

ECRYP Project - Sieve of Eratosthenes

Krzysztof Rudnicki 307585

Bartłomiej Dybcio 303854

January 8, 2023

1 Description of the used algorithm

Sieve of Eratosthenes is used to find all prime numbers below certain limit
Let's call this limit n

It starts with number 2, which is the first prime number and marks all multiples of this number (up to predefined limit) as composite (not prime), those numbers will be later ignored

Then it takes next available number (3) and does the same thing

This is repeated until there are no more numbers below the limit which are neither prime nor crossed out

Then we return the list of all non-crossed out (prime) numbers

We used more optimized version of this algorithm and cross out composites only up to \sqrt{n} in the main loop

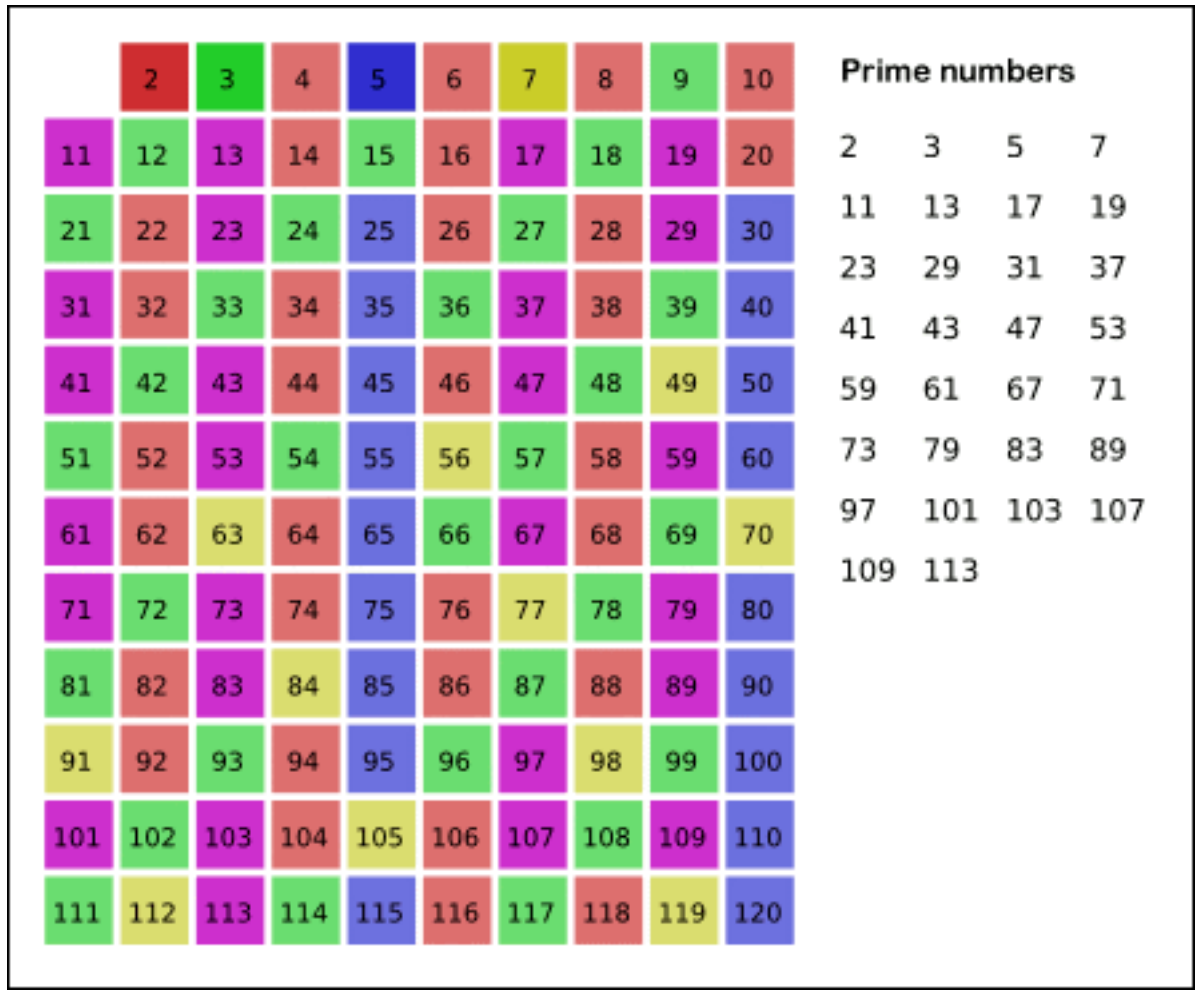


Figure 1: Sieve of erastosthenes example for numbers up to 120, prime numbers on the right, darker colors show prime numbers, lighter colors show numbers which were checked while checking whether given number was prime
SOURCE

2 Functional description of the application

First we define the limit, we name this limit as *num* which will decide how many numbers we will check, either by user interface or we hard code it in We define boolean list which will be used to distinguish between prime and

composite numbers

We start with number 2 and assign it to variable named p

Then we calculate the primes using Sieve of Eratosthenes using nested while loops

