Sieve of Eratosthenes

작성일시	@2022년 8월 3일 오후 5:42
강의 번호	Algo
유형	
자료	
Property	
Date	
복습	☐
속성	

Sieve of Eratosthenes (에라토스테네스의 체)

- 소수 prime number를 찾아내는 알고리즘
- 소수는 2 이상의 정수에서 1과 그 수 자체로만 나눌 수 있는 수
- 소수는 나열되어있는 구간이 불규칙하므로 임의로 찾기 힘들다.

4는 2와 2로, 6은 2와 3으로 나눌 수 있다. 따라서 소수가 아니다. 이러한 숫자들은 합성수 즉 숫자와 숫자가 합성된 것이라고 할 수 있다. 반대로 말하면 합성수가 아닌 숫자들이 바로 소수인 것이다.

소수 prime number

소수는 2이상의 정수 중에서 1과 그 수 자신 외에는 나눌 수 없는 숫자.

10이하에서는 2, 3, 5, 7이 소수에 해당한다.

소수 prime 에서의 '소'는 합성되지 않은 소박한 숫자 의미의 '소'를 뜻한다.

모든 수의 소(근본)를 의미하기도 한다.

소수인지 아닌지를 구분하는 것은 의외로 어렵다.

소수의 내용만으로 보면 어렵지 않아보이지만 사실은 의외로 아주 어렵다. 무엇이 어려운지도 바로 찾아내기 힘들다.

예로 3의 배수는 3, 6, 9 ,12 ,15 처럼 3개의 간격으로 나열된다. 따라서 1에서 100까지의 사이에 있는 3의 배수를 찾는 일은 간단하다. 하지만 소수는 규칙성이 없기때문에 간격이 불규칙하고 랜덤하다.

즉, 소수는 한번에 열거하기가 어렵다.

그러면 어떻게 구할 수 있을까? 가장 먼저 떠오르는 방법은 하나하나 그 수보다 작은 숫자로 나누어보고 나눌 수 있는지의 여부를 확인하는 것이다. 2에서 100까지의 소수를 찾으려면 먼저 2로 나눌 수 잇는 수를 모두 지우고 그 다음 3으로 나눌수 있는 숫자를 모두 지우고 그 다음 4로 나눌 수 있는 수를 모두 지우고……..마지막에는 99로 나눌 수 있는 수를 지우는 방법이다. 하지만 이 방법은 비효율적이고 수의 범위가 커지는 경우에는 많은 시간이 소모될 것이다.

	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

고대 그리스의 과학자인 에라토스테네스는 위의 방법을 개선하여 소수를 효율적으로 발견하는 방법을 알아냈다. 그의 이름을 따서 ‘에라토스테네스의 체’라고 부른다.

에라토스테네스의 체

어떤 수 이하의 범위에 존재하는 소수를 찾고 싶은 경우

‘ 그 수의 제곱근 보다 작은 소수의 배수만 없애면 남은 수가 소수다’

라는 생각을 바탕으로 소수를 찾는 방법이다.

예를 들어 100이하의 소수를 모두 찾아내려면

100의 제곱근 이하 소수를 선택한다.

루트 100의 제곱근은 10이다. 즉 제곱하면 100이 되는 수는 10이므로 10이하에서의 소수는 2, 3, 5, 7 4개이다.

우선 2에서 100까지의 표에서 2로 나눌 수 있는 즉 2의 배수 중 2를 제외한 나머지 모두를 삭제한다.

