알고리즘 문제 풀이

에꼴캡스톤디자인

2021763040 성현욱

2025.10.10

수 찾기 : 1920

문제

N개의 정수 A[1], A[2], …, A[N]이 주어져 있을 때, 이 안에 X라는 정수가 존재하는지 알아내는 프로그램을 작성하시오.

입력

첫째 줄에 자연수 N(1 ≤ N ≤ 100,000)이 주어진다.
다음 줄에는 N개의 정수 A[1], A[2], ..., A[N]이 주어진다.
다음 줄에는 M(1 ≤ M ≤ 100,000)이 주어진다.
다음 줄에는 M개의 수들이 주어지는데, 이 수들이 A안에 존재하는지 알아내면 된다.
모든 정수의 범위는 -2^{31} 보다 크거나 같고 2^{31}보다 작다.

출력

M개의 줄에 답을 출력한다. 존재하면 1을, 존재하지 않으면 0을 출력한다.

예제 입력 1 복사

```
5
4 1 5 2 3
5
1 3 7 9 5
```

예제 출력 1 복사

```
1
1
0
0
1
```

```java
static int  binarySearch(int[] arr, int target, int start, int end) {
    if (start > end)
        return 0;

    int mid = start + (end - start) / 2;

    if (arr[mid] == target)
        return 1;
    else if (arr[mid] < target)
        return binarySearch(undefined: arr, undefined: target, start: mid + 1, undefined: end);
    else
        return binarySearch(undefined: arr, undefined: target, undefined: start, end: mid - 1);
}

Run | Debug
public static void main(String[] args) {
    Scanner in = new Scanner(source: System.in);

    int n = in.nextInt();
    int[] a = new int[n];

    for (int i = 0; i < n; i++)
        a[i] = in.nextInt();
    Arrays.sort(undefined: a);

    int m = in.nextInt();
    int[] x = new int[m];
    for (int i = 0; i < m; i++)
        x[i] = in.nextInt();

    for (int target : x)
        System.out.println(x: binarySearch(arr: a, undefined: target, start: start: 0, end: a.length - 1));
}
```

```javascript
const fs = require("fs");
const input = fs.readFileSync("/dev/stdin").toString().trim().split( separator: "\n");

function binarySearch(arr, target, left, right) {
  if (left > right)
    return "0";

  const mid = Math.floor( x: left + (right - left) / 2);
  if (arr[mid] < target)
    return binarySearch(arr, target,  left: mid + 1, right);
  else if (arr[mid] > target)
    return binarySearch(arr, target, left,  right: mid - 1);
  else
    return "1";
}

const a = input[1].split( separator: " ").map( callbackfn: Number);
const x = input[3].split( separator: " ").map( callbackfn: Number);

a.sort( compareFn: (a, b) => a - b);

for (const i of x)
  console.log( message: binarySearch(a, i, 0, a.length - 1));
```

```python
import sys
input = sys.stdin.readline

def binarySearch(arr, target, left, right):
    if left > right:
        return 0

    mid = left + (right - left) // 2
    if arr[mid] > target:
        return binarySearch(arr, target, left, right= mid - 1)
    elif arr[mid] < target:
        return binarySearch(arr, target, left= mid + 1, right)
    else:
        return 1

_ = input()
a = list(iterable= map(func= int, input().split()))
a.sort()

_ = input()
x = list(iterable= map(func= int, input().split()))

for i in x:
    print(values[0]= binarySearch(arr= a, target= i, left= 0, right= len(obj= a) - 1))
```

채점 결과

수 찾기 : 1920

hyunook02	1920	맞았습니다!!	49136 KB	536 ms	Python 3 / 수정
hyunook02	1920	맞았습니다!!	45160 KB	3600 ms	node.js / 수정
hyunook02	1920	맞았습니다!!	186212 KB	1580 ms	Java 11 / 수정

기타 레슨 : 2343

문제

동협이는 자신의 기타 강의 동영상을 블루레이로 만들어 판매하려고 한다. 블루레이에는 총 N개의 강의가 들어가는데, 블루레이를 녹화할 때 강의의 순서가 바뀌면 안 된다.

