A Comparative Study of Prime Sieves: Atkin, Sundaram, and Eratosthenes

Sieve Of Eratothenes

-Sree Keerthi Maripally

The **sieve of Eratosthenes** is an ancient algorithm for finding all prime numbers up to any given limit.

It does so by iteratively marking as composite (i.e., not prime) the multiples of each prime, starting with the first prime number, 2. The multiples of a given prime are generated as a sequence of numbers starting from that prime, with constant difference between them that is equal to that prime. Once all the multiples of each discovered prime have been marked as composites, the remaining unmarked numbers are primes.

Example:
Let us find primes less than 100

1	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

2 is next unmarked number
We will strike off multiples of 2

Prime	start	increment
2	4	2

1	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

3 is next unmarked number
We will strike off multiples of 3

Prime	start	increment
2	4	2
3	9	6

1	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

5 is next unmarked number
We will strike off multiples of 5

Prime	start	increment	prime*prime<=100
2	4	2	4
3	9	6	9
5	15	10	25

1	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

7 is next unmarked number
We will strike off multiples of

Prime	start	increment	prime*prime<=100
2	4	2	4
3	9	6	9
5	15	10	25
7	49	14	49

1	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

11 is next unmarked number
We will strike off multiples of 11

Prime	start	increment	prime*prime<=100
2	4	2	4
3	9	6	9
5	15	10	25
7	49	14	49
11			121>100 (so we can stop here)

Program In Java:

```
import java.util.*;
import java.lang.*;
class sieve_ex{
	public static void main(String args[]){
		double startTime = System.currentTimeMillis();
		int range;
		range = 100;
		boolean numbers[] = new boolean[range];


		int checkprime;
		int count=0;
		int inc;
		numbers[0]=true;
		numbers[1]=true;

		for(checkprime = 2; checkprime*checkprime<range;checkprime++){
			if(checkprime==2)
				inc=2;
			else
				inc=checkprime*2;


			for(int i=checkprime*checkprime;i<range;i+=inc){
				if(numbers[checkprime]==false){
					numbers[i]=true;
```

```
                }
            }
            }
            for (int i = 0; i < range; i++) {
                if (numbers[i] == false) {
                    System.out.println(i+" ");

                    count++;
                }
                }

            System.out.println(count);
            double endTime = System.currentTimeMillis();
            System.out.println((endTime-startTime)/1000);
        }
}
```

Time Complexity:O(n);

Sieve Of Atkin:

Compared with the ancient Sieve of Eratosthenes, which marks off multiples of primes, it does some preliminary work and then marks off multiples of squares of primes, that's why it has a better theoretical asymptotic complexity with **Complexity of (N / (log (log N)))**

How Sieve Of Atkin works:

The Sieve of Atkin algorithm works similarly to Sieve of Eratosthenes to filter out the composite numbers from a list of numbers, but this algorithm works in terms of **modulo-60 remainders**.

- All remainders are modulo-sixty remainders (divide the number by 60 and return the remainder).
- All numbers, including *x* and *y*, are positive integers.
- Flipping an entry in the sieve list means to change the marking (prime or nonprime) to the opposite marking.
- This results in numbers with an odd number of solutions to the corresponding equation being potentially prime (prime if they are also square free), and numbers with an even number of solutions being composite.

(square-free integer is an integer which is divisible by no square number other than 1. That is, its prime factorization has exactly one factor for each prime that appears in it. For example, 10 = 2 · 5 is square-free, but 18 = 2*3*3 is not, because 18 is divisible by 9 = 3^2. The smallest positive square-free numbers are 1, 2, 3, 5, 6, 7, 10)

So we first assume all the numbers within limit to be composite, and then apply filter or sieve on them. If while any filter, the number appears to be prime, we mark it as prime and move on to the next number.

The filter or sieve in this algorithms works mainly 4 cases or layers:

Case 1: If limit is greater than 2 or 3:

The algorithm treats 2, and 3 as special cases and just adds them to the set of primes to start with.

Case 2: if 4x^2+y^2=n is odd and modulo-12 remainder is 1 or 5

Since all numbers with modulo-60 remainders 1, 13, 17, 29, 37, 41, 49, or 53 have a modulo-12 remainder of 1 or 5. Therefore, for this filter as well, we have to check if the number is 1 or 5 when taken modulo with 12.

Also, These numbers are prime if and only if the number of solutions to 4x2+y2=n is odd and the number is square-free.

Case 3: if 3x^2+y^2=n is odd and modulo-12 remainder is 7

All numbers with modulo-60 remainder 7, 19, 31, or 43 have a modulo-6 remainder of 1.

These numbers are prime if and only if the number of solutions to 3x2 + y2 = n is odd and the number is square-free.