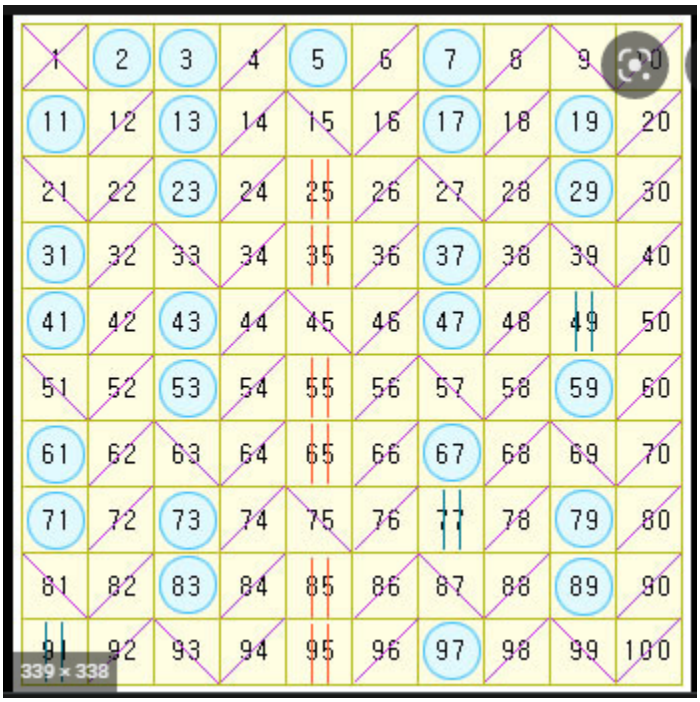
맨 처음에 소수인 2를 발견한 후 2의 배수를 모두 지운다.

	2	3		5		7		9	
11		13		15		17		19	
21		23		25		27		29	
31		33		35		37		39	
41		43		45		47		49	
51		53		55		57		59	
61		63		65		67		69	
71		73		75		77		79	
81		83		85		87		89	
91		93		95		97		99	

이런 방식으로 3보다 큰 수에 대해서도 n*n부터 지워주면 된다.

	2	3		5		7			
11		13				17		19	
		23		25				29	
31				35		37			
41		43				47		49	
		53		55				59	
61				65		67			
71		73				77		79	
		83		85				89	
91				95		97			

같은 방식으로 5와 7의 배수들도 모두 삭제한 결과는 아래와 같다.



이 방법을 사용하면 1에서 100까지의 모든 숫자를 해당 숫자보다 작은수로 나눌 수 있는지 하나 하나 전부 순서대로 확인하는 것보다는 훨씬 더 빨리 소수를 찾을 수 있다.

에라토스테네스의 체 알고리즘

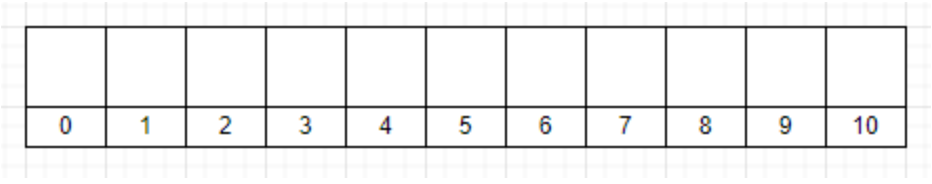
- 에라토스테네스의 체는 크개 3개의 처리로 구성한다.
  1. 어떤 수 이하의 모든 정수 데이터를 준비
  2. 어떤 수의 제곱근 보다 작은 소수의 배수들을 차례로 제거한다.
  3. 마지막까지 남은 수들을 출력한다.

우선 10이하의 소수를 구하는 경우로 생각해보자.

1. 10이하의 정수 데이터들을 준비한다.
2. 10의 제곱근 약 3. 16보다 작은 소수의 배수를 차례대로 제거한다.
3. 마지막까지 남은 수들을 출력한다.

10 이하의 정수 데이터들을 준비한다.

먼저 체의 대상이 되는 10까지의 정수를 데이터로 준비하자. 여기에서는 11개의 요소를 가지는 정수형 배열을 준비한다. 배열의 이름은 arr로 정하자. 첨자는 0부터 시작하기 대문에 첨자를 10까지로 동일하게 사용하려면 요소를 11개 준비해야 한다.



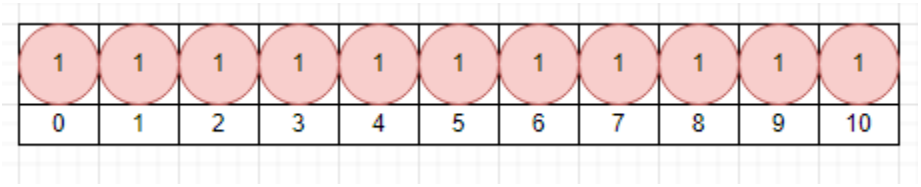
요소는 그 첨자가 소수인지의 여부를 판정하는 데이터를 넣는 것으로 사용하자.

