에라토스테네스의 체

Sieve of Eratosthenes

- 1부터 N까지 범위 안에 들어가는 모든 소수를 구하려면 에라토스테네스의 체를 사용 한다.
  ① 2부터 N까지 모든 수를 써 놓는다.
  ② 아직 지워지지 않은 수 중에서 가장 작은 수를 찾는다. (그 수는 소수에 해당한다.)
  ③ 그 수의 배수를 모두 지운다.
- 알고리즘 수행은 아래와 같다.

	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

에라토스테네스의 체

Sieve of Eratosthenes

- 2의 배수를 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	

- 3의 배수를 지운다.

	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			

에라토스테네스의 체

Sieve of Eratosthenes

- 5의 배수를 지운다.

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

- 7의 배수를 지운다.

	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			

에라토스테네스의 체

Sieve of Eratosthenes

- 1부터 N까지 모든 소수를 구하는 것이 목표이기 때문에, 구현할 때에는 for 루프를 N까지 반복한다.
- 루프를 반복할 때 i*i <= N인 때 까지 반복한다.

```
int N, M;
vi isPrime;

void eratosthenes() {
        isPrime = vi(N+10, 1);
        isPrime[0] = isPrime[1] = 0;
        for (int i = 2; i*i <= N; ++i) {
                if (isPrime[i] == 1) {
                        for (int j = i+i; j <= N; j+=i) {
                                isPrime[j] = 0;
                        }
                }
        }
}
```