



고급 알고리즘

소수(Prime Number)

- 소수란 1보다 큰 자연수 중에서 1과 자기 자신을 제외한 자연수로는 나누어 떨어지지 않는 자연수
 - 6은 1, 2, 3, 6으로 나누어 떨어지므로 소수가 아닙니다.
 - 7은 1과 7을 제외하고는 나누어 떨어지지 않으므로 소수입니다.
- 코딩 테스트에서는 어떠한 자연수가 소수인지 아닌지 판별해야 하는 문제가 자주 출제됩니다.

소수의 판별 : 기본적인 알고리즘(Python)

```
# 소수 판별 함수(2이상의 자연수에 대하여)
def is_prime_number(x):
    # 2부터 (x - 1)까지의 모든 수를 확인하며
    for i in range(2, x):
        # x가 해당 수로 나누어떨어진다면
        if x % i == 0:
            return False # 소수가 아님
    return True # 소수임

print(is_prime_number(4))
print(is_prime_number(7))
```

실행 결과

False
True

소수의 판별: 기본적인 알고리즘(C++)

```
#include <bits/stdc++.h>

using namespace std;

// 소수 판별 함수(2이상의 자연수에 대하여)
bool isPrimeNumber(int x) {
    // 2부터 (x - 1)까지의 모든 수를 확인하며
    for (int i = 2; i < x; i++) {
        // x가 해당 수로 나누어떨어진다면
        if (x % i == 0) {
            return false; // 소수가 아님
        }
    }
    return true; // 소수임
}

int main() {
    cout << isPrimeNumber(4) << '\n';
    cout << isPrimeNumber(7) << '\n';
}
```

실행 결과

0
1

소수의 판별: 기본적인 알고리즘(Java)

```
class Main {  
    // 소수 판별 함수(2이상의 자연수에 대하여)  
    public static boolean isPrimeNumber(int x) {  
        // 2부터 (x - 1)까지의 모든 수를 확인하며  
        for (int i = 2; i < x; i++) {  
            // x가 해당 수로 나누어떨어진다면  
            if (x % i == 0) {  
                return false; // 소수가 아님  
            }  
        }  
        return true; // 소수임  
    }  
  
    public static void main(String[] args) {  
        System.out.println(isPrimeNumber(4));  
        System.out.println(isPrimeNumber(7));  
    }  
}
```

실행 결과

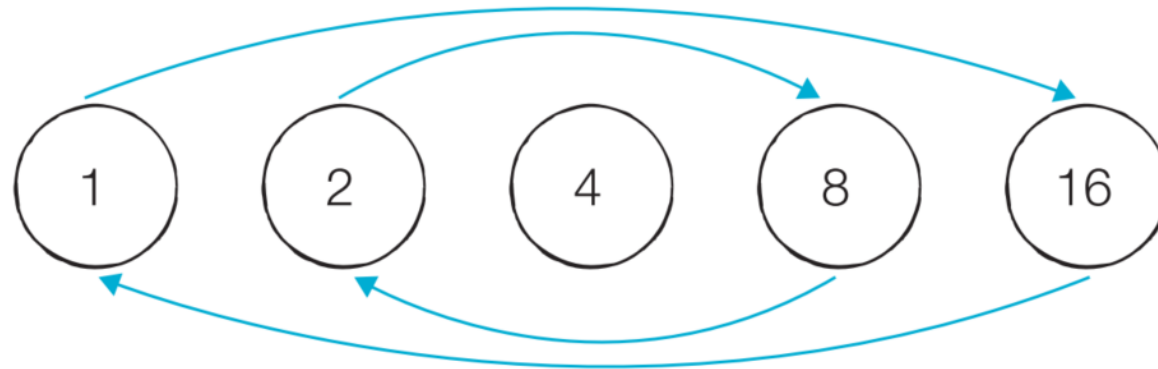
False
True

소수의 판별: 기본적인 알고리즘 성능 분석

- 2부터 $x-1$ 까지의 모든 자연수에 대하여 연산을 수행해야 합니다.
- 모든 수를 하나씩 확인한다는 점에서 시간 복잡도는 $O(X)$ 입니다.

약수의 성질

- 모든 약수가 가운데 약수를 기준으로 곱셈 연산에 대해 대칭을 이루는 것을 알 수 있습니다.
 - 예를 들어 16의 약수는 1, 2, 4, 8, 16입니다.
 - 이때 $2 \times 8 = 16$ 은 $8 \times 2 = 16$ 과 대칭입니다.
- 따라서 우리는 특정한 자연수의 모든 약수를 찾을 때 가운데 약수(제곱근)까지만 확인하면 됩니다.
 - 예를 들어 16이 2로 나누어떨어진다는 것은 8로도 나누어떨어진다는 것을 의미합니다.