동협이는 이 강의가 얼마나 팔릴지 모르기 때문에, 블루레이의 개수를 가급적 줄이려고 한다. 오랜 고민 끝에 동협이는 M개의 블루레이에 모든 기타 강의 영상을 녹화하기로 했다. 이때 블루레이의 크기(녹화 가능한 길이)를 최소로 하려고 한다. 단, M개의 블루레이는 모두 같은 크기이어야 한다.

동협이의 각 강의의 길이가 분 단위(자연수)로 주어진다. 이때, 가능한 블루레이의 크기 중 최소를 구하는 프로그램을 작성하시오.

★ 입력

첫째 줄에 강의의 수 N ($1 \le N \le 100,000$)과 M ($1 \le M \le N$)이 주어진다. 다음 줄에는 동협이의 기타 강의의 길이가 강의 순서대로 분 단위로 주어진다. 이때, 가능한 블루레이의 크기 중 최소를 구하는 프로그램을 작성하시오.

★ 출력

첫째 줄에 가능한 블루레이 크기 중 최소를 출력한다.

기타 레슨 : 2343

★ 예제

예제 입력 1 복사

```
9 3
1 2 3 4 5 6 7 8 9
```

예제 출력 1 복사

```
17
```

```java
public static void main(String[] args) {
    Scanner in = new Scanner(source: System.in);
    int n = in.nextInt(), m = in.nextInt();
    int[] lessons = new int[n];
    int min = 0, max = 0;

    for (int i = 0; i < n; i++) {
        lessons[i] = in.nextInt();
        min = Math.max(a: min, b: lessons[i]);
        max += lessons[i];
    }

    while (min <= max) {
        int mid = min + (max - min) / 2;
        int sum = 0, count = 0;

        for (int i: lessons) {
            if (sum + i > mid) {
                count++;
                sum = 0;
            }
            sum += i;
        }

        if (sum > 0)
            count++;

        if (count > m)
            min = mid + 1;
        else
            max = mid - 1;
    }

    System.out.println(x: min);
}
```

```
const input = require('fs')
    .readFileSync('/dev/stdin')
    .toString()
    .trim()
    .split( separator: '\n')

const [n, m] = input[0].split( separator: ' ').map( callbackfn: Number);
const lessons = input[1].split( separator: ' ').map( callbackfn: Number);
let min = Math.max(...lessons);
let max = lessons.reduce( callbackfn: (acc, curr) => acc + curr,  currentValue: 0);

while (min <= max) {
  const mid = Math.floor((min + max) / 2);
  let sum = 0, count = 0;

  for (const i of lessons) {
    if (sum + i > mid) {
      count++;
      sum = 0;
    }
    sum += i;
  }
  if (sum > 0)
    count++;

    count > m ? (min = mid + 1) : (max = mid - 1);
}

console.log( message: min);
```

Python


```python
import sys
input = sys.stdin.readline

n, m = map(func= int, input().split())
lessons = list(iterable= map(func= int, input().split()))

min, max = max(lessons), sum(lessons)
while min <= max:
    mid = (min + max) // 2

    sum, count = 0, 0
    for i in lessons:
        if sum + i > mid:
            count += 1
            sum = 0
        sum += i
    if sum != 0:
        count += 1

    if count > m:
        min = mid + 1
    else:
        max = mid - 1

print(values[0]= min)
```

채점 결과

수 찾기 : 1920

hyunook02	2343	맞았습니다!!	108524 KB	784 ms	Java 11 / 수정
hyunook02	2343	맞았습니다!!	42168 KB	484 ms	Python 3 / 수정
hyunook02	2343	맞았습니다!!	26332 KB	228 ms	node.js / 수정

동전 0 : 11047

문제

준규가 가지고 있는 동전은 총 N종류이고, 각각의 동전을 매우 많이 가지고 있다.

동전을 적절히 사용해서 그 가치의 합을 K로 만들려고 한다. 이때 필요한 동전 개수의 최솟값을 구하는 프로그램을 작성하시오.

입력

첫째 줄에 N과 K가 주어진다. ($1 \le N \le 10$, $1 \le K \le 100,000,000$)

둘째 줄부터 N개의 동전의 가치 A_i가 오름차순으로 주어진다. ($1 \le A_i \le 100,000$, $A_1 = 1$, $i \ge 2$인 경우에 A_i는 A_i-1의 배수)

출력

첫째 줄에 K원을 만드는데 필요한 동전 개수의 최솟값을 출력한다.

