НАЛАЖЕЊЕ ПРОСТИХ БРОЈЕВА КОРИШЋЕЊЕМ ЕРАТОСТЕНОВОГ СИТА

Бура Паунић, Институт за математику, Нови Сад

Разликовање простих од сложених бројева је важан проблем у Теорији бројева. Још у старом веку је пронађен врло једноставан поступак за налажење простих бројева, Ератостеново сито, али се оно није много користило до проналаска рачунара, јер није погодно за ручно рачунање.

Ератостеново сито је поступак за налажење свих простих бројева у неком интервалу, који је измислио александријски математичар Ератостен (2. век п.н.е.). Основна идеја Ератостеновог сита следећа:

- Исписати све бројеве почевши од 2, па до неке границе n.
- Затим прецртати све умношке 2.
- Уочити први следећи непрецртан број и прецртати све његове умношке почињући од његовог квадрата.
- Наставити овај поступак док је квадрат првог непрецртаног броја мањи или једнак n.

У следећем примеру су уочени бројеви подвучени, граница је $n = 100$, а поступак прецртавања треба урадити за 2, 3, 5 и 7.

	2,	3,	~~4~~,	5,	~~6~~,	7,	~~8~~,	~~9~~,	~~10~~,
11,	~~12~~,	13,	~~14~~,	~~15~~,	~~16~~,	17,	~~18~~,	19,	~~20~~,
~~21~~,	~~22~~,	23,	~~24~~,	~~25~~,	~~26~~,	~~27~~,	~~28~~,	29,	~~30~~,
31,	~~32~~,	~~33~~,	~~34~~,	~~35~~,	~~36~~,	37,	~~38~~,	~~39~~,	~~40~~,
41,	~~42~~,	43,	~~44~~,	~~45~~,	~~46~~,	47,	~~48~~,	~~49~~,	50,
51,	~~52~~,	53,	~~54~~,	~~55~~,	~~56~~,	~~57~~,	~~58~~,	59,	~~60~~,
61,	~~62~~,	~~63~~,	~~64~~,	~~65~~,	~~66~~,	67,	~~68~~,	~~69~~,	~~70~~,
71,	~~72~~,	73,	~~74~~,	~~75~~,	~~76~~,	~~77~~,	~~78~~,	79,	~~80~~,
~~81~~,	~~82~~,	83,	~~84~~,	~~85~~,	~~86~~,	~~87~~,	~~88~~,	89,	~~90~~,
91,	~~92~~,	~~93~~,	~~94~~,	~~95~~,	~~96~~,	97,	~~98~~,	~~99~~,	~~100~~,

Још је Ератостен уочио да се овај алгоритам може поједноставити тако да се уместо низа природних бројева користи низ непарних природних бројева већих од 1 и на њега примени исти алгоритам. Наиме, сви парни бројеви осим двојке су очигледно сложени, сваки други умножак простог броја p је паран број који

се не налази у низу непарних бројева тако да се позиција следећег броја за прецртавање, у низу непарних бројева, опет повећава за p, а квадрат непарног броја је непаран број па се p^2 сигурно налази у ситу од непарних бројева.

Друга примедба је да елементи низа који се прецртавају не морају бити природни бројеви, јер се прости бројеви добијају као **позиције** непрецртаних елемената у ситу, а не њихове вредности. За реализацију алгоритма се може користити најједноставнији низ, тј. низ елемената који заузимају најмање меморије, а то је низ битова или низ бајтова. У Паскалу је погодно користити логички низ тако да вредност елемента у низу буде `true` када је индекс прост број. Дакле, логички низ треба иницијализовати са `true`, а при каснијим проласцима ставити за сложене вредности индекса да је вредност елемента низа `false`. У следећој процедури се и штампају нађени прости бројеви. Ако нас интересује само њихов број `pi` тада у делу за бројање треба избацити штампање. Уз ова побољшања добија се:

```
procedure eratosten(granica : integer;
                    var pi : integer;
                    var ok : boolean);
const
  maxsito = 5000;
type
  logickiniz = array[1 .. maxsito] of boolean;
var
  sito : logickiniz;
  i, prost, slozen, velicina : integer;
begin
  if (granica > 2 * maxsito) or (granica < 2) then
    ok := false
  else
    begin
      ok := true;
      velicina := granica div 2;
      for i := 1 to maxsito do
        sito[i] := true;
      for i := 1 to trunc(sqrt(granica)) div 2 do
        if sito[i] then
          begin
            prost := 2 * i + 1;
            slozen := sqr(prost) div 2;
            while slozen <= velicina do
              begin
                sito[slozen] := false;
                slozen := slozen + prost
              end
          end;
      pi := 1; (* broj 2 se ovim postupkom ne dobija *)
      write(2:8); (* pa ga treba posebno brojati     *)
      if not odd(granica) then
        velicina := velicina - 1;
      for i := 1 to velicina do
        if sito[i] then
          begin
```

```
            pi := pi + 1;
            prost := 2 * i + 1;
            write(prost:8)
          end
    end
end;   (*  eratosten   *)
```

Међутим, увек постоји ограничење на величину низа, тако да се овом процедуром не може наћи велик број простих бројева, јер би требало користити врло дугачко сито, веће од расположиве меморије рачунара. Могуће је модификовати овај алгоритам тако да се дугачко сито подели на делове, а да се ти делови реализују у једном истом логичком низу, јер за израчунавање позиције следећег умношка простог броја у ситу, сито није потребно у целини него само позиција претходно прецртаног елемента и корака потребног за прецртавање следећег елемента. Дакле, сито је могуће реализовати тако да се један исти логички низ користи више пута као део једног "великог" сита. Да се постигне поновно коришћење истог низа, треба у првом пролазу запамтити сваки прост број чијим умношцима треба да се прецртава, позицију првог појављивања тог простог броја у следећем "наставку" сита и њихов укупан број. У осталим пролазима треба направити "ново", неиспрецртавано, парче сита у истом логичком низу, прерачунати позиције у "новом" ситу од којих треба наставити "прецртавање" упамћеним простим бројевима с обзиром да нови део сита почиње од 1 и наставити "прецртавати". Због доброг уклапања делова сита погодно је да индекс низа `sito` почиње од 0 и да се прости бројеви броје до `maxsito - 1`.

