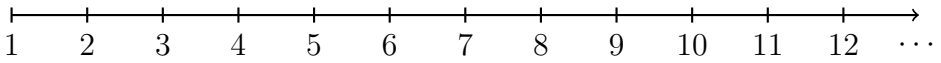
Eratostenovo rešeto

4. marec, 2007

1 Uvod

Doma (v točki nič) nam postane dolgčas in podamo se na sprehod po številski premici.



Slika 1: Naravna števila, predstavljena na številski premici.

Hitro opazimo, da se večkratniki števil pojavljajo enakomerno. Vsako sodo število se pojavi na vsakem drugem koraku (pri tem seveda privzamemo, da je dolžina našega koraka enaka ravno enoti), ali če želite, ob desni nogi (če smo začeli naš sprehod z levo). Vsak večkratnik števila tri se pojavi enkrat na levi nogi, enkrat na desni, vendar pa pred tem na obeh korakih manjka, ... Kaj pa če opazujemo cela števila, ki niso večkratniki nobenega drugega števila, kakor le samega sebe oziroma enice, ki tako ali tako pokrije vsa števila? Tem številom pravimo *praštevila* oziroma nerazcepna števila, saj se jih ne da napisati v obliki produkta dveh od ena različnih števil. Ali se tudi praštevila pojavljajo po kakšnem pravilu?

V tem sestavku bomo spoznali, kako bi se s tem lahko ukvarjal F. Gauss, če bi imel na voljo računalnik. Prav on je bil namreč tisti, ki je postavil domnevo glede urejenosti praštevil.

2 Rešeto

Gotovo ste v šoli že slišali za Eratostenovo rešeto. To je eden od najstarejših algoritmov, s pomočjo katerega lahko najdemo praštevila. Sestavimo tabelo števil od 1 do npr. 400. Za začetek prečrtamo vsa soda števila, z izjemo prvega, nato vse večkratnike števila 3, zopet z izjemo prvega. Ker smo število 4 že prečrtali, nadaljujemo z večkratniki števila 5. Po tem postopku pridejo na vrsto še večkratniki števil 7, 11, 13, 17 in 19.

> Gotovo veste zakaj nam v tabeli ostanejo samo še praštevila skupaj z enico? (Naj vam zaupava, da najmanjši izmed faktorjev sestavljenega števila ni nikoli večji od korena tega števila.)

Ni kaj, to je kar učinkovit algoritem. Morda je bolj zabavno opravljati izločanje sestavljenih števil v obratnem vrstnem redu (od večkratnikov števil 19, 17, ..., pa vse do večkratnikov števila 2)[1]. V pomoč ti je lahko tabela na naslednji strani, bolj interaktivno pa je skočiti na spletno stran `http://www.faust.fr.bw.schule.de/mhb/eratosiv.htm` in opazovati kakšne vzorce naredijo izginjajoča sestavljena števila. Zabava gor ali dol, a ne zanimajo nas praštevila samo do 400. Če hočemo še naprej, si bomo seveda pomagali z računalnikom. Zapišimo kodo, ki opravi to delo namesto nas:

[1] O tem bi lahko matematiki napisali še en cel članek.

1	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
101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120
121	122	123	124	125	126	127	128	129	130	131	132	133	134	135	136	137	138	139	140
141	142	143	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	160
161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	176	177	178	179	180
181	182	183	184	185	186	187	188	189	190	191	192	193	194	195	196	197	198	199	200
201	202	203	204	205	206	207	208	209	210	211	212	213	214	215	216	217	218	219	220
221	222	223	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239	240
241	242	243	244	245	246	247	248	249	250	251	252	253	254	255	256	257	258	259	260
261	262	263	264	265	266	267	268	269	270	271	272	273	274	275	276	277	278	279	280
281	282	283	284	285	286	287	288	289	290	291	292	293	294	295	296	297	298	299	300
301	302	303	304	305	306	307	308	309	310	311	312	313	314	315	316	317	318	319	320
321	322	323	324	325	326	327	328	329	330	331	332	333	334	335	336	337	338	339	340
341	342	343	344	345	346	347	348	349	350	351	352	353	354	355	356	357	358	359	360
361	362	363	364	365	366	367	368	369	370	371	372	373	374	375	376	377	378	379	380
381	382	383	384	385	386	387	388	389	390	391	392	393	394	395	396	397	398	399	400

Slika 2: Tabela prvih 400 naravnih števil.

