```
1  PROMPT number
2  GET number
3       FOR I in range of number
4          Array ← True
5  P ← 2
6  WHILE ( p * p <= number )
7      IF array[p] ← True
8          FOR I in range (p*p, number, p)
9              Array[i] ← False
10         ELSE
11             P += 1
12 FOR p in range (2, number)
13     IF array[p] == True
14         Prime numbers ← p
15 PRINT Prime numbers
```

The time complexity for this algorithm is O(n log n) because we are using many loops in this algorithm. The first one in line 3 because we are appending everything as True and that is O(n). There is another loop in line 6 and an inner loop in line 8 which makes it O (n log n) because we take the first loop and then we loop again until we break the first loop. There is another loop to append all of the true numbers in the new list prime numbers and that is O(n), This might be a good algorithm because it is fast for smaller numbers but might take a little bit longer if we have a long number because then the data increases.

	P	I (multiple)	Array (boolean list)
			[t,t,t,t,t,t,t,t,t,t,t]
5	2	/	
6	2	/	
7	2	/	
8	2	4	
9	2	4	[t,t,t,t,F,t,t,t,t,t,t]
8	2	6	[t,t,t,t,F,t,t,t,t,t,t]
9	2	6	[t,t,t,t,F,t,F,t,t,t,t]
8	2	8	[t,t,t,t,F,t,F,t,t,t,t]
9	2	8	[t,t,t,t,F,t,F,t,F,t,t]
8	2	10	[t,t,t,t,F,t,F,t,F,t,t]
9	2	10	[t,t,t,t,F,t,F,t,F,t,F]
11	3	/	

	P	I (multiple)	Array (boolean list)
6	3	1	[t, t, t, t, F, t, F, t, F, t, F]
7	3	1	$\downarrow$
8	3	6	
9	3	6	
8	3	9	
9	3	9	[t, t, t, t, F, t, F, t, F, F, F]
6	4 (breaks)	1	[t, t, t, t, F, t, F, t, F, F, F]
12	(appends the list From index 2 $\rightarrow$		2, 3, 5 7

$\downarrow$

15 prints $\rightarrow$ [2, 3, 5, 7]