У следећој процедури се користи поступак просејавања у коме су већ елиминисани парни бројеви, а одабрана је и релативно мала величина сита од свега `maxsito = 5000`, тј. у ситу могу да се одреде и запамте сви прости бројеви мањи од `maxgranica = 10000; (* 2 * maxsito *)`. Њих има свега 1229 и њихов број се памти у константи `brprostih`. Ератостеново сито даје све просте бројеве $< p_k^2$, где је p_k последњи прост број за који је вршено прецртавање. Дакле, овом процедуром могу се одредити сви прости бројеви мањи од сто милиона. При том би се сито "настављало" 10000 пута. Наравно ако се њом израчунава број простих бројеви до границе која је мања од 10^8, поступак ће се понављати мањи број пута. Због потенцијалне величине сита треба користити целе бројеве бар двоструке прецизности. Прости бројеви који се прецртавају, чувају се у низу `prbaza` (проста база), њихов број у `grbaze`, а позиција првог броја који треба да се прецрта у следећем комаду сита због дељивости бројем `prbaza[i]` у низу `prvi[i]`. У `ponavljanje` је број колико пута треба настављати сито, а у `pomeranje` је вредност коју треба одузети од `prvi[i]`, јер индекси поново полазе од 1. У овој процедури се прости бројеви исписују, али је и избројан укупан број простих бројева и он се даје у променљивој `pi`. Могуће је штампу простих бројева преусмерити у неку датотеку и сл.

```
procedure eratosten1(granica : integer;
                     var pi : integer;
                     var ok : boolean);
const
  maxsito = 5000;
  maxgranica = 10000; (*    2 * maxsito    *)
  brprostih = 1229; (* broj prostih manjih od maxgranica *)
```

```
type
  logickiniz = array[0 .. maxsito] of boolean;
  nizprostih = array[1 .. brprostih] of integer;
var
  sito : logickiniz;
  prbaza, prvi : nizprostih;
  i, j, n, kraj, grbaze, maxbaza,
  ponavljanje, prost, pomeranje, slozen : integer;
begin
  maxbaza := trunc(sqrt(granica));
  if (granica < maxgranica) or (maxbaza > maxgranica) then
    ok := false
  else
    begin
      ok := true;
      pi := 1;    (* odbrojati i napisati 2 *)
      write(2:8);
      for i := 0 to maxsito do (* inicijalizovati *)
        sito[i] := true;
      grbaze := 0;
      for i := 1 to maxsito - 1 do  (* prvi prolaz *)
        if sito[i] then
          begin
            prost := 2*i + 1;
            pi := succ(pi);
            write(prost:8);
            if prost <= maxbaza then
              begin
                slozen := sqr(prost) div 2;
                while slozen < maxsito do
                  begin
                    sito[slozen] := false;
                    slozen := slozen + prost
                  end;
                grbaze := grbaze + 1;
                prbaza[grbaze] := prost;
                prvi[grbaze] := slozen
              end
          end;
      pomeranje := 1;
      granica := granica - maxgranica;
      ponavljanje := granica div maxgranica;
      for n:= 1 to ponavljanje do   (* ostali prolazi  *)
        begin                       (* kroz celo sito. *)
          for j := 0 to maxsito do
            sito[j] := true; (* inicijalizovati novo   *)
          for j := 1 to grbaze do (* parce i nastaviti *)
            begin     (*  precrtavanje prostim iz baze *)
              prost := prbaza[j];
              slozen := prvi[j] - maxsito;
              while slozen < maxsito do
```

```
                begin
                  sito[slozen] := false;
                  slozen := slozen + prost
                end;
              prvi[j] := slozen (* upamtiti sledeceg  *)
            end;          (* za nastavak precrtavanja *)
          pomeranje := pomeranje + maxgranica;
          for i := 0 to maxsito - 1 do
            if sito[i] then  (* izbrojati neprecrtane *)
              begin
                pi := succ(pi);
                write(i+i+pomeranje:8)
              end;
          granica := granica - maxgranica
        end;
      if granica > 0 then  (* ako ima ostatak na kraju *)
        begin
          for j := 0 to maxsito do
            sito[j] := true;
          for j := 1 to grbaze do
            begin
              prost := prbaza[j];
              slozen := prvi[j] - maxsito;
              while slozen < maxsito do
                begin
                  sito[slozen] := false;
                  slozen := slozen + prost
                end;
              prvi[j] := slozen
            end;
          pomeranje := pomeranje + maxgranica;
          kraj := granica div 2;
          if not odd(granica) then
            kraj := pred(kraj);
          for i := 0 to kraj do
            if sito[i] then
              begin
                pi := succ(pi);
                write(i+i+pomeranje:8)
              end
        end;
      write('Ima ',pi,' prostih brojeva manjih od ');
      writeln(pomeranje + granica - 1)
    end
end;   (*   eratosten1   *)
```

Статијата прв пат е објавена во списанието Тангента на ДМ на Србија во 1997/98 година