Siųstuvai ir imtuvas

Šis uždavinys yra atskiras šioje olimpiadoje ankstesniame etape buvusio uždavinio variantas. Iš pirmo žvilgsnio jis atrodo žymiai paprastesnis: siųstuvų jungimo tvarka aiški, tereikia perrinkti visus įmanomus variantus, reikia tiktai supaprastinti ankstesnio uždavinio algoritmą.

Siųstuvus jungsime nuo pirmojo iki *n*-ojo. Sakykime, kad sėkmingai įjungėme pirmuosius *k* siųstuvų. *k*-asis siųstuvas buvo įjungtas momentu $laikas_k$ .Paskutinysis šių siųstuvų signalas imtuvą pasieks laiko momentu *pask_e*. Taigi, jungiant (*k*+1)-ajį siųstuvą jį reikia bandyti jungti visais laiko momentais nuo $laikas_k$+1 iki *pask_e*+1. Deja, bandydami jungti visais šiais momentais, gauname labai daug panašių variantų ir vykdymo laikas labai pailgėja.

Tenka ieškoti kito algoritmo. Siųstuvus jungsime atvirkščiai... – nuo *n*-ojo iki pirmojo. Pradėsime nuo *n*-ojo siųstuvo, kurį įjungsime nuliniu laiko momentu. Paskutinis šio siųstuvo signalas pasieks imtuvą momentu ilg_n, (čia ilg_n – šio siųstuvo pranešimo perdavimo laikas). Taigi optimaliame sprendinyje nuo *n*-tojo siųstuvo įjungimo laiko iki momento, kada paskutinysis signalas pasieks imtuvą gali praeiti ilg_n arba daugiau sekundžių, tačiau negali praeiti mažiau sekundžių. Šią siųstuvo charakteristiką (minimalų laiką nuo siųstuvo įjungimo iki momento kada paskutinysis signalas pasieks imtuvą) pavadinkime opt_i, kur *i* – siųstuvo numeris.

Pirmą kartą jungėme tik *n*-ąjį siųstuvą, antrą kartą jungiame jau (*n*-1)-ąjį ir *n*-ąjį ir t. t. Kiekvieną kartą pradedame jungti nuo vienu numeriu žemesnio siųstuvo ir radę optimalų šių siųstuvų jungimo variantą kartu randame ir to siųstuvo charakteristiką opt_i.

Sakykime, kad mokame optimaliai įjungti siųstuvus nuo *i*-ojo iki *n*-ojo. Ieškosime kaip optimaliai sujungti siųstuvus su numeriais *i*-1, i, i+1, ...n. Pažymėkime *pask* laiko momentą, kada paskutinysis šių siųstuvų signalas pasiekia imtuvą kuriame nors teisingame, tačiau nebūtinai optimaliame sprendinyje. Tokį sprendinį rasti nesunku: galima siųstuvus jungti kas 200 sekundžių.

(*i*-1)-ąjį siųstuvą jungiame nuliniu laiko momentu: $laikas_{i-1} = 0$;

i-tąjį siųstuvą bandome jungti visais laiko momentais nuo $laikas_{i-1}$+1 iki $pask - opt_i$.

Tai yra $laikas_i \in [\ laikas_{i-1}+1;\ pask - opt_i]$. Jei žinome, kad nuo *i*-ojo siųstuvo įjungimo iki darbo pabaigos būtinai turi praeiti opt_i sekundžių, tai nėra prasmės nagrinėti atvejų, kur $laikas_i + opt_i \geq pask$;

Analogiškai (*i*+1)-ąjį siųstuvą bandome jungti laiko momentais $laikas_{i+1} \in [\ laikas_i+1;\ pask - opt_{i+1}]$

```
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 }

    skl: lentelė;  { rodo kada imtuvą pasiekia kiekvieno }
    { siųstuvo signalai, jei siųstuvas įjungiamas nuliniu momentu }

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 įjungti (k, sek: integer;
                   var imt: imtuvas; var galima: boolean);
{ k-ąjį siųstuvą bando įjungti laiko momentu sek }
{ jei galima tai padaryti, tai pakeičiamas masyvas imt }
  var sign, i: integer;
begin
  sign := 1;
  galima := true;
  while (skl[k, sign] <> 0) and galima 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;
                galima := 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 k-ąjį siųstuvą (atstato masyvą imt) }
  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 jungimas (var kada: siųstuv);
{ randa optimalią siųstuvų įjungimo tvarką }
  var laikas, { kada pradeda veikti kiekvienas siųstuvas }
      ilg: siųstuv; { siųstuvų pranešimų perdavimo ilgiai }
      k, i,
      pask, { laiko momentas, kada imtuvą pasiekia }
            { paskutinis signalas optimaliame sprendinyje }
      pask_e: integer; { ir nagrinėjamame sprendinyje }
      imt: imtuvas;    { kuriais momentais imtuvą pasiekia signalai }
```

```
      gerai: boolean;
      opt: siųstuv;

  procedure dėlioti (k, pask_e: integer; laikas: siųstuv;
                     var kada: siųstuv; { optimali jungimo tvarka }
                     var pask: integer);
  { rekursinė procedūra, dėliojanti siųstuvus }
  { k – nagrinėjamas siųstuvas }
  { pask_e – kada imtuvą pasiekia jau įjungtų siųstuvų paskutinis signalas }
    var senas, sek: integer;
        gerai: boolean;
  begin
    if k > n       { jei visi siųstuvai įjungti }
    then begin     { tai radome geresnį sprendinį }
           kada := laikas;
           pask := pask_e;
         end
    else begin
           sek := laikas[k-1] + 1;
           senas := pask_e;
           while sek + opt[k] < pask do { jei yra prasmė jungti }
             begin
               { bandome jungti }
               įjungti (k, sek, imt, gerai);
               if gerai then
                  begin
                    laikas[k] := sek;
                    if pask_e < sek + ilg[k]
                       then pask_e := sek + ilg[k];
                    dėlioti (k+1, pask_e, laikas, kada, pask);
                    pask_e := senas;
                    išjungti (k, sek, imt);
                    if k = n then sek := pask;
                  end;
               sek := sek + 1;
             end;
         end;
  end; { dėlioti }

  var j: integer;
begin
  skleistinė (skl);
  { randame kiekvieno pranešimo ilgį }
  for k := 1 to n do
    ilg[k] := l[k] + (p[k]-1) * t[k];
  opt[n] := ilg[n];
```

```
  for i := n-1 downto 1 do
    { optimaliai sujungsime siųstuvus nuo i-ojo iki n-ojo }
    begin
      { imtuvo nepasiekė nei vienas signalas }
      for j := 1 to LAIK*MAX_N do
         imt[j] := false;
      { i-ąjį siųstuvą įjungiame nuliniu momentu }
      laikas[i] := 0;
      įjungti (i, 0, imt, gerai);
      pask_e := ilg[i];
      { randame teisingą, tačiau nebūtinai optimalų sprendinį }
      for k := i to n do
        begin
          if k = 1
             then kada[k] := 0
             else kada[k] := pask + 1;
          pask := kada[k] + ilg[k];
        end;
      dėlioti (i+1, pask_e, laikas, kada, pask);
      opt[i] :=  pask;
    end;
end;  { jungimas }
```