동전 0 : 11047

★ 예제

예제 입력 1 복사

```
10 4200
1
5
10
50
100
500
1000
5000
10000
50000
```

예제 출력 1 복사

```
6
```

예제 입력 2 복사

```
10 4790
1
5
10
50
100
500
1000
5000
10000
50000
```

예제 출력 2 복사

```
12
```

```java
public static void main(String[] args) {
    Scanner in = new Scanner(source: System.in);
    int n = in.nextInt(), k = in.nextInt();
    int[] coins = new int[n];
    for (int i = 0; i < n; i++)
        coins[i] = in.nextInt();

    int count = 0;
    for (int i = n - 1; i >= 0; i--) {
        if (k >= coins[i]) {
            count += k / coins[i];
            k %= coins[i];
        }
    }

    System.out.println(x: count);
}
```

```
let input = require('fs')
    .readFileSync('/dev/stdin')
    .toString()
    .trim()
    .split( separator: '\n')

let [n, k] = input.shift().split( separator: ' ').map( callbackfn: Number);
const coins = input.reverse().map( callbackfn: Number);

let count = 0;

for (let i of coins) {
  if (k >= i) {
    count += Math.floor( x: k / i);
    k %= i;
  }
}

console.log( message: count);
```

```python
n, k = map(func= int, input().split())
coins = listiterable= (int(x= input()) for _ in range(stop= n))
coins.reverse()

count = 0
for i in coins:
    if k >= i:
        count += k // i
        k %= i

print(values[0]= count)
```

채점 결과

동전 0 : 11047

hyunook02	11047	맞았습니다!!	32544 KB	36 ms	Python 3 / 수정
hyunook02	11047	맞았습니다!!	17728 KB	172 ms	Java 11 / 수정
hyunook02	11047	맞았습니다!!	9596 KB	92 ms	node.js / 수정

카드 정렬하기 : 1715

문제

정렬된 두 묶음의 숫자 카드가 있다고 하자. 각 묶음의 카드의 수를 A, B라 하면 보통 두 묶음을 합쳐서 하나로 만드는 데에는 A+B 번의 비교를 해야 한다. 이를테면, 20장의 숫자 카드 묶음과 30장의 숫자 카드 묶음을 합치려면 50번의 비교가 필요하다.

매우 많은 숫자 카드 묶음이 책상 위에 놓여 있다. 이들을 두 묶음씩 골라 서로 합쳐나간다면, 고르는 순서에 따라서 비교 횟수가 매우 달라진다. 예를 들어 10장, 20장, 40장의 묶음이 있다면 10장과 20장을 합친 뒤, 합친 30장 묶음과 40장을 합친다면 (10 + 20) + (30 + 40) = 100번의 비교가 필요하다. 그러나 10장과 40장을 합친 뒤, 합친 50장 묶음과 20장을 합친다면 (10 + 40) + (50 + 20) = 120 번의 비교가 필요하므로 덜 효율적인 방법이다.

N개의 숫자 카드 묶음의 각각의 크기가 주어질 때, 최소한 몇 번의 비교가 필요한지를 구하는 프로그램을 작성하시오.

입력

첫째 줄에 N이 주어진다. ($1 \le N \le 100,000$) 이어서 N개의 줄에 걸쳐 숫자 카드 묶음의 각각의 크기가 주어진다. 숫자 카드 묶음의 크기는 1,000보다 작거나 같은 양의 정수이다.

출력

첫째 줄에 최소 비교 횟수를 출력한다.

