Algorithm

- 소수

소수

- 소수란?
  - 1보다 큰 자연수 중 1과 자기 자신만을 약수로 가지는 수

  예시) 2, 3, 5, 7, 11, 13, 17, 19, 23....
  - 그렇다면 숫자 N이 소수가 되려면?

    → N은 2보다 크거나 같고, N-1보다 작거나 같은 자연수로 나누어 떨어지면 안된다!!

소수

```
public static boolean isPrime(int n) {
    if(n<2) {
        return false;
    }
    for(int i=2; i<=n-1; i++) {
        if(n%i ==0) {
            System.out.print(i+" ");
            return false;
        }
    }
    return true;
}
```

소수

```
public static boolean isPrime(int n) {
    if(n<2) {
        return false;
    }
    for(int i=2; i<=n-1; i++) {
        if(n%i ==0) {
            System.out.print(i+" ");
            return false;
        }
    }
    return true;
}
```

→ 시간 복잡도 : O(N)

이 코드로

M개의 숫자에 대해 소수 판별을 해야 한다면, 시간 복잡도는?

→ 최악의 경우 : M^2

소수

- 소수란?
  - 1보다 큰 자연수 중 1과 자기 자신만을 약수로 가지는 수
  - 숫자 N 이 소수가 아니라면, N = X*Y (X<=Y)
    - 36 : 1 2 3 4 6 9 12 18 36
    - 72 : 1 2 3 4 6 8 9 12 18 24 36 72
    - 1004 : 1 2 4 251 502 1004

: N의 약수 중 가장 큰 것은 N/2보다 작거나 같다.

(왜...그래요?? 숫자 N을 두 수의 곱으로 표현하면 X*Y 의 형태
이때, X의 값이 작을 수록 Y가 커지는데, 가능한 X중 1을 제외하면
가장 작은 값은 2이기 때문에 Y는 N/2를 넘지 않는다.)

소수

```
public static boolean isPrime2(int n) {
    if(n<2) {
        return false;
    }
    for(int i=2; i<=n/2; i++) {
        if(n%i ==0) {
            System.out.print(i+" ");
            return false;
        }
    }
    return true;
}
```

소수

```
public static boolean isPrime2(int n) {
    if(n<2) {
        return false;
    }
    for(int i=2; i<=n/2; i++) {
        if(n%i ==0) {
            System.out.print(i+" ");
            return false;
        }
    }
    return true;
}
```

→ 시간 복잡도 : O(N/2)

이 코드로

M개의 숫자에 대해 소수 판별을 해야 한다면, 시간 복잡도는?

→ 최악의 경우 : M*M/2

소수

- 소수란?
  - 1보다 큰 자연수 중 1과 자기 자신만을 약수로 가지는 수
  - 숫자 N 이 소수가 아니라면, $\sqrt{N}$을 기준으로 대칭 구조를 가지고 있음
  - 36 : 1 2 3 4 6 9 12 18 36 $\sqrt{36} = 6$
  - 72 : 1 2 3 4 6 8 9 12 18 24 36 72 $\sqrt{72} = 8.485$

  → 2 ~ $\sqrt{N}$보다 작거나 같은 자연수로 나누어 떨어지면 N은 소수가 아니다!!

소수

```java
public static boolean isPrime3(int n) {
    if(n<2) {
        return false;
    }
    for(int i=2; i*i<=n; i++) {
        if(n%i ==0) {
            System.out.print(i+" ");
            return false;
        }
    }
    return true;
}
```

컴퓨터는 실수를 근사값으로 나타내기 때문에 유의!

소수

```
public static boolean isPrime3(int n) {
    if(n<2) {
        return false;
    }
    for(int i=2; i*i<=n; i++) {
        if(n%i ==0) {
            System.out.print(i+" ");
            return false;
        }
    }
    return true;
}
```

→ 시간 복잡도 : O($\sqrt{N}$)

이 코드로

M개의 숫자에 대해 소수 판별을 해야 한다면, 시간 복잡도는?

→ 최악의 경우 : M* $\sqrt{M}$

에라토스테네스의 체

- **특정 범위(1부터 N까지) 안의 모든 소수를 구할 때 사용**
  - 예시) 1부터 1,000,000 까지 모든 소수를 구하는데 걸리는 시간 복잡도는?
    - 각 수에 대해 소수인지 아닌지 판별 : 각 수마다 $O(\sqrt{N})$ 시간 복잡도
    - 1,000,000*1,000 = > 10억

1. 2부터 N까지 모든 수를 써 놓는다
2. 아직 지워지지 않은 수 중에서 가장 작은 수를 찾는다.
3. 그런데 그 수는 소수이다.
4. 그 수의 배수를 모두 지운다.(그 수의 제곱이 N보다 크면 더 이상 지울 수가 없다)

에라토스테네스의 체

1. 2부터 N까지 모든 수를 써 놓는다

	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

2. 아직 지워지지 않은 수 중에서 가장 작은 수를 찾는다.

→그런데 그 수는 소수이다.

→ 2는 소수이고, 가장 작은 수이다.

	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

3. 그 수의 배수를 모두 지운다.(그 수의 제곱이 N보다 크면 더 이상 지울 수가 없다)

→ 2의 배수를 모두 지운다.

	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

3. 그 수의 배수를 모두 지운다.(그 수의 제곱이 N보다 크면 더 이상 지울 수가 없다)

→ 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	

에라토스테네스의 체

2. 아직 지워지지 않은 수 중에서 가장 작은 수를 찾는다.

→그런데 그 수는 소수이다.

→ 3는 소수이고, 가장 작은 수이다.

	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	

에라토스테네스의 체

3. 그 수의 배수를 모두 지운다.(그 수의 제곱이 N보다 크면 더 이상 지울 수가 없다)

→ 3의 배수를 모두 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

3. 그 수의 배수를 모두 지운다.(그 수의 제곱이 N보다 크면 더 이상 지울 수가 없다)

→ 5의 배수를 모두 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			

에라토스테네스의 체

3. 그 수의 배수를 모두 지운다.(그 수의 제곱이 N보다 크면 더 이상 지울 수가 없다)

→ 7의 배수를 모두 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

3. 그 수의 배수를 모두 지운다.(그 수의 제곱이 N보다 크면 더 이상 지울 수가 없다)

→11의 제곱은? 121로 N보다 커지므로 더 이상 지울 수 없음!

→수행 종료

	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

참고용 문제

- 소수 구하기 : https://www.acmicpc.net/problem/1929

```
public static void isPrime(int m) {
    check[0] = check[1] = true;
    for (int i = 2; i * i <= m; i++) {
        if (check[i] == true) {
            continue;
        }
        for (int j = i + i; j <= m; j += i) {
            check[j] = true;
        }
    }
}
```