 2 .→ 5 → -1	Tas pats, kas praeitame pvz, tik nepadedamas nei vienas tomas;.
	3	1 → 4 → 2 .→ 5 → 0	Tas pats, kas praeitame pvz, tik sumažėja -1 skaičius ir nėra grandinės galo
Skaičius pakeičiamas -1	4	1 → 4 → -1... .→ 5 → -1	Padidėja -1 skaičius, lieka nutrūkęs grandinės galas, išdėliojami ne visi leidinio tomai;
	5	-1... → 4 → 2 .→ 5 → -1	Tas pats, tik nepadedamas nei vienas tomas;
Skaičius keičiamas skaičiumi			
grandinė užsiciklina	6	1 → 4 → 2 .→ 4 .... → -1	Nepavyksta išdėlioti visų tomų, lieka nutrūkęs grandinės galas, grandinė baigiasi nekorektiškai (ne -1), į kažkurią poziciją egzistuoja dvi nuorodos;
	7	1 → 4 → 2 .→ 5 → 4	Visi tomai išdėliojami, tačiau padidėja -1 skaičius ir grandinė baigiasi nekorektiškai, į kažkurią poziciją egzistuoja dvi nuorodos;
nuoroda į nulinį langelį	8	1 → 14... → 2 .→ 5 → -1	Lieka nutrūkusi grandinė, išdėliojami ne visi tomai, grandinė baigiasi nekorektiškai, atsiranda nuoroda į nulį,
	9	1 → 4 → 2 .→ 5 → 14	Nelieka nutrūkusios grandinės, išdėliojami visi tomai, tačiau grandinė baigiasi nekorektiškai, atsiranda nuoroda į nulį
Susilieja dvi grandinės	10	1 → 4 → 2 .→ 5 → -1 3 → 2.... → 7 → 9 →> -1	*Jei pirma dedami pirmojo leidinio tomai*: Lieka nutrūkusi grandinė, pirma grandinė išdėliojama korektiškai, antra – nekorektiškai, į kažkurią poziciją egzistuoja dvi nuorodos;;
	11		*Jei pirma dedami antrojo leidinio tomai*: Lieka nutrūkusi grandinė, pirma grandinė išdėliojama nekorektiškai, antra korektiškai, tačiau išdėliojama per mažai antrosios tomų, į kažkurią poziciją egzistuoja dvi nuorodos;
	12	1 → 4 → 2 .→ 5 → 3 3.. → 6 → 7 → 9 → -1	Lieka nutrūkusi grandinė, pirma grandinė išdėliojama nekorektiškai, antrosios – nepadedamas nei vienas tomas, į kažkurią poziciją egzistuoja dvi nuorodos;
	13	1 → 4 → 7 .→ 5 → -1 3 → 6 → 7 ... → 9 → -1	Lieka nutrūkusi grandinė, viena grandinių išdėliojama nekorektiškai. Tačiau bendros abiejų grandinių dalies ilgis lygus pamestos grandinės ilgiui, todėl nustatyti klaidos vietos neįmanoma, į kažkurią poziciją egzistuoja dvi nuorodos;
Pametama grandinės vidinė dalis	14	1 → 4 → 5 → -1 2 → 5	Lieka pamesta grandinės dalis, padėtų tomų skaičius nelygus leidinio tomų skaičiui, egzistuoja dvi nuorodos į tą patį laukelį

Kiekviena klaida sukelia įvairiausių pasekmių (lieka nutrūkęs grandinės galas, nekorektiškai baigiasi grandinė, padidėja nulių skaičius ir t. t.). Klaidas galima įvairiausiai klasifikuoti pagal pasekmes, tas pasekmes identifikuoti ir pagal tai ieškoti klaidos vietos. Mes pasirinkome vieną variantą, kuris mums atrodė priimtiniausias.

Pirmiausia atkreipkite dėmesį, kad kokia bebūtų padaryta klaida, arba nors viena grandinė baigsis nekorektiškai arba kurio nors leidinio išdėliotų tomų skaičius nebus lygus tikrajam to leidinio tomų skaičiui. Užtenka patikrinti šias dvi sąlygas, kad pasakytum, ar perrašant lentelę buvo padaryta klaida, ar ne. Tai ir daroma procedūroje *atstatyti*.