즉, 소수가 아닌 것으로 판정된 첨자의 요소에 '소수가 아니다'라는 것을 나타내는 데이터를 넣자.

즉, 소수의 가능성이 있는 경우에는 1을 대입하고 소수가 아닌 경우에는 0을 대입한다.

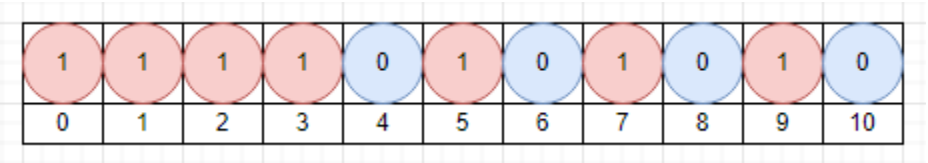
위에서의 개념 소개에서는 소수가 아닌 수를 제거하는 방식을 0을 대입하는 방식으로 처리한다. 따라서 초기값을 1로 전부 대입해 둔다. 그리고 소수가 아닌 것으로 선정된 수(첨자)의 요소에는 0을 대입한다.

이러한 방식을 통해 소수의 배수를 모두 제거(0을 대입)한 후 마지막에 '1이 남아있는 요소들은 모두 소수이고 0이 대입되어 있는 요소는 소수가 아니다'라고 구별 할 수 있다.



10의 제곱근 약 3. 16보다 작은 소수(2,3)의 배수를 차례대로 제거한다.

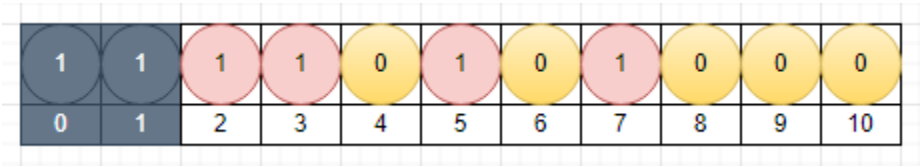
먼저 2의 배수들을 모두 지운다. 이때 지운다의 의미를 0을 대입한다는 표현으로 적용하자.



다음으로 3의 배수들을 모두 지운다. 이때 지운다의 의미를 0을 대입한다는 표현으로 적용하자.

마지막까지 남은 수들을 출력한다.

제곱근 이하의 소수들의 모든 배수를 제거(요소의 값이 0)했다면 나머지는 모두 소수(요소의 값이 1)뿐이다. 그럼 이제 첨자가 2이상이고 1이 들어있는 요소의 첨자들만 뽑아내면 그것들이 바로 10이하의 소수가 된다.



