Vilniaus Universitetas

**Matematikos ir Informatikos fakultetas**

**3 kursas 1 grupė - Informatika**

*Marius Pozniakovas*

Užduotis 23:

*Pirminiai skaičiai  
(Eratosteno rėčio ir Miller-Rabin algoritmai)*

***Užduotis:***Duotas natūralusis skaičius n. Nustatyti, ar n yra pirminis skaičius.

Natūralusis skaičius – skaičius, kuris **tik** dalinasi *iš savęs* ir *iš vieneto*. Mokyklos matematikos kurse pasakyti ar skaičius yra pirminis atrodo nesunku, nes skaičiai, kurie yra tikrinami dažniausiai būna mažesni už 100. Toliau gilinantis, skaičiai didėja ir natūralu, kad dalinti iš 2,3,5 ir t.t. užtrunka tiesiog per daug laiko ir yra neoptimalu. Todėl matematikai atranda įvairių algoritmų, kurie gali pagelbėti šitai problemai. Su šiuo laboratoriniu darbu bus analizuojami **Eratosteno rėčio** ir **Miller-Rabin** algoritmai.

***Algoritmų kintamieji ir apipavidalinimas programoje:***

Algoritmo įgyvendinimui naudojamas python3.8.2. Failai ir aplankalai esantys projekto direktorijoje:

* *main.py* (naudojamas pagrindinei programos tėkmei)
* ***py\_files***aplankas:
  + *prime\_test.py* (naudojamas saugoti pagrindinius pirminių skaičių testavimo algoritmus)
  + *file\_operation.py* (naudojamas saugoti funkcijas skirtas darbui su failais)
  + *generate\_graphs.py* (naudojamas generuoti algoritmo sudėtingumo grafikus)
  + *\_\_init\_\_.py* (tuščias failas, skirtas sujungti visus tris viršuje paminėtus failus)
* *task.png* (paveiksliukas su užduotimi)
* ***files*** *aplankas*
  + *numbers.txt* (tekstinis failas skirtas lengvesniam programos testavimui)
  + *eratosthenes.txt* (tekstinis failas sugeneruojamas ***test*** *režimu)*
  + *miller\_rabin.txt* (tekstinis failas sugeneruojamas ***test*** *režimu)*

**Eratosteno rėčio** algoritmas yra faile *prime\_test.py*, ***eratosthenes***funkcijoje.

**Miller-Rabin** algoritmas yra faile *prime\_test.py,* (***miller\_rabin, compute, single\_miller\_test***) funkcijose.

Programos paleidimas: ***python main.py [režimas]***. Programa gali būti vykdoma dvejais būdais – ***test*** ir ***file***. Įrašius neteisingą režimą (sisteminį kintamajį) programa gražina pranešimą: ***usage main.py [test/file]***

***Test*** režimu programa patikrins visus skaičius nuo 1 iki 10000 naudodama abu aukščiau minėtus algoritmus ir automatiškai sugeneruos grafikus. Taip pat, programa sukurs failus *eratosthenes.txt* ir *miller\_rabin.txt*, kuriuose išsaugos informaciją apie algoritmo vykdymo laiką ir testų rezultatus.

***File*** režimu programa patikrins skaičius užrašytus *numbers.txt* faile ir sugeneruos grafikus su skaičiais, kurie buvo patikrinti. Taigi atlikti pirminio skaičiaus testą tam tikram skaičiui galima papildžius failą *numbers.txt* (instrukcijos tame pačiame faile).

**Eratosteno rėčio algoritmas**

***Paaiškinimas*:**

Tarkime turime skaičių n ir norime patikrinti ar n yra pirminis. Atliekame Eratosteno rėčio algoritmą.

