Siųstuvai ir imtuvas

Tai buvo vienas sunkiausių Lietuvos informatikos olimpiadų uždavinių. Jo pilnai nepavyko išspręsti nei vienam dalyviui. Sprendimo idėja gana paprasta – perrinkimas. Tiesa, gan sunku jį optimizuoti taip, kad programos vykdymo laikas neviršytų leistino.

Pirmiausia sugeneruojamas teisingas, tačiau nebūtinai optimalus sprendinys. Jis gaunamas naują siųstuvą jungiant tada, kai ankstesnis jau būna pabaigęs darbą. Įsimenama, kada šiuo atveju paskutinysis paskutinio siųstuvo signalas pasiekia imtuvą. Šį momentą pavadinkime *pask.*

Tuomet perrinkinėjama rekursiškai. Vienu rekursijos žingsniu įjungiamas vienas siųstuvas. Siųstuvą bandoma jungti visais laiko momentais pradedant nuliniu ir baigiant momentu *pask*. Atkreipkite dėmesį, kad būtina išbandyti visus jungimo variantus. Negalima sustoti suradus vieną tinkamą. *Šykštus algoritmas* šiam uždaviniui netinka. Kai įjungiami visi siųstuvai, patikrinama, ar rastas optimalesnis sprendinys. Jei taip, jis įsimenamas, grįžtama per žingsnį atgal ir dirbama toliau.

Sprendimas buvo optimizuotas taip:

a) siųstuvai buvo surikiuoti jų pranešimų perdavimo trukmės mažėjimo tvarka. Yra didesnė tikimybė greičiau rasti sprendinį, jei ilgiau dirbančius siųstuvus bandome jungti pirmiau;

b) prieš pradedant perrinkimą kiekvienam siųstuvui buvo apskaičiuota jo *skleistinė*, t. y. suskaičiuotas laikas, kada kiekvienas siųstuvo signalas pasieks imtuvą, jei siųstuvas buvo įjungtas nuliniu laiko momentu. Taip pat sudaroma lentelė, kurioje pažymėta, kokiais momentais imtuvą pasiekia signalai. Šios lentelės pagreitina darbą, tikrinant, ar kažkokiu momentu galima įjungti siųstuvą, t. y. ar tuo momentu įjungus siųstuvą, imtuvas negaus blogų signalų.

c) vartojama nemažai globalių kintamųjų. Masyvų, kaip *parametrų-reikšmių* perdavimas procedūroms ar funkcijoms labai lėtina darbą;

d) Siųstuvą bandoma jungti ne iki momento *pask*. Žiūrima, ar siųstuvas perduos visą savo pranešimą iki šio momento.