소수의 판별: 개선된 알고리즘(Python)

```
import math

# 소수 판별 함수 (2이상의 자연수에 대하여)
def is_prime_number(x):
    # 2부터 x의 제곱근까지의 모든 수를 확인하며
    for i in range(2, int(math.sqrt(x)) + 1):
        # x가 해당 수로 나누어떨어진다면
        if x % i == 0:
            return False # 소수가 아님
    return True # 소수임

print(is_prime_number(4))
print(is_prime_number(7))
```

실행 결과

False
True

소수의 판별: 개선된 알고리즘(C++)

```
#include <bits/stdc++.h>

using namespace std;

// 소수 판별 함수 (2이상의 자연수에 대하여)
bool isPrimeNumber(int x) {
    // 2부터 x의 제곱근까지의 모든 수를 확인하며
    for (int i = 2; i <= (int) sqrt(x); i++) {
        // x가 해당 수로 나누어떨어진다면
        if (x % i == 0) {
            return false; // 소수가 아님
        }
    }
    return true; // 소수임
}

int main() {
    cout << isPrimeNumber(4) << '\n';
    cout << isPrimeNumber(7) << '\n';
}
```

실행 결과

0
1

소수의 판별:개선된 알고리즘(Java)

```
import java.util.*;

class Main {
    // 소수 판별 함수(2이상의 자연수에 대하여)
    public static boolean isPrimeNumber(int x) {
        // 2부터 x의 제곱근까지의 모든 수를 확인하며
        for (int i = 2; i <= Math.sqrt(x); i++) {
            // x가 해당 수로 나누어떨어진다면
            if (x % i == 0) {
                return false; // 소수가 아님
            }
        }
        return true; // 소수임
    }

    public static void main(String[] args) {
        System.out.println(isPrimeNumber(4));
        System.out.println(isPrimeNumber(7));
    }
}
```

실행 결과

False
True

소수의 판별: 개선된 알고리즘 성능 분석

- 2부터 x 의 제곱근(소수점 이하 무시)까지의 모든 자연수에 대하여 연산을 수행해야 합니다.
 - 시간 복잡도는 $O(N^{\frac{1}{2}})$ 입니다.

소수 구하기

소수(prime number)는 자신보다 작은 2개의 자연수를 곱해 만들 수 없는 1보다 큰 자연수를 말한다. 같은 의미로 1과 자기 자신 외에 약수가 존재하지 않는 수를 말한다.

소수를 구하는 대표적인 판별법으로는 에라토스테네스의 체를 들 수 있다.

에라토스테네스의 체 원리

- ① 구하고자 하는 소수의 범위만큼 1차원 배열을 생성합니다.
- ② 2부터 시작하고 현재 숫자가 지워지지 않을 때는 현재 선택된 숫자의 배수에 해당하는 수를 배열에서 끝까지 탐색하면서 지웁니다. 이때 처음으로 선택된 숫자는 지우지 않습니다.
- ③ 배열의 끝까지 ②를 반복한 후 배열에서 남아 있는 모든 수를 출력합니다.

소수 구하기

에라토스테네스의 체의 원리 이해하기

1부터 30까지의 수 중 소수를 구하는 예시를 보면서 에라토스테네스의 체의 원리를 알아보자.

- 1 주어진 범위까지 배열을 생성한다. 1은 소수가 아니므로 삭제하고, 2부터 시작한다.



소수 구하기

2 선택한 수의 배수를 모두 삭제한다. 현재의 경우 2의 배수를 모두 삭제했다.

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

3 다음 지워지지 않은 수를 선택한다. 즉, 3을 선택하고 선택한 수의 모든 배수를 삭제한다.

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

소수 구하기

4 앞의 과정을 배열 끝까지 반복한다.

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

5 삭제되지 않은 수를 모두 출력한다.

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

소수 구하기

빈출

소수 구하기

시간 제한 2초 | 난이도 실버 III | 백준 온라인 저지 1929번

M 이상 N 이하의 소수를 모두 출력하는 프로그램을 작성하시오.

↓ 입력

1번째 줄에 자연수 M과 N이 빈칸을 사이에 두고 주어진다($1 \leq M \leq N \leq 1,000,000$). M 이상 N 이하의 소수가 1개 이상 있는 입력만 주어진다.

↑ 출력

1줄에 1개씩, 증가하는 순서대로 소수를 출력한다.

소수 구하기

예제 입력 1

3 16

예제 출력 1

3
5
7
11
13

소수 구하기

01단계 문제 분석하기

1. N 의 최대 범위가 1,000,000이므로 일반적인 소수 구하기 방식으로 문제를 풀면 시간 초과가 발생한다.
2. 에라토스테네스 방법으로 문제를 해결한다.

02단계 손으로 풀어 보기

- 1 크기가 $N + 1$ 인 배열을 선언한 후 값은 각각의 인덱스 값으로 채운다.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----

