Newton School of Technology

General Maths

by Gladden Rumao

Data Structures and Algorithms

Lecture Agenda :

- What is Prime number.
- Multiple approaches.
- Time and Space complexity.

Newton School of Technology

Prime Number

Newton School of Technology

- **A number that can be divided exactly only by itself and 1.**

  Example: 7 is a prime number.
  Example: 12(2*2*3) is not a prime number.

Approach 1

Newton School of Technology

- **Check if the number N is divisible by any integer from 2 to N-1**

  Example: 7 is a prime number.
  Check if **7** is divisible by any number between **2-6**.

Approach 1

Newton School of Technology

- **Check if the number N is divisible by any integer from 2 to N-1**

  Example: 7 is a prime number.
  Check if **7** is divisible by any number between **2-6**.

  Time Complexity: O(N)
  Space Complexity: O(1)

Code

Newton School of Technology

```
def is_divisible(N):
    for i in range(2, N):
        if N % i == 0:
            return True
    return False
```

Newton School of Technology

Do we really need check till N-1 ?

Approach 2

Newton School of Technology

- **Instead for checking from 2 to N-1, we can simply check if the number is divisible by any integer from 2 to √N**

  Example: 7 is a prime number.
  Check if **7** is divisible by any number between **2 to √7**.

Approach 2

Newton School of Technology

- **Instead for checking from 2 to N-1, we can simply check if the number is divisible by any integer from 2 to √N**

Example: 7 is a prime number.
Check if **7** is divisible by any number between **2-√7**.

Time Complexity: O(√N)

Space Complexity: O(1)

Why till Root(N) ?

Newton School of Technology

Consider a natural number N (greater than 1) which exists as a product of two numbers a and b(assume a<=b) that is

- $N = a \cdot b$ where, 1 < a ≤ b < n

By multiplying the relation a ≤ b by a, the relation becomes:

- $a^2 \leq ab$

By multiplying the relation a ≤ b by b, the relation becomes:

- $ab \leq b^2$

From the above-mentioned inequalities, the relation becomes:

- $a^2 \leq ab \leq b^2$

We know that N = a * b, so by replacing a * b with N, the following relation is obtained:

- $a^2 \leq N \leq b^2$

Taking the square root of the equation, considering both a and b as positive numbers, we get:

- $a \leq \sqrt{N} \leq b$

Code

Newton School of Technology

```
def is_divisible(N):
    for i in range(2, int(math.sqrt(N)) + 1):
        if N % i == 0:
            return True
    return False
```

Newton School of Technology

Can we do even better?

Approach 3 (Sieve of Eratosthenes)

Newton School of Technology

- **Used for finding all the prime numbers smaller than or equal to N.**

1. Create a list of size N+1.
2. Start from first prime number k.
3. Mark all the numbers divisible by k in range between k^2 to N.
4. Repeat the process for all unmarked numbers up to √N.

Approach 3 (Sieve of Eratosthenes)

Newton School of Technology

- **Example: 100**

1. Create a list of size 101.

	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

Approach 3 (Sieve of Eratosthenes)

Newton School of Technology

- **Example: 100**

1. Create a list of size 101.
2. Start from first prime number k=2.

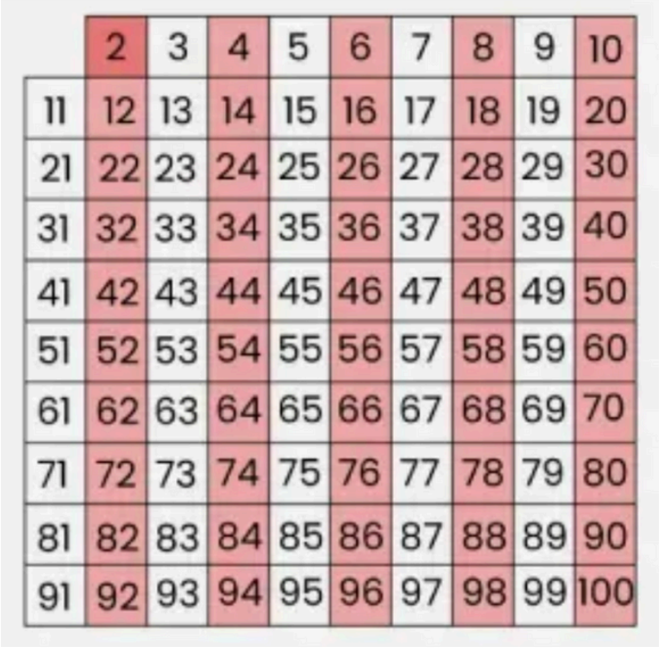
	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

Approach 3 (Sieve of Eratosthenes)

Newton School of Technology

- **Example: 100**

1. Create a list of size 101.
2. Start from first prime number k=2.
3. Mark all the numbers divisible by k in range between k^2 to N.



	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

Approach 3 (Sieve of Eratosthenes)

Newton School of Technology

- **Example: 100**

1. Create a list of size 101.
2. Start from first prime number k=2.
3. Mark all the numbers divisible by k in range between k^2 to N.
4. Repeat the process for all unmarked numbers up to √N.

Prime Numbers: 2, 3

	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

Approach 3 (Sieve of Eratosthenes)

Newton School of Technology

- **Example: 100**

1. Create a list of size 101.
2. Start from first prime number k=2.
3. Mark all the numbers divisible by k in range between k^2 to N.
4. Repeat the process for all unmarked numbers up to √N.

Prime Numbers: 2, 3, 5

	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

Code

Newton School of Technology

```
def sieve_of_eratosthenes(N):
    is_prime = [True] * (N + 1)  # True means "assumed prime"
    is_prime[0] = is_prime[1] = False  # 0 and 1 are not prime numbers

    for k in range(2, int(math.sqrt(N)) + 1):
        if is_prime[k]:  # If k is a prime number
            for multiple in range(k * k, N + 1, k):
                is_prime[multiple] = False

    primes = []
    for num in range(len(is_prime)):
        if is_prime[num]:
            primes.append(num)
    return primes
```

Time & Space Complexity

Newton School of Technology

Time Complexity : n*log(log(n))
Space Complexity : O(N)