카드 정렬하기 : 1715

★ 예제

예제 입력 1 복사

```
3
10
20
40
```

예제 출력 1 복사

```
100
```

```java
public static void main(String[] args) {
    Scanner in = new Scanner(source: System.in);
    int n = in.nextInt();
    PriorityQueue<Integer> q = new PriorityQueue<>();
    for (int i = 0; i < n; i++)
        q.add(e: in.nextInt());

    int sum = 0;
    while (q.size() > 1) {
        int i = q.remove() + q.remove();
        sum += i;
        q.add(e: i);
    }

    System.out.println(x: sum);
}
```

```
class PriorityQueue {
  constructor() {
    this.size = 0;
    this.heap = [null];
  }

  put(data) {
    this.heap.push( items[0]: data);

    let curIndex = this.heap.length - 1;
    let parIndex = Math.floor( x: curIndex / 2);
    while (parIndex !== 0 && this.heap[parIndex] > data) {
      const tmp = this.heap[parIndex];
      this.heap[parIndex] = this.heap[curIndex];
      this.heap[curIndex] = tmp;
      curIndex = parIndex;
      parIndex = Math.floor( x: curIndex / 2);
    }

    this.size++;
  }
```

```
  get() {
    if (this.size === 0)
      return undefined;

    const returnValue = this.heap[1];
    if (this.size === 1) {
      this.heap.pop();
      this.size--;
      return returnValue;
    }

    this.heap[1] = this.heap.pop();
    this.size--;

    let currentIndex = 1;
    while (true) {
      const leftChildIndex = currentIndex * 2;
      const rightChildIndex = currentIndex * 2 + 1;
      let smallestChildIndex = leftChildIndex;

      if (leftChildIndex > this.size)
        break;
      if (
          rightChildIndex <= this.size &&
          this.heap[rightChildIndex] < this.heap[leftChildIndex]
      )
        smallestChildIndex = rightChildIndex;

      if (this.heap[currentIndex] <= this.heap[smallestChildIndex])
        break;

      [this.heap[currentIndex], this.heap[smallestChildIndex]]= [this.heap[smallestChildIndex], this.heap[currentIndex]];
      currentIndex = smallestChildIndex;
    }

    return returnValue;
  }
}
```

JavaScript


```
let cards = require('fs')
    .readFileSync('/dev/stdin')
    .toString()
    .trim()
    .split( separator: '\n')
    .map( callbackfn: Number);

const n = cards.shift();
const q = new PriorityQueue();
let sum = 0;

for (let i of cards)
  q.put( data: i);

while (q.size > 1) {
  const i = q.get() + q.get();
  sum += i;
  q.put( data: i);
}

console.log( message: sum);
```

Python


```python
from queue import PriorityQueue

n = int(x= input())
q = PriorityQueue()
for _ in range(stop= n):
    q.put(item= int(x= input()))

sum = 0
while q.qsize() > 1:
    i = q.get() + q.get()
    sum += i
    q.put(item= i)

print(values[0]= sum)
```

채점 결과

카드 정렬하기 : 1715

hyunook02	1715	맞았습니다!!	114612 KB	924 ms	Java 11 / 수정
hyunook02	1715	맞았습니다!!	40188 KB	3020 ms	Python 3 / 수정
hyunook02	1715	맞았습니다!!	19568 KB	228 ms	node.js / 수정

회의실 배정 : 1931

문제

한 개의 회의실이 있는데 이를 사용하고자 하는 N개의 회의에 대하여 회의실 사용표를 만들려고 한다. 각 회의 I에 대해 시작 시간과 끝나는 시간이 주어져 있고, 각 회의가 겹치지 않게 하면서 회의실을 사용할 수 있는 회의의 최대 개수를 찾아보자. 단, 회의는 한번 시작하면 중간에 중단될 수 없으며 한 회의가 끝나는 것과 동시에 다음 회의가 시작될 수 있다. 회의의 시작 시간과 끝나는 시간이 같을 수도 있다. 이 경우에는 시작하자마자 끝나는 것으로 생각하면 된다.

입력

첫째 줄에 회의의 수 N(1 ≤ N ≤ 100,000)이 주어진다. 둘째 줄부터 N+1 줄까지 각 회의의 정보가 주어지는데 이것은 공백을 사이에 두고 회의의 시작시간과 끝나는 시간이 주어진다. 시작 시간과 끝나는 시간은 231-1보다 작거나 같은 자연수 또는 0이다.

출력

첫째 줄에 최대 사용할 수 있는 회의의 최대 개수를 출력한다.