에라토스테네스의 체 순서도

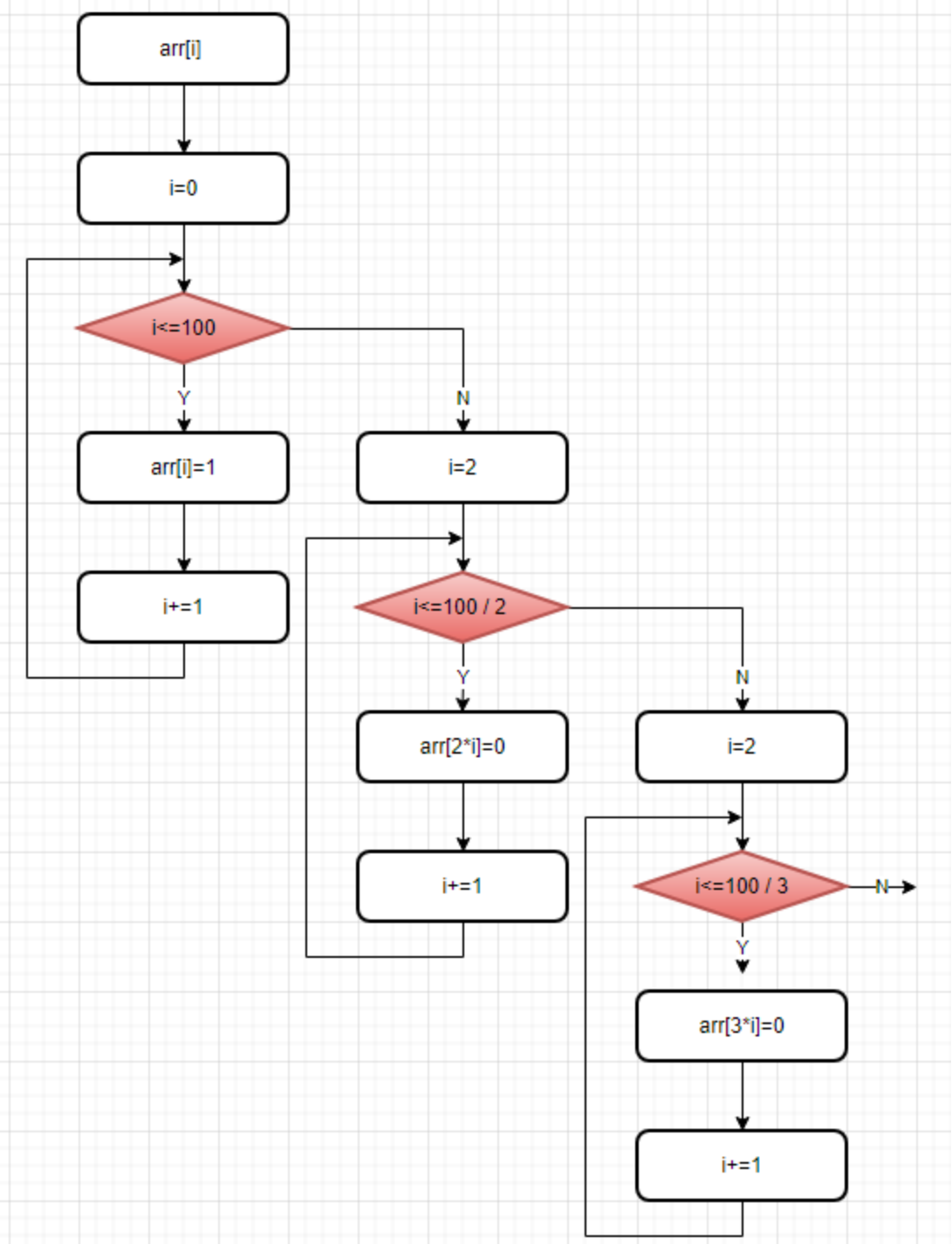
이번에는 수를 조금 늘려서 100이하의 소수를 모두 구하는 경우로 생각해 보자.

따라서 배열의 요소수는 101로 한다.

앞에서 생각한 것과 같이 첨자들을 수로 사용하고자 하기 때문이다.

첨자는 0부터 시작하기 때문에 요소의 수는 101개 필요하다.

모든 요소에는 초기값으로 1을 대입하여 초기화하자.