1. Susikuriame natūralių skaičių sąrašą [2;N]

(žinome, kad jeigu n < 2, tada skaičius nėra pirminis)

1. Pradedame nuo 2. Iš skaičiaus sąrašo ištriname skaičius, kurie dalijasi iš 2 (be liekanos). Juos iš sąrašo pašaliname.
2. Taip einame per skaičius, kol pasiekiame skaičių, kuris lygus √(N)
3. Jeigu skaičius N dar yra skaičių sąraše, vadinasi jis yra *pirminis*, jeigu skaičiaus nėra – jis *sudėtinis.*
4. [2, 3, 4, 5, 6, 7, ..., N]
5. Daliname iš 2 ir šaliname:

[2, 3, 5, 7, ..., N]

1. Kartojame iki √(N) ir šaliname
2. [19, 29....],

***Pavyzdys:***

Tarkime turime skaičių 13 ir norime patikrintis ar 13 yra pirminis skaičius. Atliekame Eratosteno rėčio algoritmą.

Susikuriame natūralių skaičių sąrašą:

[2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]

Pradedame nuo 2, ir šaliname skaičius, kurie dalinasi iš jo be liekanos:

[2, 3, 5, 7, 9, 11, 13]

Ir t.t., iki tol, kol mūsų paimtas skaičius bus didesnis nei √(13) ~ 3.6 = 3:

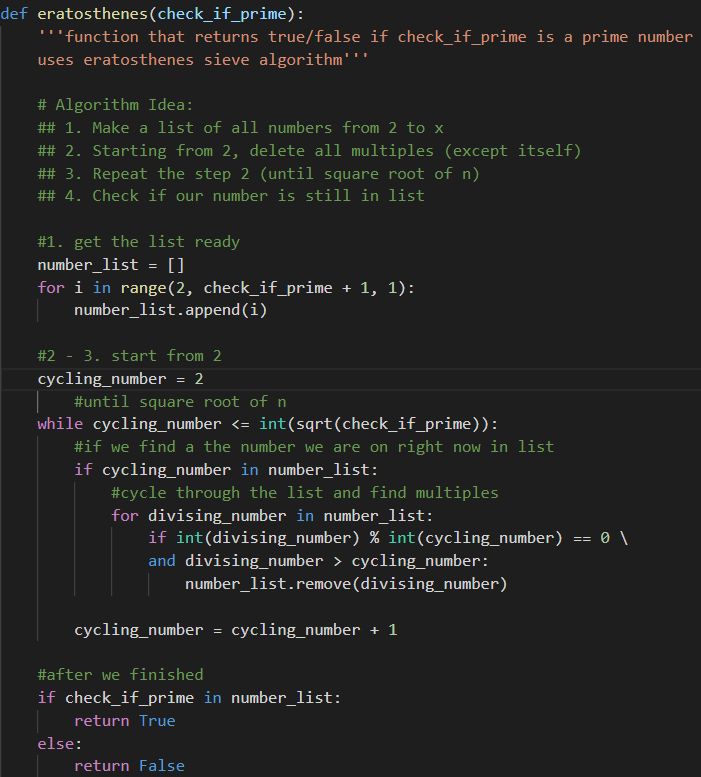
[2, 3, 5, 7, 11, 13]

Tikriname ar skaičius yra numerių sąraše:

[2, 3, 5, 7, 11, 13]

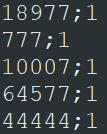
Kadangi skaičius yra numerių sąraše, jis *pirminis* skaičius.

***Kodo funkcija:***

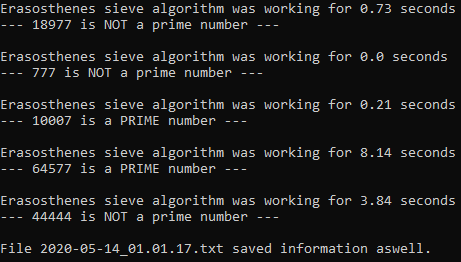


***Kodo funkcionalumas:***

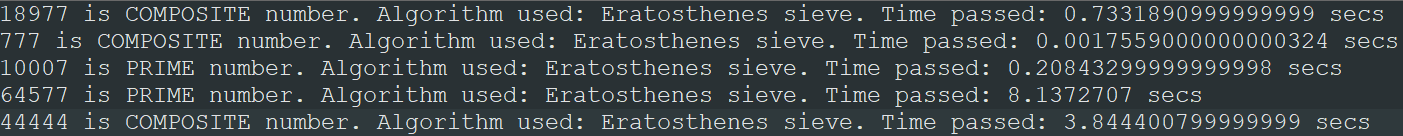
Faile *numbers.txt* paliekame skaičius, kuriuos norime patikrinti. Taip pat užrašome 1 atskirtą kabliataškiu, norėdami panaudoti Eratosteno rėčio algoritmą:



