Kvadratinė šaknis

Jei reikėtų ištraukti šaknį iš nedidelio natūraliojo skaičiaus, turbūt naudotume dalijimo pusiau metodą. Tarkime, kad turime skaičių *x*. Akivaizdu, kad jo šaknis priklauso intervalui [1..*x*]. Padalijame šį intervalą į dvi dalis, imame intervalo vidurio tašką *ab*, ir keliame kvadratu. Jei gautasis skaičius didesnis už *x*, tai ieškoma šaknis yra pirmoje intervalo pusėje, jei mažesnis – antroje. Taip gavome dvigubai trumpesnį intervalą, kurį vėl dalijame į dvi dalis. Galiausiai intervale beliks du skaičiai. Arba vienas jų bus ieškoma šaknis, arba taps aišku, kad sveikos šaknies iš *x* ištraukti negalima.

Tačiau duotasis skaičius gali turėti iki 80 skaitmenų. Tai gerokai didesnis už *maxlongint* skaičius. Didelių skaičių modeliavimas bei veiksmai su jais aprašyti [2]. Skaičių teks modeliuoti kitomis duomenų struktūromis, pavyzdžiui masyvu. Kiekvienas masyvo elementas yra skirtas vienam skaitmeniui. Šaknį galima traukti tik iš teigiamo skaičiaus. Jei duotasis skaičius galėtų būti neigiamas, tuomet skaičių tektų vaizduoti įrašu, kur vienas įrašo laukas būtų skaičiaus ženklas, kitas – skaičiaus modulis:

```
const SK_ILG = 80;  { maksimalus skaičiaus ilgis }
type didelis = record
                  ženklas: (plius, minus);
                  modulis: array [1..SK_ILG] of 0..9;
                end;
```

Mūsų atveju užtenka paprasto masyvo. Maksimalų skaičiaus ilgį reikėtų padidinti. Mat keliant šaknies artinį kvadratu, galime gauti skaičių, turintį daugiau nei 80 skaitmenų.

```
const SK_ILG = 90;  { maksimalus skaičiaus ilgis }
type didelis = array [1..SK_ILG] of 0..9;
```

Peržiūrėję kvadratinės šaknies traukimo algoritmą (60 uždavinys), pamatysime, kad su dideliais skaičiais reikės atlikti sudėties, atimties, daugybos, sveikųjų skaičių dalybos (**div**) bei lyginimo operacijas.

Pradėkime nuo sudėties operacijos. Skaičiai sudedami stulpeliu.

```
procedure suma (n: integer; sk1, sk2: didelis;
                var sum: didelis);
{ sudeda du teigiamus didelius skaičius, turinčius iki n skaitmenų }
  var perk, tarp, i: integer;
begin
  perk := 0;                  { perkėlimas }
  for i := n downto 1 do      { pradedama nuo paskutiniojo skaitmens }
    begin
      tarp := sk1[i] + sk2[i] + perk;
      sum[i] := tarp mod 10;
      perk := tarp div 10;
    end;
end;  { suma }
```

Atimties operaciją galime pakeisti sudėtimi. Vietoj to, kad tikrintume, ar $b - a > 1$, galime tikrinti, ar $b > a + 1$.

Dauginsime stulpeliu, kaip ir įprasta matematikoje, pirmąjį skaičių dauginsime iš kiekvieno antrojo skaičiaus skaitmeniu ir gautas sandaugas sumuosime.

```
procedure sandauga (n: integer; sk1, sk2 : didelis;
                    var san :didelis);
{ san := sk1 * sk2, kai daugikliai ir rezultatas turi iki n skaitmenų }
  var i, j, k,laik,  tarpinis : integer;
begin
  for i := n downto 1 do
    san[i] := 0;
  for i := n downto 1 do        { kiekvieną skaitmenį }
    for j := n downto 1 do      { dauginame iš antrojo skaičiaus }
      begin
        k := j + i - n;
        tarpinis := sk1[i] * sk2[j];
        while tarpinis <> 0 do  { tarpinę sandaugą pridedame prie }
          begin                 { turimos }
            laik := san[k];
            san[k] := (san[k] + tarpinis) mod 10;
            tarpinis := (laik + tarpinis) div 10;
            k := k - 1;
          end
      end;
end;  { sandauga }
```

Dalinti sveikuosius skaičius reiktų kampu. Tačiau vienintelė algoritme atliekama dalybos operacija yra dalyba iš dviejų. Padalinti iš dviejų galima kur kas paprasčiau. Dalinti pradedama nuo galo. Rezultato skaitmuo bus lygus atitinkamam dalijamo skaičiaus skaitmeniui, padalintam iš dviejų. Jei skaitmuo nesidalija iš dviejų be liekanos, tuomet prie ankstesnio skaitmens pridedama 5. Padalinkime iš dviejų 3594102:

Dalijimo kryptis ←

3	5	9	4	1	0	2
1	2	4	2	0	0	1
	+5	+5	+5		+5	
1	7	9	7	0	5	1

Gavome 1797051

```
procedure div_2 (n: integer; sk: didelis;
                 var dalmuo: didelis);
{ dalmuo := sk div 2; sk turi iki n skaitmenų }
  var i: integer;
begin
  dalmuo[n] := sk[n] div 2;
  for i := n-1 downto 1 do
    begin
      dalmuo[i] := sk[i] div 2;
      if sk[i] mod 2 = 1
         then dalmuo[i+1] := dalmuo[i+1] + 5;
    end;
end; { div_2 }
```