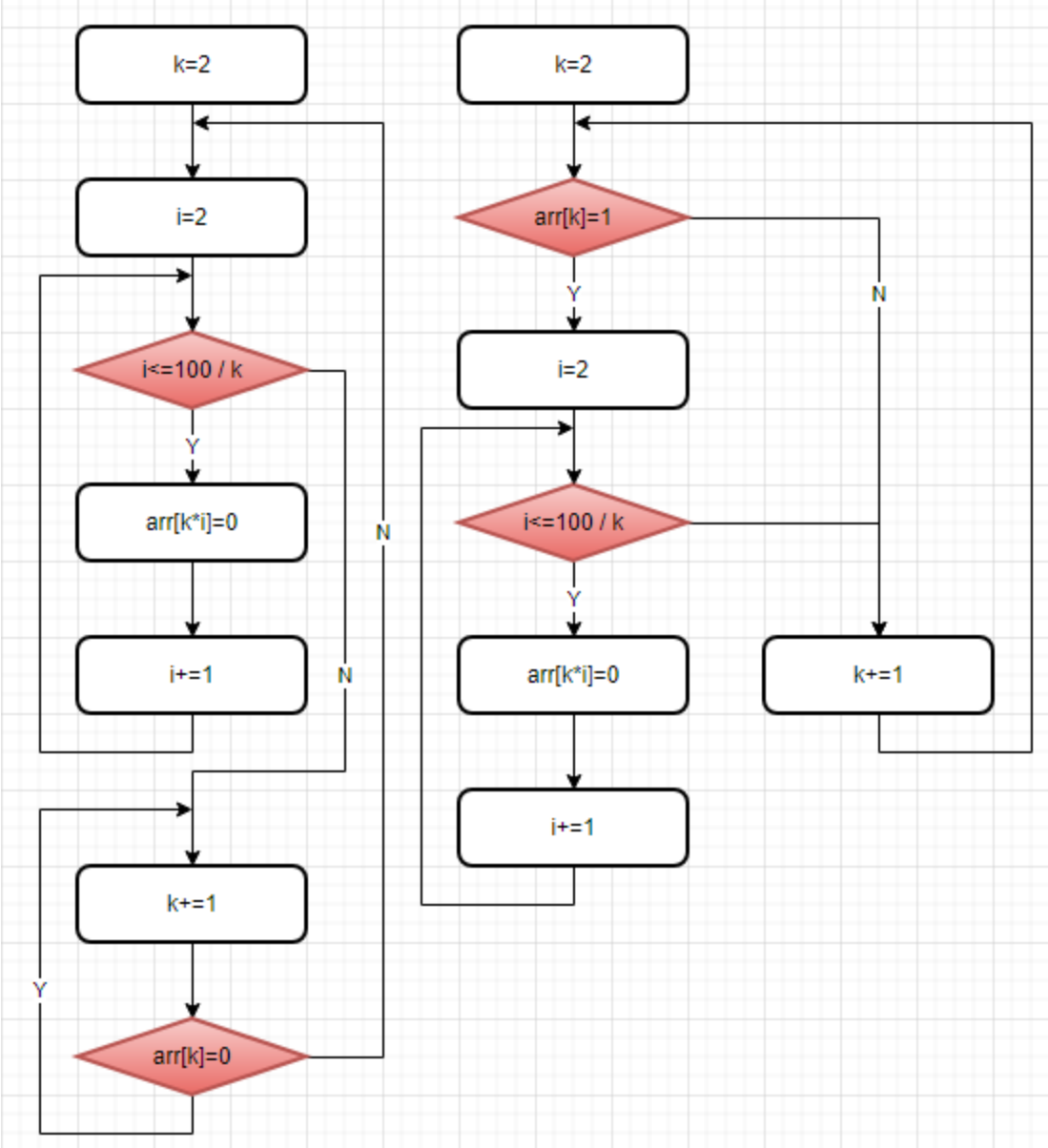
변수 k는 하나씩 증가한다. 2인 경우에는 2의 배수들을 전부 제거 즉 0으로 바꾸고 3인 경우에는 3의 배수들을 전부 제거하고 즉 0으로 바꾸고 계속 반복 시킨다. 그러나...4는 소수가 아니다.

3다음의 소수는 5이므로 5의 배수를 제거해야 한다. k값이 소수인지를 판단하는 처리를 추가해야한다.

2와 3의 배수 제거 처리가 끝난 단계에서의 배열을 생각해보면 이미 2의 배수와 3의 배수는 모두 제거된 즉 0 인 상태가 되었다. 이 시점에서 첨자가 소수가 아닌 요소에는 0이 대입되어 있다.

즉 4에는 이미 0이 대입된 상태이다.

k에 대입된 값이 소수인지의 여부에 따라 반복문으로의 진입을 판단해야 한다. 따라서 arr[k]=1 이면 소수 arr[k]=0이면 k는 소수가 아니라는 것이다.



arr[k] = 1 이 Yes인 경우 k는 소수이므로 k의 배수를 제거하는 반복 처리가 실행된다.

이와는 반대로 arr[k]=1 No인 경우는 k는 소수가 아니므로 k를 하나 늘려 다음 k가 소수인지를 다시 판명하게 된다.