Gautas atsakymas:

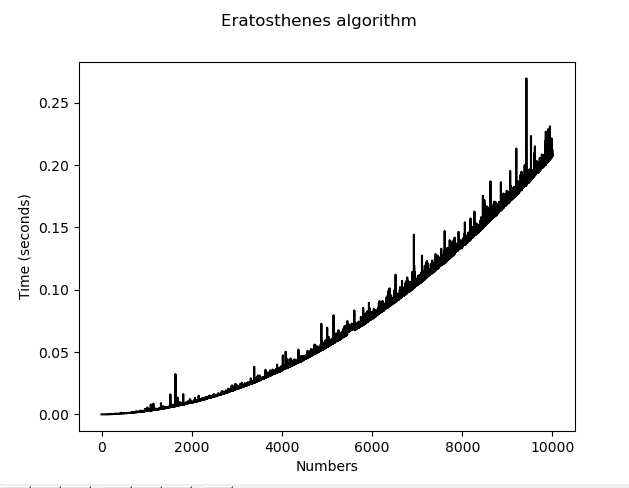


Failas *2020-05-14\_01.01.17.txt*:



***Algoritmo sudėtingumas:***

Pabandžius algoritmą ***test*** režimu gautas grafikas:



**Miller - Rabin algoritmas**

***Paaiškinimas*:**

Tarkime turime skaičių n ir norime patikrinti ar n yra pirminis. Atliekame Miller-Rabin algoritmą. Šis algoritmas nėra tikslus, nes šis testas įrodo, kad skaičius nėra pirminis, vadinasi, jeigu per mažai kartų yra atliktas testas, jis nebegali įrodyti, kad skaičius nėra pirminis.

1. Jeigu skaičius yra lyginis (t.y. dalinasi iš 2 be liekanos) – skaičius nėra pirminis
2. (n – 1) dalindami iš dviejų iki tol, kol gausime liekaną suskaičiuojame n – 1 = 2k \* m

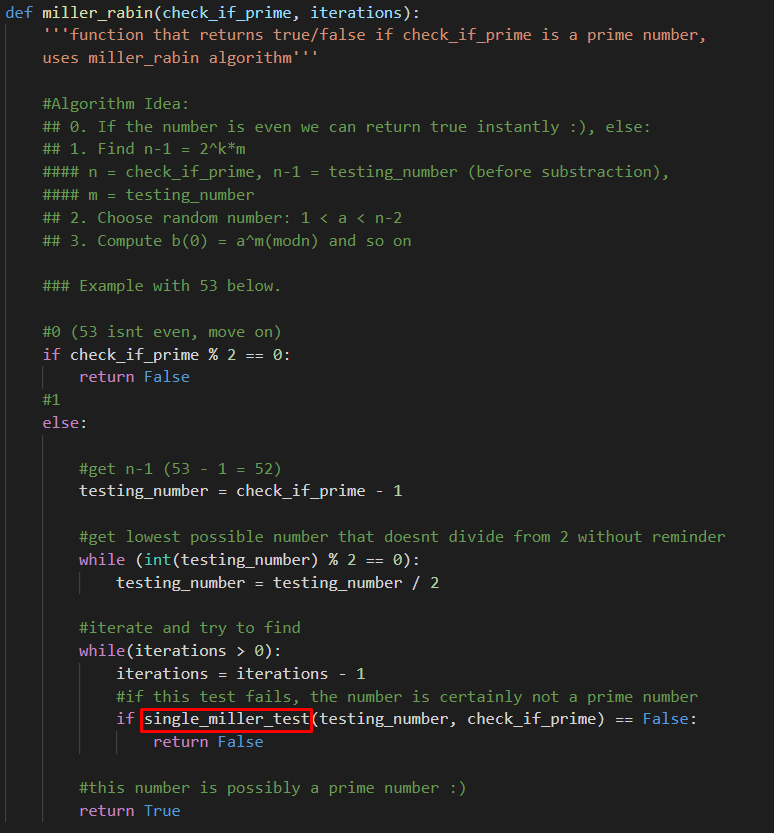
(k, m – sveikieji skaičiai)

