Exercise class 8

(week 15)

Introduction to Programming and Numerical Analysis

Class 4 and 8

Rosa Haslund Meyer
Spring 2024

KØBENHAVNS UNIVERSITET

Algorithms

Problem set 5

Tips + comments on Bisection and the sieve of Eratosthenes

Algorithms!

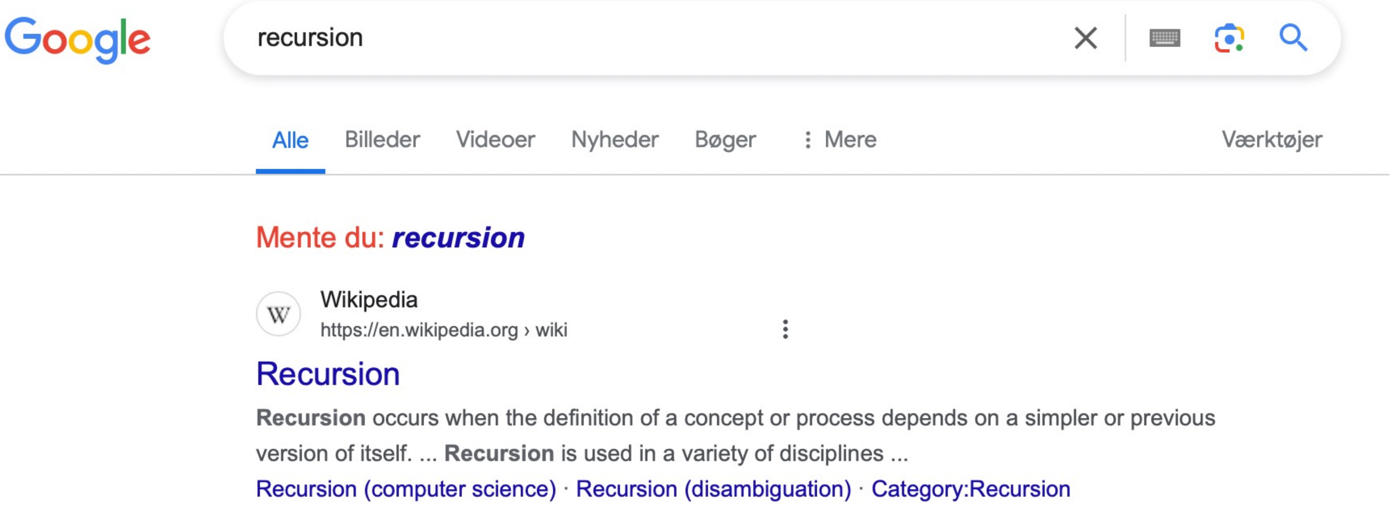
Now to the more **numerical methods-part** of the course!

Algorithms are unambiguous specifications of how to solve classes of problems – or simply put a recipe.

Examples of algorithms you've encountered so far:

- Grid search optimisation
- Numerical solvers use algorithms
- Random number generators
- The split-apply-combine approach could be called an algo…

Recursion



recursion

Mente du: ***recursion***

Wikipedia
https://en.wikipedia.org › wiki

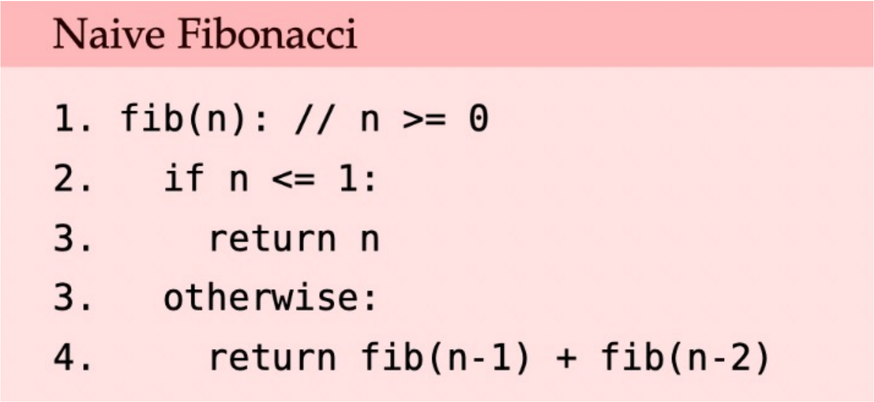
Recursion

Recursion occurs when the definition of a concept or process depends on a simpler or previous version of itself. ... **Recursion** is used in a variety of disciplines ...