Case 4: if 3x^2-y^2=n is odd and modulo-12 remainder is 11

All numbers with modulo-60 remainder 11, 23, 47, or 59 have a modulo-12 remainder of 11.

These numbers are prime if and only if the number of solutions to 3x2 – y2 = n is odd and the number is square-free.

Case 5: Filtering out all the residual primes which have not yet been found

Due to the filtering of the Sieve of Atkin algorithm, there might be some prime numbers who have been discarded or not found in the above cases.

So to find out those, select all non-primes within limit, and mark all their squares as non-primes.At the end of all of the filters above, the positions in the Sieve with a true value will be the list of primes within limit.

Sieve of Atkin algorithm step-by-step:

1) Create a results list, filled with 2, 3, and 5.
2) Create a sieve list with an entry for each positive integer; all entries in this list should initially be marked non-prime.
3) For each entry number n in the sieve list, with modulo-sixty remainder r:
    1. If r is 1, 13, 17, 29, 37, 41, 49, or 53, flip the entry for each possible solution to 4x2 + y2 = n.
(effectively perform n%12,if result is 1 or 5 then flip the status in sieve list)
    2. If r is 7, 19, 31, or 43, flip the entry for each possible solution to 3x2 + y2 = n.
(effectively perform n%12, if result is 7 then flip the status in sieve list)
    3. If r is 11, 23, 47, or 59, flip the entry for each possible solution to 3x2 – y2 = n when x > y.
(effectively perform n%12, if result is 11 then flip the status in sieve list)
    4. If r is something else, ignore it completely…
4) Start with the lowest number in the sieve list.
5) Take the next number in the sieve list, still marked prime.
6) Include the number in the results list.
7) Square the number and mark all multiples of that square as non-prime. Note that the multiples that can be factored by 2, 3, or 5 need not be marked, as these will be ignored in the final enumeration of primes.
8) Repeat steps four through seven.

Example:
Lets take range of 100

1	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

Let x and y be two variables;

	Start	until
x	1	x*x<100
Y	1	y*y<100

If (4x^2+y^2)%12=1or5 mark it
If (3x^2+y^2)%12=7 mark it
If (3x^2-y^2)%12=11 mark it

x	y	4*x*x+y*y<=100	3*x*x+y*y<=100	3*x*x-y*y<=100 and x>y
1	1	4*1*1+1*1=5%12=5	3*1*1+1*1=3	
1	2	8	7%12=7	
1	3	13%12=1	12	
1	4	20	19%12=7	
1	5	29%12=5	28	
1	6	40	49	
1	7	53%12=5	52	
1	8	68	67%12=7	
1	9	85%12=1	84	
2	1	17%12=5	13	11%12=11
2	2	20	16	
2	3	25%12=1	21	
2	4	32	28	
2	5	41%12=5	37	
2	6	52	48	
2	7	65%12=5	61	
2	8	80	76	
2	9	97%12=1	93	
3	1	37%12=1	28	26
3	2	40	31%12=7	23%12=11
3	3	45	36	
3	4	52	43%12=7	
3	5	61%12=1	52	
3	6	72	63	
3	7	85%12=1	76	

3	8	100	92	
x	y	4*x*x+y*y<=100	3*x*x+y*y<=100	3*x*x-y*y<=100 and x>y
4	1	65%12=5	49	47%12=11
4	2	68	52	44
4	3	73%12=1	57	39
4	4	80	64	
4	5	89%12=5	73	
4	6	100	84	
4	7		97	
5	1		76	74
5	2		79%12=7	71%12=11
5	3		84	64
5	4		91%12=7	59%12=11
5	5		100	
6	3			99
6	4			92
6	5			83%12=11

(note: 85 and 65 first becomes true then becomes false as they came two times)

1	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

Now some of the numbers are not prime these can be removed by selecting non prime(unmarked) number and Square the number and mark all multiples of that square as non-prime. Note that the multiples that can be factored by 2, 3, or 5 need not be marked, as these will be ignored in the final enumeration of primes. Means unmark all numbers which are not square free(like 25)

