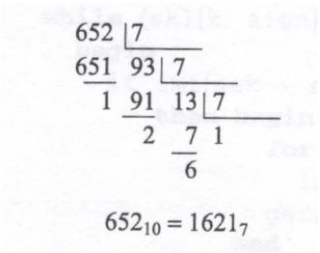
Trys vienodi skaitmenys

Pats paprasčiausias sprendimo būdas perrinkimas. Imame pirmąjį skaičių iš intervalo [n, m]. Jį perrašome visomis skaičiavimo sistemomis nuo 2 iki *n* (t. y. iki paties skaičiaus) ir žiūrime, ar radome dvi sistemas, kuriose skaičius užsirašo trimis vienodais skaitmenimis. Po to imame kitą skaičių iš intervalo ir t. t., kol peržiūrime visus skaičius nuo *n* iki *m*.

Norint parašyti tokį algoritmą, reikia žinoti, kaip dešimtainį skaičių *s* užrašyti *p*-aine sistema. Skaičių s dalijame iš sistemos pagrindo, tol kol liekana tampa mažesnė už pagrindą, o tuomet gautus skaitmenis perrašome atbulai.



Mūsų atveju nėra būtina gauti patį skaičių. Tereikia paimti vieną skaitmenį (paskutinysis skaitmuo bus lygus *s* **mod** *p*) ir patikrinti, ar likusieji skaitmenys jam lygus ir ar skaitmenų skaičius lygus trims.

```
type vienodi = array [1..2] of   { dviems rastiems skaičiams saugoti }
                 record { pagrindas ir skaitmuo }
                   pagr, sk: integer
                 end;

procedure išvesti (s: integer; v: vienodi; var f: text);
{ išveda į bylą rastą skaičių }
begin
  writeln(f, s);
  writeln(f,v[1].pagr,' ',v[1].sk,' ',v[1].sk,' ',v[1].sk);
  writeln(f,v[2].pagr,' ',v[2].sk,' ',v[2].sk,' ',v[2].sk);
end;  { išvesti }

procedure ieškoti_vienodų (m, n: integer; var f: text);
{ randa ieškomus skaičius iš intervalo [n, m] ir juos įrašo į bylą f }
  var s, tiriu, kiek, skaitmuo, p, dvi: integer;
      v: vienodi;
```

```
begin
  rewrite (f);
  for s := m to n do   { tirsime visus skaičius iš duoto intervalo }
    begin
      p := 2;     { pradedama nuo mažiausio sistemos pagrindo }
      dvi := 0;  { kol kas nerasta nė vienos sistemos }
      repeat   { tiriamą skaičių užrašysime įvairiomis sistemomis }
        tiriu := s;
        skaitmuo := tiriu mod p;
        kiek := 1;  { keliais skaitmenimis sistemoje p užsirašo skaičius s }
        tiriu := tiriu div p;
        while (tiriu <> 0) and       { ar visi trys skaitmenys vienodi }
              (kiek < 3) and (tiriu mod p = skaitmuo) do
          begin
            kiek := kiek + 1;
            tiriu := tiriu div p
          end;
        if (kiek=3) and (tiriu=0)  { jei trys skaitmenys vienodi }
           then begin
                  dvi := dvi + 1;  { jau rastas užrašas viena sistema }
                  v[dvi].pagr := p;
                  v[dvi].sk := skaitmuo
                end;
         p := p + 1
         { tiriame tol, kol randame dvi sistemas }
      until (dvi = 2) or (p > s);
      if dvi = 2 then išvesti (s, v, f)
    end;
  close (f);
end;  { ieškoti_vienodų }
```

Galime pasiūlyti ir kitą, matematiškesnį, tačiau žymiai greičiau veikiantį, sprendimą.