Psevdo koda

```
1  n ← 1000;
2  p ← array[1..n];
3  p[1] = 1;
4  for i ← 2 to n do
5     if p[i] = 0 then
6        for j ← 2 to n / i do
7           p[j * i] ← 1;
```

Poglejmo si kako koda deluje. V prvi vrstici povemo, da bomo iskali vsa praštevila do 1000. Nato si v drugi vrstici pripravimo tabelo `p` prvih 1000 naravnih števil. Vrednosti v tabeli so postavljene na 0, kar naj pomeni, da so še neprečrtane. Črtanje števila i v tabeli bomo označili tako, da bomo vrednost `p[i]` postavili na 1. V tretji vrstici tako prečrtamo število 1, ki ni ne praštevilo ne sestavljeno število. Nato pa sledimo postopku, ki smo ga opisali zgoraj. Tako se v četrti vrstici s for zanko sprehodimo po vseh naravnih številih od 2 do `n`, saj v naprej ne vemo katera števila bodo že prečrtana in katera ne. To ugotovimo v peti vrstici, ko z if stavkom preverimo, če je trenutno število še neprečrtano (`p[i] = 0`). Samo v tem primeru namreč črtamo njegove večkratnike. To naredi for zanka v šesti vrstici, ki nas popelje od drugega pa do $\lfloor \frac{n}{i} \rfloor$-tega večkratnika. Samo črtanje večkratnikov opravimo v sedmi vrstici. Ko se postopek konča, imamo v tabeli `p` spravljene informacije o tem katera števila so sestavljena (imajo vrednost 1) in katera ne (vrednost 0). Zgornjo kodo lahko seveda hitro zapišemo v poljubnem programskem jeziku in popravimo tako, da uporabnik sam vpiše zgornjo mejo `n`.

Pozoren bralec bo hitro opazil, da se da zgornjo kodo še bistveno izboljšati. V zadnji forzanki smo črtali kar vse večkratnike, ne glede na to ali so bili že črtani ali ne. Ker je branje precej hitrejše od pisanja, lahko z dodatnim if stavkom, ki preveri ali je večkratnik že prečrtan ali ga še moramo, naš program precej pohitrimo. Nadalje smo že omenili, da je v prvi for zanki dovolj gledati le do korena števila `n`. S čimer v teoriji prihranimo veliko korakov, v praksi pa ti dodatni klici if stavka v peti vrstici ne stanejo skoraj nič, tako da je prihranek neopazen. Dobro pa ga je imeti v mislih pri veliko večjih številih.

Vsak izkušen programer bi nam tudi povedal, da je deljenje števil v računalniku dosti počasnejše od množenja, ki je tudi počasnejše od seštevanja. Zato bi lahko šesto in sedmo vrstico zgornjega programa spremenili tako, da bi začeli začeli z dvakratnikom števila i in na vsakem koraku prištevali i namesto 1. S tem se znebimo tako deljenja kot množenja (če odštejemo enkratno množenje z 2 na začetku).

Zadnja izboljšava, ki jo bomo omenili pa je, da nam v for zanki iz šeste vrstice ni potrebno začeti pri dvakratniku števila i, ampak lahko začnemo šele pri i-kratniku števila i, torej z i^2. Vsi večkratniki števila i, ki so manjši od i^2 so namreč ravno i-ti večkratniki števil $2, \ldots, i-1$.

Poglejmo si kako bi izgledal naš program z vsemi zboljšavami v programskem jeziku C. Iskal bo vsa praštevila do 100.000.000.

C

```c
#include <stdio.h>

#define MAX 100000000
#define SQRT 10000

char p[MAX + 1];

int main(void) {
   int i, j;

   for (i = 2; i <= SQRT; i++)
      if (p[i] == 0)
         for (j = i * i; j <= MAX; j += i)
            if (p[j] == 0)
               p[j] = 1;

   return 0;
}
```

Naš program nam da pri različnih vrednostih spremenljivke `MAX` naslednje rezultate:

MAX	# praštevil	n / ln(n)
100	25	1.151292546
1.000	168	1.160502887
10.000	1.229	1.131950832
100.000	9.592	1.104319811
1.000.000	78.498	1.084489948
10.000.000	664.579	1.071174789
100.000.000	5.761.455	1.061299232
1.000.000.000	??????????	

Si predstavljate, da je Gauss prišel do tega na roke in postavil naslednjo domnevo:

$$\#\text{ praštevil} \approx \frac{n}{\ln n}.$$

Pri tem imejte v mislih, da je bilo takrat že računanje naravnega logaritma netrivialno opravilo, saj še ni bilo kalkulatorjev. Na voljo so bile le Vegove tablice.

Iz zgornje ocene lahko dobimo tudi oceno za velikost n-tega praštevila p_n:

$$p_n \approx n \ln n.$$

Aleksandar Jurišić in Matjaž Urlep