External loop goes through numbers smaller than \sqrt{num} , starting with current value of p

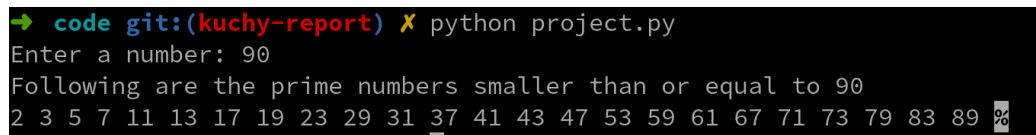
It checks if the number we are currently was checked out by checking the value of boolean table

if it was not checked out it gets a new number which is the p multiplied by 2, then it goes into inner loop

inner loop sets all multiplicities of p as crossed out by setting their value in boolean table to false

then we increment the p and the whole loop repeats until we run out of numbers

we return array of prime numbers to function which prints those numbers



```
→ code git:(kuchy-report) X python project.py
Enter a number: 90
Following are the prime numbers smaller than or equal to 90
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 %
```

Figure 2: Example of full program use

2.1 Input data format

There is single input, variable named num which is the upper limit of numbers to checked

It is a simple int variable, it cannot be less than 2 and has to be a whole number.

2.2 Output text on console

As an output we put out all input prompts which ask for an upper limit, and the list of prime numbers found by our algorithm.

2.3 Format of output data

Output data is a string

2.4 Description of designed code structure

There are three code blocks, `sieve_of_eratosthenes`, `print_sieve` and `main` function executed at the beginning, `main` functions gets no arguments and outputs string consisting of prime numbers, `print_sieve` function receives number which describes how many numbers should be checked in the sieve, similarly `sieve_of_eratosthenes` takes this number as an argument, and returns list of all prime numbers found by the sieve.

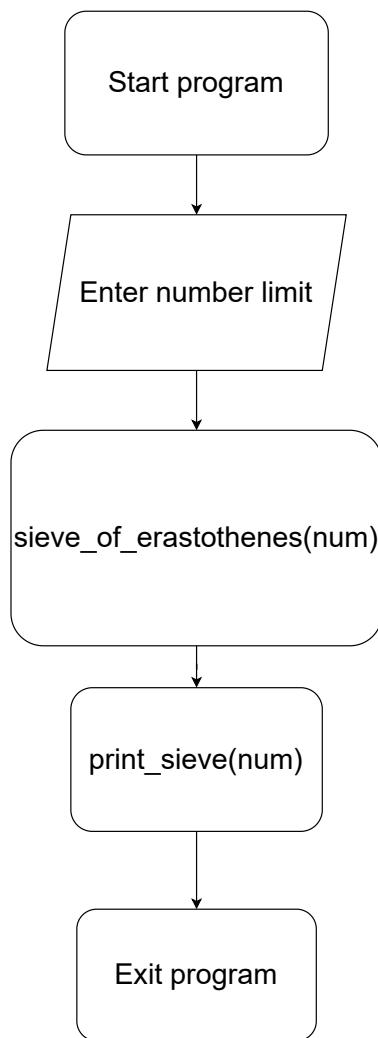


Figure 3: Flowchart of code flow

3 Tests

We did three types of tests, positive tests which inputted prime numbers into the sieve and checked if the sieve correctly validated them as prime, negative tests which inputted composite numbers into the sieve and checked if the sieve correctly determined them to not be prime.

We know which numbers are prime and which are composite based on the reference value which source was given below.

We created two command line arguments:

- primes - which sets maximal number up to which we check correctness of our program in determining whether the number is prime
- composites - which sets maximal number up to which we check correctness of our program in determining whether the number is composite

3.1 Source of reference values

All reference values were taken from: <http://www.naturalnumbers.org/primes.html>

3.2 Correctness of results

All of the tests passed successfully with upper limit as high as 100000

```
PS C:\Users\bartek\GitHub\ECRYPT_PROJECT\code> pytest test_main.py
===== test session starts =====
platform win32 -- Python 3.10.5, pytest-7.2.0, pluggy-1.0.0
rootdir: C:\Users\bartek\GitHub\ECRYPT_PROJECT\code
plugins: anyio-3.6.1
collected 99 items

test_main.py ..... [100%]

===== 99 passed in 0.25s =====
PS C:\Users\bartek\GitHub\ECRYPT_PROJECT\code> pytest test_main.py --primes 45 --composites 67
===== test session starts =====
platform win32 -- Python 3.10.5, pytest-7.2.0, pluggy-1.0.0
rootdir: C:\Users\bartek\GitHub\ECRYPT_PROJECT\code
plugins: anyio-3.6.1
collected 62 items

test_main.py ..... [100%]

===== 62 passed in 0.22s =====
```

Figure 4: Test results for up to 100 numbers

[illegible]

Figure 5: Test results for up to 100000 numbers