Recursion (computer science) · Recursion (disambiguation) · Category:Recursion

Recursion

Consider a classic example, the Fibonacci series – the next number in the series is the two previous numbers summed up:



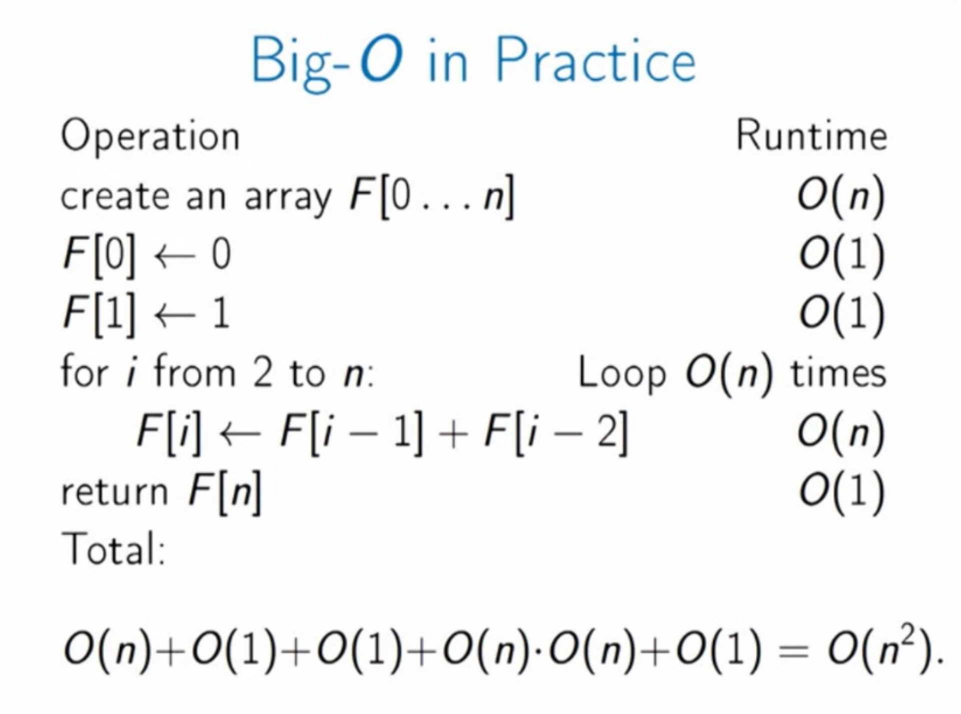
Naive Fibonacci

```
fib(n): // n >= 0
  if n <= 1:
    return n
  otherwise:
    return fib(n-1) + fib(n-2)
```

```
fib(0) = 0
fib(1) = 1
fib(n) = fib(n-1) + fib(n-2)  if n>1
```

For n > 1, the fib function is ***defined in terms of itself***.

Fibonacci and algorithmic complexity



Big-O in Practice

Operation	Runtime
create an array $F[0 \ldots n]$	$O(n)$
$F[0] \leftarrow 0$	$O(1)$
$F[1] \leftarrow 1$	$O(1)$
for i from 2 to n:	Loop $O(n)$ times
$\quad F[i] \leftarrow F[i-1] + F[i-2]$	$O(n)$
return $F[n]$	$O(1)$

Total:

$$O(n)+O(1)+O(1)+O(n) \cdot O(n)+O(1) = O(n^2).$$

Problem set 5

Today’s problem set consists of five increasingly difficult problems.