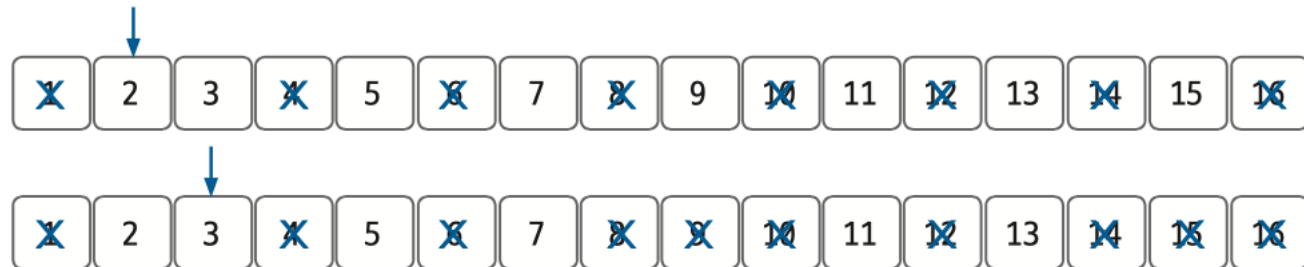
소수 구하기

2 1은 소수가 아니므로 삭제한다.



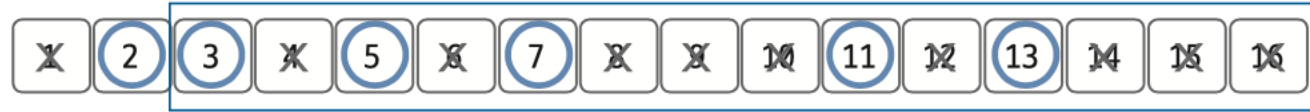
3 2부터 N의 제곱근까지 값을 탐색한다.

값이 인덱스 값이면 그대로 두고, 그 값의 배수를 탐색해 0으로 변경한다.



소수 구하기

4 배열에 남아 있는 수 중 M 이상 N 이하의 수를 모두 출력한다.



소수 구하기

코드 구현하기

K진수에서 소수의 개수 구하기

(출처) tech.kakao.com

2022 카카오 공채 코딩 테스트 문제

양의 정수 n 이 주어집니다. 이 숫자를 k 진수로 바꿨을 때, 변환된 수 안에 아래 조건에 맞는 소수(Prime number)가 몇 개인지 알아보려 합니다.

- oPo처럼 소수 양쪽에 o이 있는 경우
- Po처럼 소수 오른쪽에만 o이 있고 왼쪽에는 아무것도 없는 경우
- oP처럼 소수 왼쪽에만 o이 있고 오른쪽에는 아무것도 없는 경우
- P처럼 소수 양쪽에 아무것도 없는 경우

단, P는 각 자릿수에 o를 포함하지 않는 소수입니다.

예를 들어, 101은 P가 될 수 없습니다.

예를 들어, 437674를 3진수로 바꾸면 211020101011입니다. 여기서 찾을 수 있는 조건에 맞는 소수는 왼쪽부터 순서대로 211, 2, 11이 있으며, 총 3개입니다. 211은 Po 형태에서 찾을 수 있으며, 2는 oPo에서, 11은 oP에서 찾을 수 있습니다.

정수 n 과 k 가 매개변수로 주어집니다. n 을 k 진수로 바꿨을 때, 변환된 수 안에서 찾을 수 있는 위 조건에 맞는 소수의 개수를 return 하도록 solution 함수를 완성해 주세요.

제한사항

$1 \leq n \leq 1,000,000$

$3 \leq k \leq 10$

k진수에서 소수의 개수 구하기

(출처) tech.kakao.com

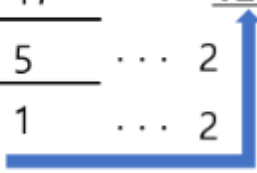
비밀지도 - 2018 카카오 공채 코딩 테스트 문제

입출력 예	<u>입력</u> 437674 3	<u>출력</u> 3
	<u>입력</u> 110011 10	<u>출력</u> 2
	110011을 10진수로 바꾸면 110011입니다. 여기서 찾을수 있는 조건에 맞는 소수는 11, 11 2개입니다. 이와 같이 중복되는 소수를 발견하더라도 모두 따로 세어야 합니다.	n=437674, k=3 인 경 우 k 진 수 는 211020101011 입 니 다 . 조건에 맞는 소수는 211 2 11 3개 입니다.

K진수에서 소수의 개수 구하기

- 이 문제는 진법 변환후에 변환된 숫자를 0으로 기준으로 파싱하고 파싱 한 숫자를 소수 판별해 해결하는 문제
- 제한 사항을 살펴보면 n이 1부터 1,000,000까지이고 K는 3부터 10이므로 1,000,000을 3진수로 바꾸면 1,212,210,202,001입니다.
- K진수로의 변환

ex) $k = 3$

$$\begin{array}{r} 3 \overline{) 17} \quad \quad \quad \underline{122} \\ 3 \overline{) 5} \quad \dots 2 \\ \quad 1 \quad \dots 2 \end{array}$$


K진수에서 소수의 개수 구하기 (Python)



K진수에서 소수의 개수 구하기 (Java)