1. Sugeneruojame *atsitiktinį* skaičių a, kuris 1 < a < n -1
2. Skaičiuojame b0 = am mod(n). Jeigu suskaičiuotas b0 lygus ±1, darome prielaidą, kad skaičius n yra *turbūt* pirminis
3. Jeigu suskaičiuotas b0 nelygus ±1, tada skaičiuojame b1, o b1 = b02 mod(n).
   1. Jeigu b1 ir t.t. lygus +1, vadinasi n nėra pirminis
   2. Jei b1 ir t.t. lygus -1, n yra *turbūt* pirminis
   3. Jeigu b1 ir t.t. nelygus +1 ar -1, skaičiuojame b2

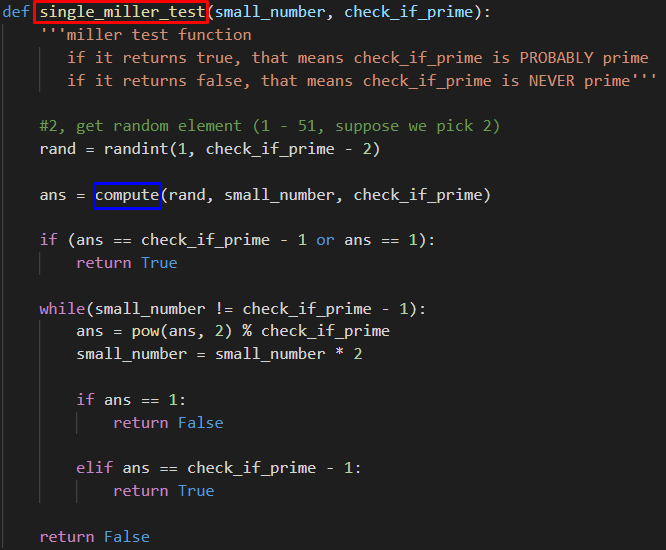
***Pavyzdys:***

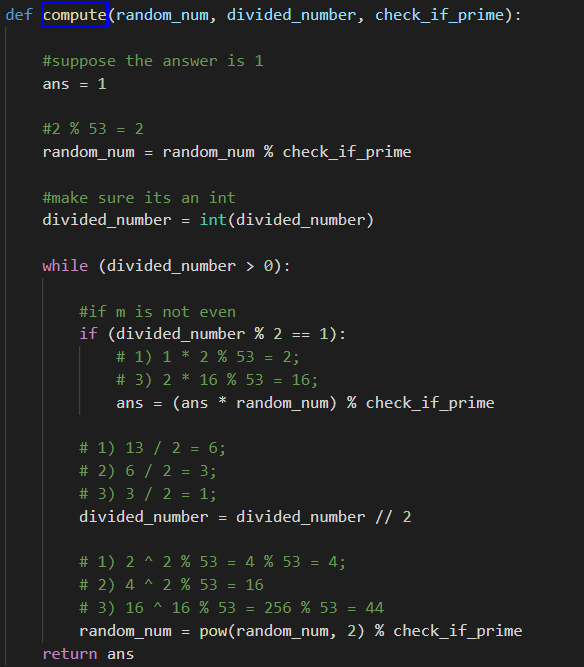
Tarkime turime skaičių 53 ir norime patikrintis ar 13 yra pirminis skaičius. Atliekame Miller-Rabin algoritmą.

1. Skaičius nėra lyginis. Tęsiame toliau.
2. Skaičiuojame (n – 1) = 2k \* m.
   1. m = 13
   2. k = 2
3. Pasirenkame *atsitiktinį* skaičių, kuris didesnis nei 1, bet mažesnis nei 52.
   1. Pasirenkame skaičių 2, a = 2
4. Skaičiuojame b0. b0 = 213 mod53 = 30 mod53
5. Kadangi 30 != ±1, skaičiuojame b1.
6. b1 = 302 mod 53 = -1 mod53.
7. Kadangi gavome -1, skaičius yra *turbūt* pirminis.