If you get stuck on a problem, try to write the problem down in pseudo code and/or look at similar algorithms in the lecture notebooks.

For this problem set I recommend you use google/ChatGPT/solutions as little as possible to be sure you understand the concept of recursion and writing algos in general!

Tips for problem 1-3

Problem 1:

- Should be recursive
- You might get inspiration from the pseudo code for the naïve Fibonacci on slide 5
- You can check your answer like by using `math.factorial(5)`

Problem 2:

- Have a look at the sorting-lecture and the visualizations of Bubble-sort
- Write down pseudo code!

Tips for problem 1-3

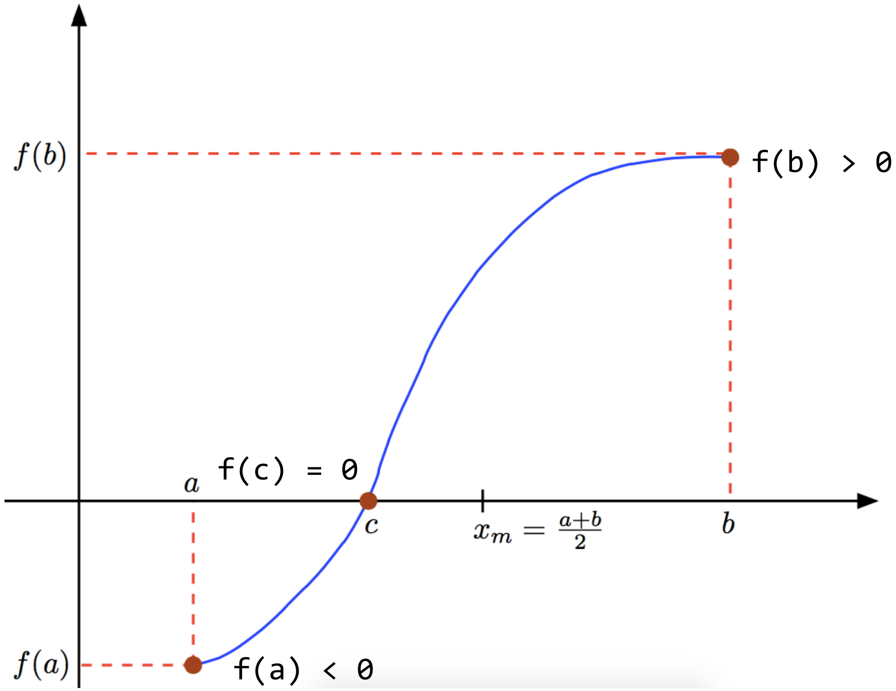
Problem 3:

- Strat by sorting the list
- Forget about the pre-written test function until you think you solved the problem - instead simply print the function by calling the list L and some element, i.e. the number you want to find the index of
- There’s quite a few different ways to implement this algorithm

Tips and comments on problem 4

Problem 4: Bisection

- Bisection is a root finding algo (numerical method)
- It finds an approximation to the root
- There’s no corresponding code or explanation in the lectures, but the algo is written in pseudo code in the problem set notebook
- Some intuition can be found to the right –––>