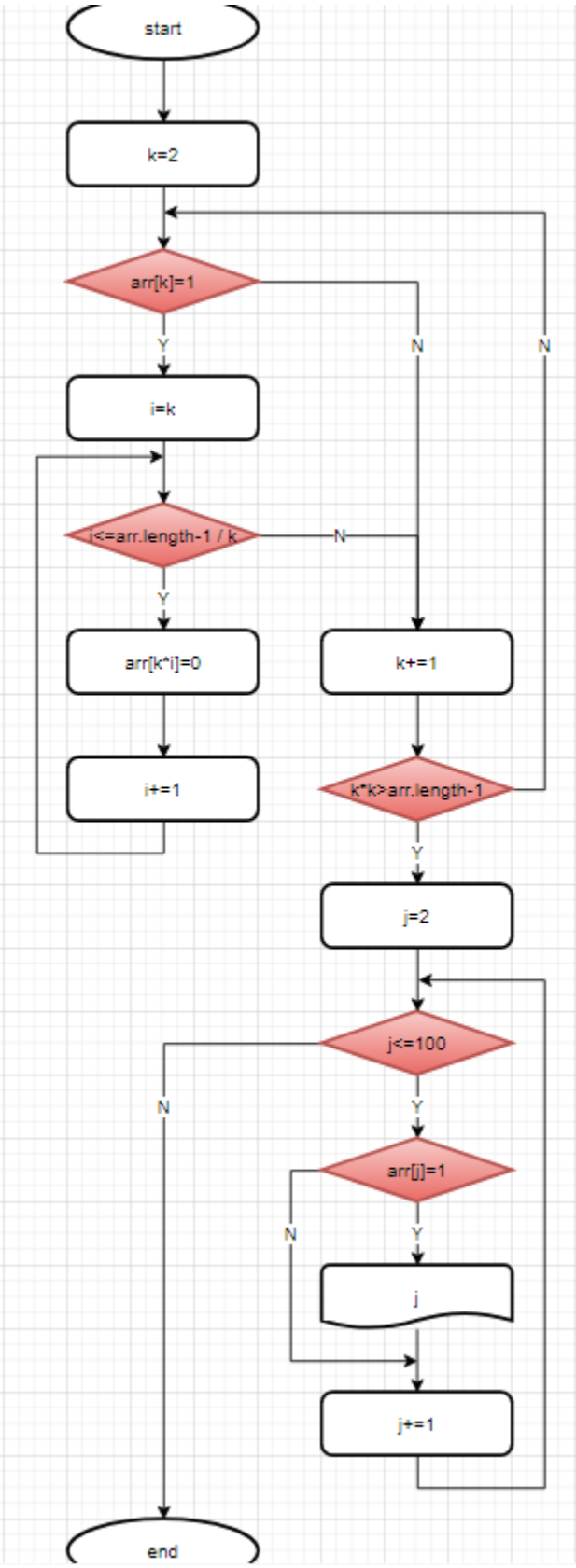
k값은 이제 2,3 으로 증가하고 k=4인 경우에는 arr[4]=0 이므로 배수를 제거하는 반복처리로 들어가지 않고 k만 하나 증가시키게 된다.

그 다음 k=5 인 경우 arr[5]=1이므로 5의 배수들을 모두 제거하고 … 계속 반복 된다.

따라서 무한루프에 빠지게 된다.

에라토스테네스의 체는 어떤 숫자의 ‘제곱근보다 작은 소수의 배수를 제거하면 남은 수가 소수이다’라는 이론이다. 따라서 이번 예제에서는 그 어떤 수가 100이므로 k는 100의 제곱근 즉, 10 이하가 된다.

따라서 k가 100의 제곱근 이하인가? 라는 판별식을 추가하여 반복을 마치게 된다.


start
k=2
arr[k]=1
Y
N
N
i=k
i<=arr.length-1 / k
N
Y
arr[k*i]=0
k+=1
i+=1
k*k>arr.length-1
Y
j=2
j<=100
N
Y
arr[j]=1
N
Y
j
j+=1
end