***Kodo funkcijos(1) (miller\_rabin)*:**

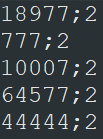
***Kodo funkcijos(2) (single\_miller\_test)*:**



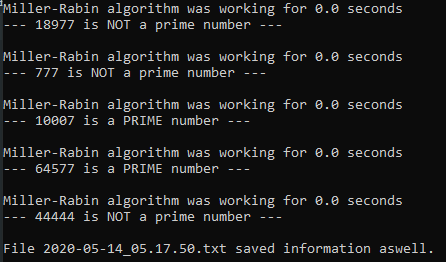
***Kodo funkcijos(3) (compute)*:**

***Kodo funkcionalumas:***

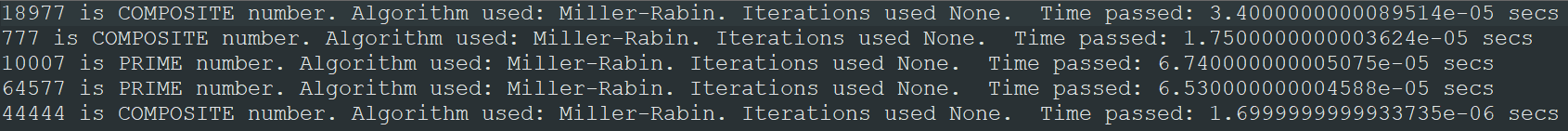
Faile *numbers.txt* paliekame tuos pačius skaičius, kuriuos tikrinome Eratosteno rėčio algoritmu. Taip pat užrašome 2 atskirtą kabliataškiu, nustatydami Miller – Rabin algoritmą:



Gautas atsakymas:

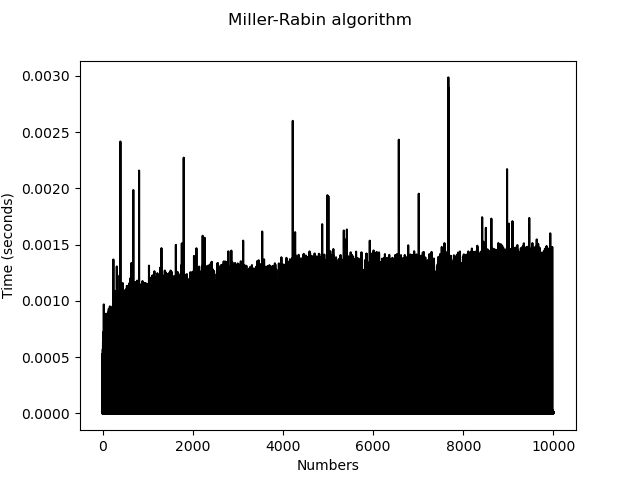


Failas *2020-05-14\_05.17.50.txt*:



***Algoritmo sudėtingumas:***

Pabandžius algoritmą ***test*** režimu gautas grafikas:



**Išvados**

* Eratosteno rėčio algoritmas įrodo, kad skaičius yra pirminis visais atvejais, o Miller – Rabin algoritmas įrodo, kad skaičius nėra pirminis.
* Abiejų algoritmų skaičiaus testavimo ilgis yra tiesiogiai proporcingas skaičiaus ilgiui
* Eratosteno rėčio algoritmas veiks daug ilgiau su ilgesniais skaičiais.
* Didėjant skaičiams Miller – Rabin algoritmas labai minimaliai ilgina algoritmo veikimo ilgį.
* Palyginus to paties skaičiaus testo ilgį abiejuose algoritmuose Miller – Rabin algoritmas yra daug efektyvesnis.
* Ilgiausias trukęs skaičiavimas Eratosteno rėčio algoritme truko virš 0.25 sekundės, o Miller – Rabin – apie 0.0030 sekundės.

**Literatūra:**

<https://research.cs.wisc.edu/techreports/1990/TR909.pdf>

<http://www.mat.uniroma2.it/~schoof/millerrabinpom.pdf>

<https://primes.utm.edu/prove/prove2_1.html>

<https://primes.utm.edu/prove/prove2_3.html#MillersERHTest>