Tips and comments on problem 5

Problem 5: The sieve of Eratosthenes – finding prime numbers

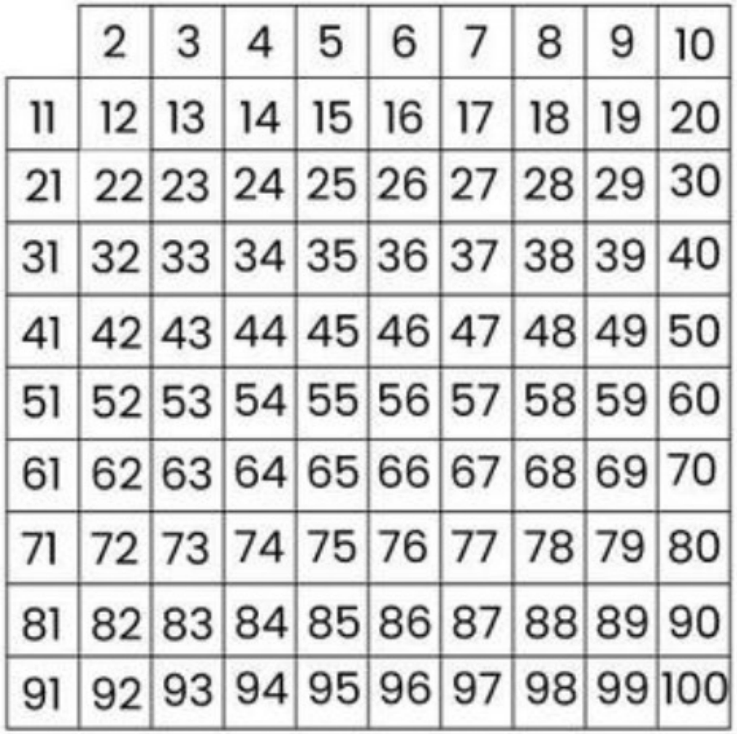
- *Prime numbers are a whole, natural number greater than 1 that cannot be exactly divided by any whole number other than itself and 1 - e.g. 2, 3, 5, 7, 11).*

- Therefore, all numbers that are not primes are products of two smaller numbers – thus we eliminate all numbers that are products of two smaller numbers.

- Intuition on next couple of slides –––>

Intuition on problem 5 – step 0:

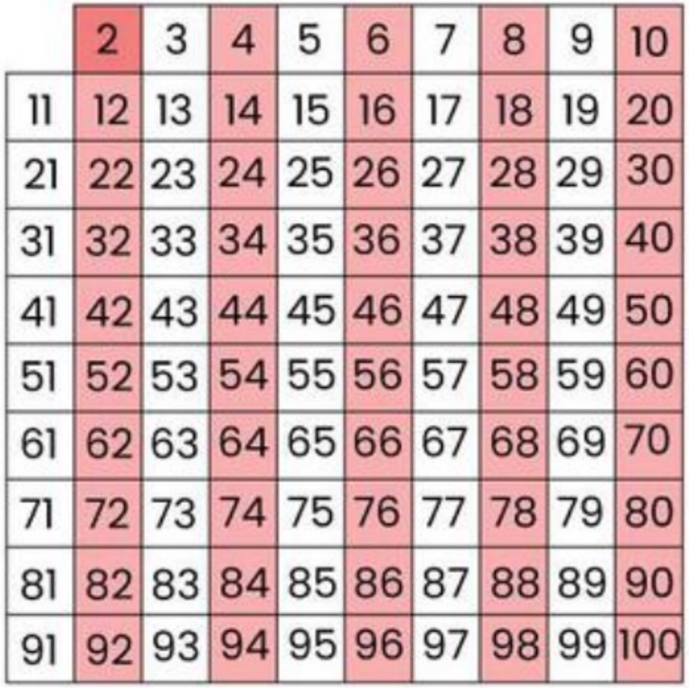
Create a list of all numbers from 2 to 100:



	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

Intuition on problem 5 – step 1:

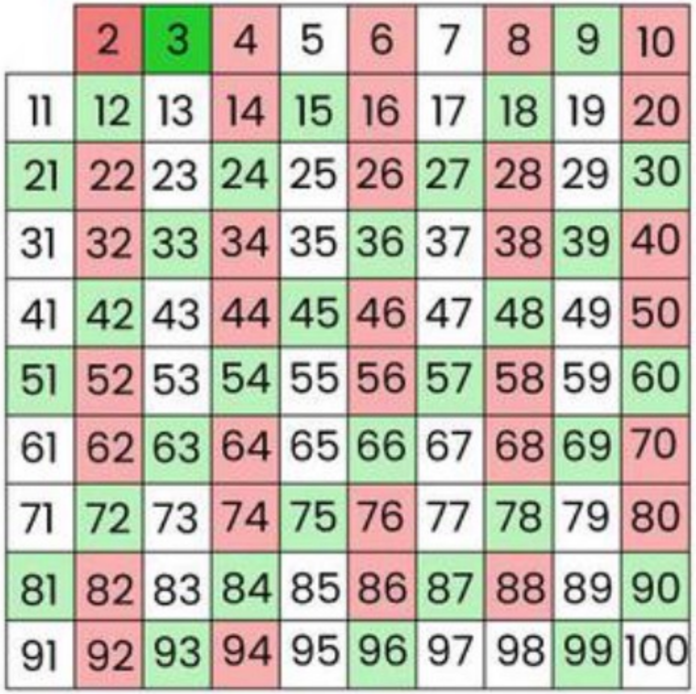
Mark all numbers which are divisible by 2 and are greater than or equal to the square of it:



	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

Intuition on problem 5 – step 2:

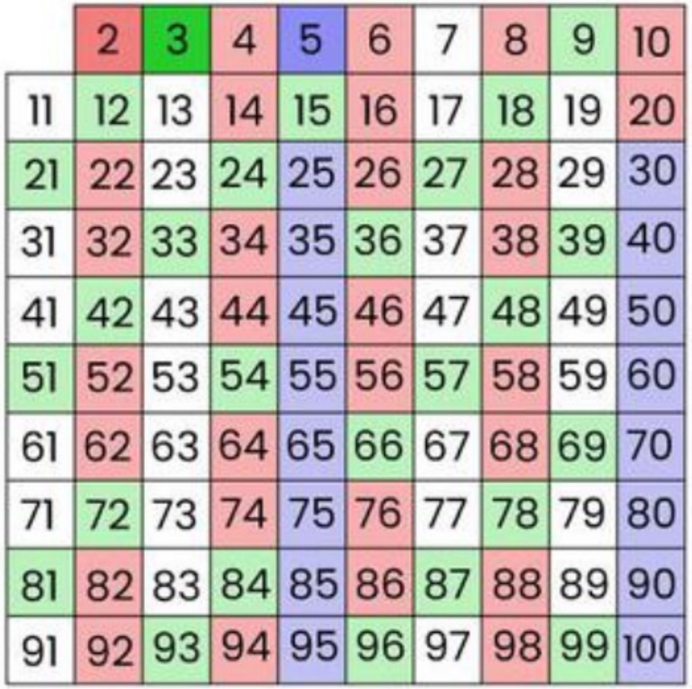
Move to next *unmarked* number, 3, and mark all numbers which are multiples of 3 and are greater than or equal to the square of it:



	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

Intuition on problem 5 – step 3:

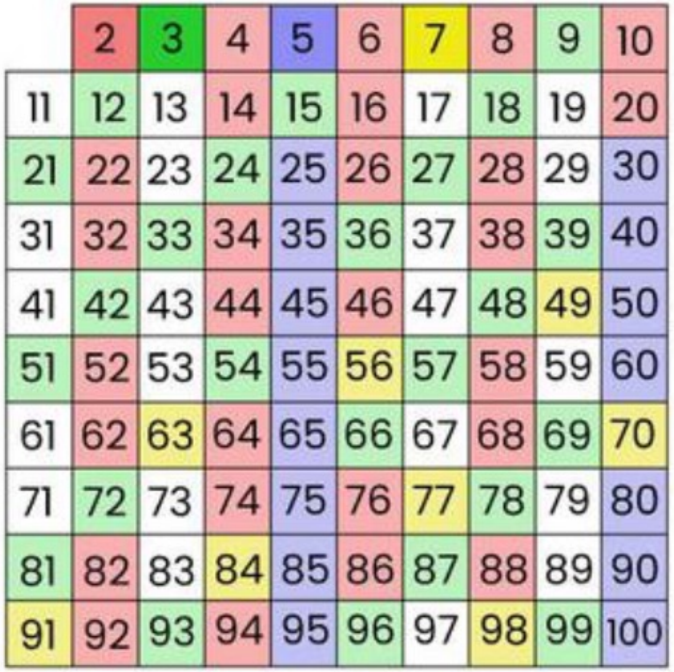
Move to next *unmarked* number, 5, and mark all multiples of 5 and are greater than or equal to the square of it:



	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

Intuition on problem 5 – step 4:

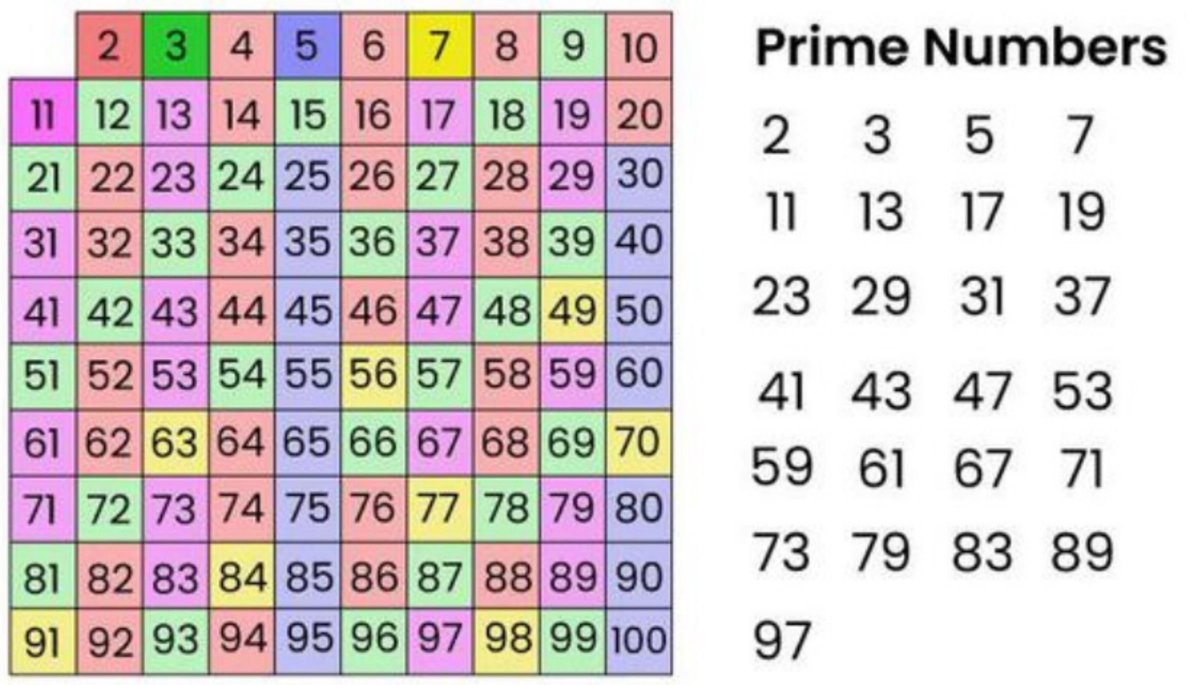
Move to next *unmarked* number, 7, and mark all multiples of 7 and are greater than or equal to the square of it:



	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

Intuition on problem 5 – step 4-?:

This process continues and the final table of prime numbers from 2-100 will look like:



	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

Prime Numbers

2	3	5	7
11	13	17	19
23	29	31	37
41	43	47	53
59	61	67	71
73	79	83	89
97			

Next time…

Video lectures:

- Linear equation systems
- Non-linear equations
- Symbolic math

Exercises – Problem set 6. Solving the Solow model

Remember to hand in your data project by April 14th and do peer feedback by 21st.