1	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

Program in java:

```
import java.util.*;
class sieve_atkin{
   public static void main(String args[]){
      int limit=100;
      double startTime = System.currentTimeMillis();


      boolean numbers[] = new boolean[limit+1];
      for(int i=0;i<=limit;i++){
         numbers[i]=false;
      }

      for(int x=1;x*x<=limit;x++){
         for(int y=1;y*y<=limit;y++){
            int temp = 4*x*x+y*y;
            if(temp<=limit && (temp%12==5 || temp%12==1)){
               numbers[temp]^=true;
            }

            temp = 3*x*x+y*y;
            if(temp<=limit && (temp%12==7)){
               numbers[temp]^=true;
            }

            temp = 3*x*x-y*y;
            if(x>y&& temp<=limit && (temp%12==11)){
               numbers[temp]^=true;
            }
         }

      }

      for(int multiples=5;multiples*multiples<=limit;multiples++){
         if(numbers[multiples]){
            for(int i=multiples*multiples;i<=limit;i+=multiples){
               numbers[i]=false;
            }
         }
      }
```

```
        int count=2;
        for(int a=5;a<=limit;a++){
            if(numbers[a]){
                System.out.println(a+" ");


                count++;

            }
        }
        double endTime = System.currentTimeMillis();
        System.out.println(count);
        System.out.println((endTime-startTime)/1000);
    }
}
```

Time Complexity:O(range)

Sieve of Sundaram:

In general Sieve of Sundaram, produces primes smaller than (2*x + 2) for given number x. Since we want primes smaller than n, we reduce n-1 to half. We call it nNew.

1) nNew = (n-1)/2;

For example, if n = 102, then nNew = 50.

if n = 103, then nNew = 51

2) Create an array **marked[n]** that is going to be used to separate numbers of the form i+j+2ij from others where 1 <= i <= j

3) Initialize all entries of marked[] as false.

4) Mark all numbers of the form i + j + 2ij as true where 1 <= i <= j

Loop for a) i=1 to nNew

b) j=I to I+j+2ij<=nNew

5) Remaining primes are of the form 2i + 1 where i is index of NOT marked numbers. So print 2i + 1 for all i such that marked[i] is **false**.

How does this work?

When we produce our final output, we produce all integers of form 2x+1 (i.e., they are odd) except 2 which is handled separately.

Let q be an integer of the form 2x + 1.

q is excluded if and only if x is of the form i + j + 2ij. That means,

q = 2(i + j + 2ij) + 1

=2i+2j+4ij+1

=1(2i+1)+2j(2i+1)

= (2i + 1)(2j + 1)

So, an odd integer is excluded from the final list if and only if it has a factorization of the form (2i + 1)(2j + 1) which is to say, if it has a non-trivial odd factor.

Example:

Lets take up to range 100

new_range = (100-1)/49

1	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	

i	j	I+j+2*I*j
1	1	4
1	2	7
1	3	10
1	4	13
1	5	16
1	6	19
1	7	22
1	8	25
1	9	28
1	10	31
1	11	34
1	12	37
1	13	40
1	14	43
1	15	46
1	16	49
2	2	12
2	3	17
2	4	22
2	5	27
2	6	32
2	7	37
2	8	42
2	9	47
3	3	24
3	4	31
3	5	38
3	6	45
4	4	40
4	5	49

(i=1 to new_range

J=I to (I+j+2ij)<=new_range)

mark all I+j+2*I*j numbers

1	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	

I	2*I+1(primes)
1	3
2	5
3	7
5	11
6	13
8	17
9	19
11	23
14	29
15	31
18	37
20	41
21	43
23	47
26	53
29	59
30	61
33	67
35	71
36	73
39	79
41	83

44 89

48 97

Program in java:

```
import java.util.*;
class sieve_sundaram{
   public static void main(String args[]){
      double startTime = System.currentTimeMillis();
      long range;
      long new_range;
      range = 100;
      new_range=(range-1)/2;
      int count=1;
      boolean numbers[] = new boolean[(int)(new_range+1)];

      for(long i=1;i<new_range;i++){
         for(long j=i;(i+j+2*i*j)<=new_range;j++){
            numbers[(int)(i+j+2*i*j)]=true;
         }
      }

      for(long i=1;i<=new_range;i++){
         if(numbers[(int)i]==false){
            System.out.println((2*i+1)+" ");
            count++;
         }
      }
      double endTime = System.currentTimeMillis();
      System.out.println(count);
      System.out.println((endTime-startTime)/1000);
   }
}
```

Time Complexity:O(n log n)

Time taken(in sec) to count number of primes in given range:

Range	Sieve of eratothenes (approx)	Sieve of Atkin (approx)	Sieve of Sundaram (approx)
100	0	0	0
10000	0	0	0
10^6	0.004	0.015	0.004
10^7	0.061	0.137	0.047
10^8	1.041	2.766	1.466
10^9	13.469	38.341	28.625

Observation:

Though sieve of atkin has better theoretical approach it is complex and for larger numbers it takes more time to calculate. Similarly even though sieve of sundaram reduces the range to half, but for larger numbers it takes more time. Where as sieve of eratothenes approach takes much less time than other two approaches.

Hence from above analysis we can conclude Sieve of Eratothenes is more efficient algorithm to find prime numbers up to given range.