resitev

January 28, 2024

1 Praštevila na spirali (Ulamova spirala)

Tudi resnični avtor te naloge je taisti Ulam, s čigar števili smo se zabavali pred kratkim, le da smo jim rekli "Zanesljiva števila", da ne bi bilo prehudih skušnjav z Googlanjem.

Gre za Ulamovo spiralo.

Nalogo lahko seveda rešujemo tako, da se sprehodimo po spirali in za vsako število, ki je na diagonali, preverimo, ali je praštevilo. Lažje pa gre, če odkrijemo formulo za števila na spirali.

Poglejmo nekaj rešitev.

1.0.1 Formula za diagonalo

Ker gre za kvadratno spiralo, bomo najbrž nekje na njej opazili kvadrate števil. V resnici imamo 4, 16, 36 (in to se potem nadaljuje, poskusite!) takoj desno od diagonale, ki gre levo gor. Števila na tej diagonali so torej enaka 0^2+1, 2^2+1, 4^2+1, 6^2+1 … $(2i)^2+1$. Števila na diagonali, ki gre desno gor so za i manjša, števila na diagonali levo dol pa za i večja. Števila na diagonali desno dol nas ne zanimajo, saj med njimi očitno ni praštevil.

[1]:
```
# Program je nekoliko šlampast z mejami: če ne bi želeli ravno 10000 števil,
# bi morali popaziti, da ne gremo čez gornjo mejo!

prastevil = 0
for i in range(2, 101, 2):
    n = i ** 2 + 1
    for f in (-1, 0, 1):
        kandidat = n + i * f
        d = 2
        while d < kandidat:
            if kandidat % d == 0:
                break
            d += 1
        else:
            prastevil += 1
    i += 2
print(prastevil)
```

```
50
```

Tole je zaradi vse teh zank precej nepregledno. Čeprav se uradno še nismo učili pisanja funkcij, napišimo ločeno funkcijo za ugotavljanje praštevilskosti.

[2]:
```
def je_prastevilo(n):
    for i in range(2, n):
        if n % i == 0:
            return False
    return True

prastevil = 0
for i in range(2, 101, 2):
    n = i ** 2 + 1
    for f in (-1, 0, 1):
        kandidat = n + i * f
        if je_prastevilo(kandidat):
            prastevil += 1
print(prastevil)
```

```
50
```

1.0.2 Koraki

Namesto formule lahko opazujemo razlike med števili na diagonalah. Števila na diagonalah so 3, 5, 7, 9; 13, 17, 21, 25; 31, 37, 43, 49; 57, 65, 73, 81; … Razdelili smo jih v skupinice po štiri: razlike med prvo četverko so 2, med drugo 4, med tretjo 6 in tako naprej. Potrebujemo torej program, ki začne pri 1 in potem štirikrat prišteje 2, štirikrat 4, štirikrat 6 …

[3]:
```
prastevil = 0
n = 1
korak = 2
while n < 10000:
    for f in range(4):
        n += korak
        if je_prastevilo(n):
            prastevil += 1
    korak += 2
print(prastevil)
```

```
50
```

1.0.3 Numpy

Ker smo si na dodatnih predavanjih ogledali numpy, je tu še rešitev z njim. Nekateri so ga tudi dejansko uporabljali, čeprav nam tu pravzaprav ne pomaga bistveno: najbolj časovno zahteven del naloge je ugotavljanje praštevilskosti in to je zoprno z numpyjem ali brez.

Najprej funkcija `je_prastevilo`. Sestavimo tabelo ostankov po deljenju `x` z vsemi števili od 2 do $\lceil\sqrt{x}\rceil$ in se vprašamo, če so vsi različni od 0. To ne deluje pri $x = 2$, vendar me to ne vznemirja, saj 2 ni na diagonali.

[4]:
```
import numpy as np

def je_prastevilo(x):  # ne deluje za 2, vendar ga ne potrebujemo
    return np.all(x % np.arange(2, int(np.ceil(1 + np.sqrt(x)))))
```

Zdaj pa sestavimo tabelo vseh števil na diagonali. Formula bo takšna, kot v rešitvi v čistem Pythonu. V `d` bomo naračunali diagonalo desno gor, `d = i ** 2 + 1`. Ostali dve diagonali sta `d - i` in `d + i`. Vse tri zbašemo v isto tabelo.