Procedūroje *ieškoti_vietos* klaidas suskirstėme į tris grupes:

1) yra nuoroda į nulinį elementą (1, 2, 3, 8, 9 pavyzdžiai);

2) į kažkurią poziciją egzistuoja dvi nuorodos (6, 7, 10, 11, 12, 13, 14) pavyzdžiai;

3) kai skaičius pakeičiamas *minus vienetu* (4, 5 pavyzdžiai).

Šios trys grupės nepersikerta ir išsemia visus galimus atvejus.

```
procedure ieškoti_vietos (lent: lentyna; leid: duomenys;
                          var kl_vieta: integer);
{ ieško pozicijos, kurioje padaryta klaida }
  var poz, i, l, sk, n1, n2: integer;
begin
  kl_vieta := 0;
  { gal būt yra nuoroda į nulinį elementą }
  ieškoti0 (kl_vieta);
  if kl_vieta = 0    { jei tokios klaidos nerasta }
     then begin      { ieškome kitų klaidų }
             { gal ta pati pozicija įrašyta į du langelius }
             sk := 0; i := 1;
             while (sk <> 2) and (i <= pozicijos) do
               begin
                 if perr2[i] <> 0
                    then nuoroda (i, sk, n1, n2);
                 i := i + 1;
               end;
             if sk = 2
              then du_langeliai(lent,leid,n1,n2,kl_vieta)
              else galas (leid, kl_vieta);
          end;
end;  { ieškoti_vietos }
```

Imame pirmąjį atvejį: nuoroda į nulinį elementą. Atkreipkite dėmesį, kad šiuo atveju terminas *nuoroda* suprantamas kitaip. Tai reiškia ne langelio, kuriame įrašytas *nulis*, numerį, bet langelio, kuriame įrašytas pozicijos (kurioje įrašytas *nulis*) numeris. Pirmiausia reikia patikrinti, ar yra nuoroda į *nulį*. Klaida gali būti padaryta dvejopa. Imkime sąlygoje pateiktą antrąją lentelę:

1	2	3	4	5	6	7	8	9	10	11	12	13	14
4	5	6	2	-1	7	9	-1	-1	8	-1	0	0	0

Joje galime skaičių pakeisti *nuliu* (klaidingai perrašytas langelis patamsintas):

1	2	3	4	5	6	7	8	9	10	11	12	13	14
4	5	6	0	-1	7	9	-1	-1	8	-1	0	0	0

arba galime vietoj nuorodos (ar *minus vieneto*) įrašyti nuorodą į nulinį langelį:

1	2	3	4	5	6	7	8	9	10	11	12	13	14
4	5	14	2	-1	7	9	-1	-1	8	-1	0	0	0

Pirmuoju atveju klaida bus tame langelyje, kuriame įrašytas *nulis*, antruoju – langelyje, kuriame yra nuoroda į nulį. Pirmuoju atveju padidėja nulių skaičius lentelėje, antruoju – ne.

Procedūra *ieškoti0* pirmiausia turi surasti langelį, iš kurio yra nuoroda į nulį. Vienu atveju tai bus *pirmasis* langelis, kitu – *trečiasis*. Radus tokį langelį, pagal nulių skaičių lentelėje reikia nustatyti, ar tai pirmasis ar antrasis atvejis.

```
procedure ieškoti0 (var kl_vieta: integer);
{ taiso klaidą - nuorodą į nulinį langelį }
{ kl_vieta = 0, jei tokia klaida nepadaryta }
  var poz, i, l, sk, turi0, x1, x2: integer;
begin
  poz := 0; { ieškome nuorodos į nulį iš pirmosios }
  for l := 1 to n do
    if perr2[vieta[l]] = 0
       then poz := -l;
  { ir iš antrosios lentelės }
```

```
  for i := 1 to pozicijos do
    if perr2[i] > 0       { jei tai nuoroda, o ne grandinės pabaiga }
       then if perr2[perr2[i]] = 0
                then poz := i;
  if poz <> 0     { jei nuoroda yra }
     then begin
             { rasime kiek nulių turi būti lentelėje }
             turi0 := pozicijos;
             for l := 1 to n do
                turi0 := turi0 - skaič[l];
             if turi0 = kiek(0)
                then { vietoj skaičiaus įrašytas skaičius, rodantis į nulį }
                  if poz < 0
                     then kl_vieta := vieta[-poz]
                     else kl_vieta := poz
                else if poz < 0   { vietoj skaičiaus įrašytas 0 }
                        then kl_vieta := vieta[-poz]
                        else kl_vieta := perr2[poz];
          end;
end;  { ieškoti0 }
```

Einame prie antros klaidos: į tą pačią poziciją egzistuoja dvi nuorodos. Pažymėkime jas *n1* ir *n2*. tai reiškia, kad klaidingai perrašytos lentelės pozicijose *n1* ir *n2* įrašytas tas pats skaičius. Akivaizdu, kad klaida padaryta, arba pozicijoje *n1* arba pozicijoje *n2*.