Algoritme vartojamos dvi lyginimo operacijos: *daugiau* ir *daugiau arba lygu*. Jas abi patogu pakeisti operacija *mažiau*.

```
function mažiau (n: integer; sk1, sk2: didelis): boolean;
{ ar sk1 < sk2, kai skaičiai turi iki n skaitmenų }
  var i: integer;
begin
  i := 1;
  while (i < n) and (sk1[i] = sk2[i]) do
    i := i + 1;
  mažiau := sk1[i] < sk2[i];
end; { mažiau }
```

Sumodeliavę veiksmus su dideliais skaičiais, galime rašyti šaknies traukimo procedūrą. Traukiant šaknį iš nedidelio skaičiaus *x*, šaknies ieškoma intervale [1..*x*]. Šiuo atveju reikia stengtis intervalą kaip galima sumažinti. Jei skaičius turi *n* skaitmenų, tai jo šaknis turi (*n* - 1) **div** 2 + 1 skaitmenį, t. y. tiek, kiek didžiausias dešimties laipsnis, mažesnis už šį skaičių. Tai galite nesunkiai parodyti paeksperimentavę su dešimties laipsniais.

Kitas dalykas dėl ko reikia susitarti, tai – didelio skaičiaus laikymas masyve. Tarkime, kad dideliam skaičiui skiriama dešimt skilčių, o mes turime triženklį skaičių. Tuomet galimi du variantai. Arba skaičių dėti į paskutines skiltis, jo priekį užpildyti nuliais ir dirbti su visais 10 skaitmenų:

1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	8	4	1

Arba skaičių saugoti į pirmosiose trijose skiltyse, likusias skiltis palikti neapibrėžtas ir dirbti tik su trimis skiltimis:

1	2	3	4	5	6	7	8	9	10
8	4	1							

Dažniausiai yra sutinkamas pirmasis atvejis. Tačiau įdomumo dėlei savo algoritme pasirinkome antrąjį. Jei skaičius turi *n* skaitmenų, tai keliant šaknies artinį kvadratu, gali prireikti dirbti su skaičiais, turinčiais *n*+1 skaitmenį. Pavyzdžiui, traukiame šaknį iš 625. Žinome, kad šaknis yra dviženklis skaičius. Imame intervalo [10..99] vidurio tašką 54, keliame kvadratu ir gauname 2916 – keturženklį skaičių. Todėl skaitydami pradinius duomenis prieš skaičių įrašome nulį. Pavyzdžiui, skaičių 2916 įrašytume taip:

1	2	3	4	5	6	7	8	9	10
0	2	9	1	6					

ir dirbtume su pirmomis *n*+1 skiltimis.

```
procedure šaknis (n: integer;  { x skaitmenų skaičius }
                  x: didelis;  { didelis skaičius įrašytas į n+1 skiltį }
                  var galima: boolean;  { ar galima ištraukti šaknį }
                  var rez: didelis   { šaknis });
  var sk,     { skaitmenų skaičius ieškomoje šaknyje }
      i: integer;
      a, b, ab,  { intervalo rėžiai ir vidurio taškas }
      a_plius_1, a_plius_b, vienas,
      xx: didelis;   { vidurio taškas, pakeltas kvadratu }
begin
  n := n + 1;  { bus dirbama su n+1 skiltimi }
  for i := 1 to n-1 do
    vienas[i] := 0;
  vienas[n] := 1;                    { a := 1 }
  sk := (n - 2) div 2 + 1;           { randame šaknies skaitmenų skaičių }
  for i := n downto 1 do             { sudarome intervalą }
    begin                            { a := 1000...000; }
      a[i] := 0;                     { b := 9999...999; }
      if i >= n - sk + 1
         then b[i] := 9
         else b[i] := 0;
    end;
  a[n - sk + 1] := 1;
  sandauga (n, a, a, xx);
  if mažiau (n, xx, x) or mažiau (n, x, xx)  { jei a nėra šaknis }
  then begin
         suma (n, a, vienas, a_plius_1);  { sudarome a_plius_1 }
         while mažiau (n, a_plius_1, b) do  { while b - a > 1 do }
           begin
             suma (n, a, b, a_plius_b);      { a_plius_b := a + b }
             div_2 (n, a_plius_b, ab);     { ab := a_plius_b div 2 }
             sandauga (n, ab, ab, xx);          { xx := ab * ab }
             if mažiau (n, xx, x)               { if xx < x }
                then a := ab                     { then a := xx }
```

```
            else b := ab;                          { else b := xx }
          suma (n, a, vienas, a_plius_1)  { a_plius_1 := a+1}
        end
      end
  else b := a;   { jei a yra šaknis }
  sandauga (n, b, b, xx);                    { xx := ab * ab }
  galima := not mažiau(n,xx,x) and not mažiau(n,x,xx);
  if galima
     then rez := b;
end;  { šaknis }
```

Šiek tiek efektyvesnis šaknies traukimo algoritmas būtų vadinamasis šaknies traukimas kampu. Tačiau šį algoritmą žino stipriau besimokantys matematiką. Be to jame vis tiek reikia modeliuoti didelius skaičius. Kadangi galima apsieiti ir be labai matematiško sprendimo, jo nenagrinėjame.