[5]:
```
i = np.arange(2, 101, 2)
d = i ** 2 + 1
a = np.vstack((d - i, d, d + i))
a
```

[5]:
```
array([[    3,    13,    31,    57,    91,   133,   183,   241,   307,
         381,   463,   553,   651,   757,   871,   993,  1123,  1261,
        1407,  1561,  1723,  1893,  2071,  2257,  2451,  2653,  2863,
        3081,  3307,  3541,  3783,  4033,  4291,  4557,  4831,  5113,
        5403,  5701,  6007,  6321,  6643,  6973,  7311,  7657,  8011,
        8373,  8743,  9121,  9507,  9901],
       [    5,    17,    37,    65,   101,   145,   197,   257,   325,
         401,   485,   577,   677,   785,   901,  1025,  1157,  1297,
        1445,  1601,  1765,  1937,  2117,  2305,  2501,  2705,  2917,
        3137,  3365,  3601,  3845,  4097,  4357,  4625,  4901,  5185,
        5477,  5777,  6085,  6401,  6725,  7057,  7397,  7745,  8101,
        8465,  8837,  9217,  9605, 10001],
       [    7,    21,    43,    73,   111,   157,   211,   273,   343,
         421,   507,   601,   703,   813,   931,  1057,  1191,  1333,
        1483,  1641,  1807,  1981,  2163,  2353,  2551,  2757,  2971,
        3193,  3423,  3661,  3907,  4161,  4423,  4693,  4971,  5257,
        5551,  5853,  6163,  6481,  6807,  7141,  7483,  7833,  8191,
        8557,  8931,  9313,  9703, 10101]])
```

Če hočemo lepo urejeno tabelo, jo transponirajmo in sploščimo.

[6]:
```
a.T[:10]  # izpišimo samo prvih deset vrstic, da ne gledamo dolgih izpisov
```

[6]:
```
array([[  3,   5,   7],
       [ 13,  17,  21],
       [ 31,  37,  43],
       [ 57,  65,  73],
       [ 91, 101, 111],
       [133, 145, 157],
       [183, 197, 211],
       [241, 257, 273],
       [307, 325, 343],
       [381, 401, 421]])
```

[7]:
```
a.T.flatten()[:40]  # izpišimo samo začetek
```

[7]:
```
array([  3,   5,   7,  13,  17,  21,  31,  37,  43,  57,  65,  73,  91,
       101, 111, 133, 145, 157, 183, 197, 211, 241, 257, 273, 307, 325,
       343, 381, 401, 421, 463, 485, 507, 553, 577, 601, 651, 677, 703,
       757])
```

Vse skupaj je torej:

[8]:
```
i = np.arange(2, 101, 2)
d = i ** 2 + 1
a = np.vstack((d - i, d, d + i)).T.flatten()
```

Program pa bi enako dobro deloval tudi, če števila ne bi bila urejena: namesto `vstack` uporabimo `hstack`, ki tabele spne skupaj v eno samo vrstico, pa se izognemu transponiranju in sploščevanju.

[9]:
```
i = np.arange(2, 101, 2)
d = i ** 2 + 1
a = np.hstack((d - i, d, d + i))
```

Kakorkoli se že lotimo, na koncu le še seštejemo, kolikokrat `je_prastevilo` za ta števila vrne `True`.

Celoten program je

[10]:
```
import numpy as np

def je_prastevilo(x):  # ne deluje za 2, vendar ga ne potrebujemo
    return np.all(x % np.arange(2, int(np.ceil(1 + np.sqrt(x)))))

i = np.arange(2, 101, 2)
d = i ** 2 + 1
a = np.hstack((d - i, d, d + i))

print(sum(je_prastevilo(x) for x in a))
```

```
50
```

Zlaganje v isto tabelo je sicer nepotrebno. Naredimo lahko tudi tako:

[11]:
```
i = np.arange(2, 101, 2)
a = i ** 2 + 1
print(sum(je_prastevilo(x) for d in (a - i, a, a + i) for x in d))
```

```
50
```