Jei nors viena šių nuorodų yra iš pirmosios lentelės, matome, kad klaida padaryta antrojoje: juk pirmoji lentelė perrašyta teisingai.

Jei ne, randame kurių leidinių tomai įrašyti langeliuose *n1* ir *n2*. Pažymėkime juos *l1* ir *l2*. Jei nors vienoje šių pozicijų nepadėtas joks leidinys, tai reiškia, kad buvo pamesta vienos grandinės dalis (14 pavyzdys) ir klaida padaryta toje pozicijoje, kurioje yra padėta knyga.

Dar vis neradus klaidos, patikrinkime, ar sutampa *l1* ir *l2*. Jei sutampa, tai reiškia, kad grandinė užsiciklino ir klaida padaryta toje pozicijoje, kur pavyko padėti šio leidinio tomą su didžiausiu numeriu (nebūtinai tai paskutinysis tomas).

Na ir liko paskutinysis atvejis, kai susilieja dvi grandinės. Jei pamestoji grandinės dalis lygi bendrajai abiejų grandinių daliai – klaidos vietos nustatyti neįmanoma. Jei tos dalys nelygios, žinodami kiek kiekvienas šių dviejų leidinių turi tomų, randame klaidos vietą.

```
procedure du_langeliai (lent: lentyna; leid: duomenys;
                        n1, n2: integer;
                        var kl_vieta: integer);
{ taisoma klaida, kai į du langelius (n1 ir n2) įrašytas tas pats skaičius }
  var poz, i, pam, bendra, l1, l2: integer;
begin
  { pirmoje lentelėje klaidos negali būti }
  if n1 < 0  then kl_vieta := n2 { n1 – nuoroda iš pirmos lentelės }
             else if n2 < 0 then kl_vieta := n1;
  if kl_vieta <> 0 then exit;

  { rasime leidinius, kurių tomai šiose pozicijose }
  l1 := lent[n1]; l2 := lent[n2];
  { gal buvo pamestas vienos grandinės vidurys }
  if l1 = -2 then kl_vieta := n2
             else if l2 = -2 then kl_vieta := n1;
  if kl_vieta <> 0 then exit;

  { gal susidarė ciklas vienoje grandinėje }
  if l1 = l2
     then kl_vieta := leid[l1].pask;
  if kl_vieta <> 0 then exit;
  { jei ne, tada susiliejo dvi grandinės }
  { rasime pamestų pozicijų skaičių }
  pam := 0;
  for i := 1 to pozicijos do
    if lent[i] = -2
       then pam := pam + 1;
  { rasime bendrą abiejų grandinių dalį }
  bendra := 0;
  poz := perr2[n1];
  while poz <> -1 do
    begin
      bendra := bendra + 1;
      poz := perr2[poz];
    end;
  if bendra <> pam   { bendra dalis nelygi pamestai }
    then if (perr2[n1] = l1) and
            (leid[l2].kiek+bendra = skaič[l2]) or
            (perr2[n2] = l2) and
            (leid[l1].kiek+bendra <> skaič[l1])
             then kl_vieta := n2
             else kl_vieta := n1
    { else – atstatyti neįmanoma }
end;  { du_langeliai }
```

Liko trečiasis atvejis: skaičius pakeistas *minus vienetu.* Tokiu atveju visų, išskyrus vieno, leidinių tvarkingai išdėliojami visi tomai. O tame leidinyje klaida padaryta pozicijoje, į kurią padėjus tomą užsibaigė grandinė.

```
procedure galas (leid: duomenys; var kl_vieta: integer);
{ taisoma klaida, kai pamestas vienos grandinės galas }
  var l: integer;
begin
  { rasime klaidingai perrašyto leidinio numerį }
  l := 1;
  while (l <= n) and (leid[l].kiek = skaič[l]) do
    l := l + 1;
  kl_vieta := leid[l].pask
end; { galas }
```