e) Jei paskutinįjį siųstuvą pavyko įjungti, darbas iš karto baigiamas.

```
const MAX_N     = 20;  { maksimalus siųstuvų skaičius }
      LAIK      = 200;  { pranešimo perdavimo laikas }
type siųstuv    = array [1..MAX_N] of integer;
     siųstuv_l  = array [1..MAX_N] of boolean;
     lentelė    = array [1..MAX_N, 1..LAIK] of integer;
     imtuvas    = array [1..LAIK*MAX_N] of boolean;

{ žemiau surašyti globalieji kintamieji }
var n: integer;  { siųstuvų skaičius }
    t, l, p: siųstuv;  { siųstuvų charakteristikos }

procedure skleistinė (var skl: lentelė);
{ kada imtuvą pasiekia kiekvieno siųstuvo signalai }
{ jei siųstuvas įjungiamas nuliniu laiko momentu }
  var k, sek, sign: integer;
begin
  for k := 1 to n do
    for sign := 1 to LAIK do
      skl[k, sign] := 0;
  for k := 1 to n do
    begin
      skl[k, 1] := l[k];
      sek := l[k];
      for sign := 2 to p[k] do
        begin
          sek := sek + t[k];
          skl[k, sign] := sek;
        end
    end;
end;  { skleistinė }

procedure rikiuoti (ilg: siųstuv; var tvarka: siųstuv);
{ rikiuoja siųstuvus pranešimų perdavimo laiko mažėjimo tvarka }
{ rikiuojama burbuliuko metodu }
  var i, tarp: integer;
begin
  i := 1;
  while i < n do
    if ilg[tvarka[i]] >= ilg[tvarka[i+1]]
       then i := i + 1
       else begin   { sukeičiame i-tąjį ir (i+1)-mąjį elementą }
              tarp := tvarka[i];
              tvarka[i] := tvarka[i + 1];
              tvarka[i + 1] := tarp;
              if i > 1 then i := i - 1
```

```
                    else i := i + 1;
            end
end;  { rikiuoti }

procedure jungimas (var kada: siųstuv);
{ randa optimalią siųstuvų įjungimo tvarką }
  var skl: lentelė;
      laikas,  { kada pradeda veikti kiekvienas siųstuvas }
      tvarka,  { kokia tvarka bandysime jungti siųstuvus }
      ilg: siųstuv;  { siųstuvų pranešimų perdavimo ilgiai }
      i, k, pirmas, num, sek,
      pask,  { laiko momentas, kada imtuvą pasiekia }
             { paskutinis signalas optimaliame sprendinyje }
      pask_e: integer;  { ir nagrinėjamame sprendinyje }
      imt: imtuvas;    { kuriais momentais imtuvą pasiekia signalai }

  procedure įjungti (k, sek: integer;
                     var imt: imtuvas; var galima: boolean);
  { ar galima siųstuvą k įjungti laiko momentu sek ? }
  { jei galima, tai pakeičiamas masyvas imt }
    var sign, i: integer;
  begin
    sign := 1;
    gerai := true;
    while (skl[k, sign] <> 0) and gerai do
      begin
        if imt[sek + skl[k, sign]]   { jei vieta užimta }
           then begin { atstatome atgal }
                  for i := sign-1 downto 1 do
                    imt[sek + skl[k, i]] := false;
                  gerai := false
                end
             else imt[sek + skl[k, sign]] := true;
        sign := sign + 1
      end;
  end;  { įjungti }

  procedure išjungti (k,  { siųstuvo numeris }
                      sek: integer;  { kada jis buvo paleistas }
                      var imt: imtuvas);
  { išjungia siųstuvą (atstato masyvą imt) kai grįžtama per žingsnį atgal }
    var sign: integer;
  begin
    sign := 1;
    while skl[k, sign] <> 0 do
```

```
    begin
      imt[sek + skl[k, sign]] := false;
      sign := sign + 1
    end;
end;  { išjungti }

procedure dėlioti (nr, pask_e: integer;
                   laikas: siųstuv;
                   var kada: siųstuv;
                   var pask: integer);
{ rekursinė procedūra, dėliojanti siųstuvus }
{ nr – nagrinėjamas siųstuvas }
{ pask_e – kada imtuvą pasiekia įjungtų siųstuvų paskutinis signalas }
  var k, senas, sek: integer;
      galima: boolean;
begin
  if nr > n  { jei visi siųstuvai įjungti }
  then begin        { tai radome geresnį sprendinį }
         if pask_e < pask then
           begin
             kada := laikas;
             pask := pask_e;
           end;
       end
  else begin
         k := tvarka[nr];  { jungsime siųstuvą k }
         sek := 0;  { bandysime jungti nulinę sekundę }
         while sek + ilg[k] < pask do
           begin
             įjungti (k, sek, imt, galima);
             { jei siųstuvą galima įjungti sekundę sek }
             if galima then
                begin
                 senas := pask_e;
                 if pask_e < sek + ilg[k]
                    then pask_e := sek + ilg[k];
                 laikas[k] := sek;
                 dėlioti(nr+1,pask_e,laikas,kada,pask);
                 pask_e := senas;
                 išjungti (k, sek, imt);
                 if nr = n then sek := pask;
                end;
             sek := sek + 1;
           end;
       end;
end;  { dėlioti }
```

```
begin
  skleistinė (skl);
  { randame kiekvieno pranešimo ilgį; bei teisingą, tačiau nebūtinai }
  { optimalų sprendinį }
  for k := 1 to n do
    begin
      tvarka[k] := k;
      ilg[k] := l[k] + (p[k]-1)*t[k];
      if k = 1
         then kada[k] := 0
         else kada[k] := pask + 1;
      pask :=  kada[k] + ilg[k];
   end;
  { siųstuvus surikiuosime pranešimų perdavimo laiko mažėjimo tvarka }
  { tokia tvarka juos vėliau bandysime jungti }
  rikiuoti (ilg, tvarka);
  { parenkame laiką, kada įjungsime ilgiausiai dirbantį siųstuvą }
  pirmas := 0; { ilgiausiai dirbantį įjungiame nuliniu momentu }
  num := tvarka[1];
  while (pirmas + ilg[num]) < pask do { jei dar verta skaičiuoti }
    begin
      for k := 1 to n do   { visi siųstuvai išjungti }
        laikas[k] := 0;
      for i := 1 to pask do { imtuvo nepasiekė nei vienas signalas }
        imt[i] := false;
      { įjungiame ilgiausiai dirbantį imtuvą }
      laikas[num] := pirmas;
      sek := laikas[num] + l[num];
      for i := 1 to p[num] do
        begin
          imt[sek] := true;
          sek := sek + t[num]
        end;
      pask_e := pirmas + ilg[num];
      dėlioti (2, pask_e, laikas, kada, pask);
      pirmas := pirmas + 1; { bandysime jungti sekunde vėliau }
    end;
end;  { jungimas }
```