Tarkime, kad turime skaičiavimo sistemą su pagrindu p. Nesunku pastebėti, kad

$$111_p = (1 + p + p^2)_{10}. \qquad (1)$$

Bet kuris skaičius, kurį p–ainėje skaičiavimo sistemoje galima užrašyti trimis vienodais skaitmenimis (pažymėkime juos sk) dešimtainėje sistemoje užsirašys taip:

$$sk \times (1 + p + p^2), \text{ kur } 1 \le sk < p. \qquad (2)$$

Turime dešimtainį skaičių *s*. Jis *p*-ainėje sistemoje užsirašo trimis vienodais skaitmenimis, jei

$0 < s$ **div** $(1 + p + p^2) < p$ { *p*-ainės sistemos didžiausias skaitmuo neviršys *p* }

ir

s **mod** $(1 + p + p^2) = 0$ { iš (2) }

Taigi, imame visas sistemas nuo *s* iki 2 ir tiriame, ar tenkinamos šios abi sąlygos. Jei randame dvi tinkamas skaičiavimo sistemas: turime vieną iš ieškomų skaičių.

Patogu susidaryti lentelę, kurioje būtų surašyti skaičiaus 111, užrašyto visomis sistemomis nuo 2 iki 10000 (tokia yra maksimali *n* reikšmė), reikšmės dešimtainėje skaičiavimo sistemoje.

Sistema	2	3	4	5	6	7	…
$111_n = 1 + p + p^2$	7	13	21	31	43	57	…

Aukščiau aprašytas algoritmas yra teisingas, tačiau paėmus didesnį pradinių duomenų intervalą, veiks gan ilgai. Vykdymo laikas labai pagreitės, jei tyrinėsime tik tokias skaičiavimo sistemas, kuriose galioja nelygybė

$$0 < s \textbf{ div } (1 + p + p^2) < p$$

Nėra sunku surasti šiuos rėžius. Akivaizdu, kad jei $p = s$, tai

$$s \textbf{ div } (1 + p + p^2) = 0$$

Lentelėje pritaikę dvejetainę paiešką randame tokią skaičiavimo sistemą *p*, kad

$$s \textbf{ div } (1 + p + p^2) > 0$$

bet s **div** $(1 + (p+1) + (p+1)^2) = 0$. Šias sąlygas galima perrašyti taip:

$$s < (1 + (p+1) + (p+1)^2) \text{ ir } s \geq (1 + p + p^2)$$

Perrinkimą pradedame nuo sistemos su šiuo pagrindu ir mažindami *p* tikriname tol, kol sąlyga s **div** $(1 + p + p^2) \geq p$ tampa teisinga. Sistemų su mažesniu pagrindu nėra prasmės nagrinėti, nes jose skaičius *s* užsirašys keturiais ar daugiau skaitmenų.

```
const MAX   = 20;      { maksimalus ieškomų skaičių kiekis }
      MAX_N = 10000;   { maksimali n reikšmė }
type  rezult = array [1..MAX] of
                 record
                   skaičius,     { skaičius dešimtainėje sistemoje }
                   sis1, sis2,   { abi sistemos }
                   sk1, sk2: longint
```

```
                    { skaitmenys atitinkamose sistemose }
                  end;
var lent: array [2..MAX_N] of longint;
          { lentelė, kurioje bus saugomas skaičiaus 111, užrašyto visose }
          { skaičiavimo sistemose nuo 2 iki MAX_N, atitikmuo dešimtainėje }
          { sistemoje }
          { lent – globalus kintamasis }

procedure vienodi (m, n: longint;
                   var rez: rezult;
                   var kiek: longint);
                   { kiek rasta įdomių skaičių }
  var  rasta,  { keliomis sistemomis trimis vienodais skaitmenimis }
               { užsirašo skaičius }
       pag1, pag2,  { abiejų sistemų pagrindai }
       s            : longint;
  procedure lentelė;
  { sudaro lentelę, kurioje bus saugomas skaičiaus 111, užrašyto visose }
  { skaičiavimo sistemose nuo 2 iki MAX_N, atitikmuo dešimtainėje }
  { sistemoje }
  { kintamasis lent – globalus }
    var i : longint;
  begin
    for i := 2 to MAX_N do
      lent[i] := 1 + i + i * i
  end;  { lentelė }

  function ieškoti (x: longint): longint;
  { dvejetainės paieškos algoritmas }
  { rezultatas – lentelės lent indeksas i, }
  { toks kad lent[i - 1] <= x < lent[i] }
    var i1, i2, i: longint;
  begin
    i1 := 2;
    i2 := x;
    while i1 < i2 do
      begin
        i := (i1 + i2) div 2;
        if x < lent[i]
           then i2 := i
           else i1 := i + 1
       end;
    ieškoti := i1
  end;  { ieškoti }
```

```
begin { vienodi }
  lentelė;        { sudarome lentelę }
  kiek := 0;      { dar nerasta nei vieno skaičiaus }
  if m = 1        { vieneto tikrai negalėsime užrašyti trimis skaitmenimis }
     then m := 2;
  for s := m to n do   { nagrinėsime visus skaičius iš duoto intervalo }
    begin
      rasta := 0;
      pag1 := ieškoti (s); { ieškoma didžiausio galimo pagrindo }
      { ieškosime pirmos sistemos, kurioje skaičius užsirašo }
      { trimis vienodais skaitmenimis }
      while (rasta = 0) and (pag1 > 2) do
        begin
          pag1 := pag1 - 1;
          if (s div lent[pag1] >= pag1)
             { t.y. skaičiavimo sistemose }
             { su mažesniais pagrindais skaičius s užsirašys keturiais }
             { ar daugiau skaitmenų; tų atvejų nereikia tyrinėti }
             then pag1 := 1
             else if (s mod lent[pag1] = 0)
                     { skaičius užsirašo trimis vienodais skaitmenimis }
                     then rasta := rasta + 1
        end;
      { ieškosime antrosios sistemos }
      pag2 := pag1;
      while (rasta = 1) and (pag2 > 2) do
        begin
          pag2 := pag2 - 1;
          if (s div lent[pag2] >= pag2)
             then pag2 := 1
             else if (s mod lent[pag2] = 0)
                     then rasta := rasta + 1
        end;
      if rasta = 2   { jei radome dvi sistemas }
         then begin  { įsimename rezultatą }
                kiek := kiek + 1;
                rez[kiek].skaičius := s;
                rez[kiek].sis1 := pag1;
                rez[kiek].sk1 := s div lent[pag1];
                rez[kiek].sis2 := pag2;
                rez[kiek].sk2 := s div lent[pag2];
              end;
    end;
end;
```

Intervale nuo 1 iki 10000 yra aštuoni tokie skaičiai

273 16 1 1 1 9 3 3 3	$273_{10} = 1\ 1\ 1_{16} = 3\ 3\ 3_{9}$;
546 16 2 2 2 9 6 6 6	$546_{10} = 2\ 2\ 2_{16} = 6\ 6\ 6_{9}$;
931 30 1 1 1 11 7 7 7	$931_{10} = 1\ 1\ 1_{30} = 7\ 7\ 7_{11}$;
3549 22 7 7 7 16 13 13 13	$546_{10} = 2\ 2\ 2_{16} = 6\ 6\ 6_{9}$;
3783 61 1 1 1 35 3 3 3	$3783_{10} = 1\ 1\ 1_{61} = 3\ 3\ 3_{35}$;
4557 67 1 1 1 25 7 7 7	$4557_{10} = 1\ 1\ 1_{67} = 7\ 7\ 7_{25}$;
7566 61 2 2 2 35 6 6 6	$7556_{10} = 2\ 2\ 2_{61} = 6\ 6\ 6_{35}$;
9114 67 2 2 2 25 14 14 14	$9114_{10} = 2\ 2\ 2_{67} = 14\ 14\ 14_{25}$;