회의실 배정 : 1931

★ 예제

예제 입력 1 복사

```
11
1 4
3 5
0 6
5 7
3 8
5 9
6 10
8 11
8 12
2 13
12 14
```

예제 출력 1 복사

```
4
```

Java


```java
public static void main(String[] args) {
    Scanner in = new Scanner(source: System.in);
    int n = in.nextInt();
    int[][] a = new int[n][2];
    for (int i = 0; i < n; i++) {
        a[i][0] = in.nextInt();
        a[i][1] = in.nextInt();
    }

    Arrays.sort(undefined: a, c: new Comparator<int[]>() {
        @Override
        public int compare(int[] o1, int[] o2) {
            if (o1[1] == o2[1])
                return o1[0] - o2[0];
            return o1[1] - o2[1];
        }
    });

    int count = 0, end = -1;
    for (int i = 0; i < n; i++) {
        if (a[i][0] >= end) {
            end = a[i][1];
            count++;
        }
    }

    System.out.println(x: count);
}
```

JavaScript


```
let a = require('fs')
    .readFileSync('/dev/stdin')
    .toString()
    .trim()
    .split( separator: '\n');

const n = parseInt( string: a.shift());
for (let i = 0; i < n; i++)
  a[i] = a[i].split( separator: ' ').map( callbackfn: Number);

a.sort( compareFn: (s, e) => {
  if (s[1] === e[1])
    return s[0] - e[0];
  return s[1] - e[1];
});

let count = 0, end = -1;
for (let i = 0; i < n; i++) {
  if (a[i][0] >= end) {
    end = a[i][1];
    count++;
  }
}

console.log( message: count);
```

Python


```python
n = int(x= input())
a = [list(iterable= map(func= int, input().split())) for _ in range(stop= n)]

a.sort(key=lambda x: (x[1], x[0]))

count, end = 0, -1
for i in range(stop= n):
    if a[i][0] >= end:
        end = a[i][1]
        count += 1

print(values[0]= count)
```

채점 결과

회의실 배정 : 1931

hyunook02	1931	맞았습니다!!	60360 KB	2704 ms	Python 3 / 수정	225 B
hyunook02	1931	맞았습니다!!	41344 KB	344 ms	node.js / 수정	427 B
hyunook02	1931	맞았습니다!!	174252 KB	1076 ms	Java 11 / 수정	873 B

잃어버린 괄호 : 1541

문제

건우는 양수와 **+**, **-**, **괄호**를 가지고 식을 만들었다. 그리고 나서 건우는 괄호를 모두 지웠다.
그리고 나서 건우는 괄호를 적절히 쳐서 이 식의 값을 최소로 만들려고 한다.
괄호를 적절히 쳐서 **이 식의 값을 최소**로 만드는 프로그램을 작성하시오.

입력

첫째 줄에 식이 주어진다. 식은 **'0'~'9', '+', 그리고 '-'**만으로 이루어져 있고, 가장 처음과 마지막 문자는 숫자이다.
그리고 연속해서 두 개 이상의 연산자가 나타나지 않고, 5자리보다 많이 연속되는 숫자는 없다.
수는 0으로 시작할 수 있다. 입력으로 주어지는 식의 길이는 50보다 작거나 같다.

출력

첫째 줄에 정답을 출력한다.

예제 입력 1 복사	예제 출력 1 복사
55-50+40	-35

예제 입력 2 복사	예제 출력 2 복사
10+20+30+40	100

예제 입력 3 복사	예제 출력 3 복사
00009-00009	0

생각해보기

✦ 괄호 없이 계산할 때

55	-	50	+	40	=	45

✦ 앞의 괄호일 때

(55	-	50)	+	40	=	45

✦ 뒤의 괄호일 때

55	-	(50	+	40)	=	-35

생각해보기

- 괄호 없이 계산할 때

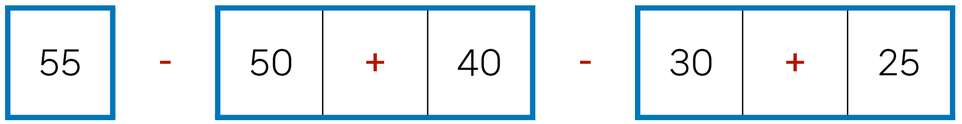
55	-	50	+	40	-	30	+	25	=	50

- 빼기를 기준으로 묶을 때

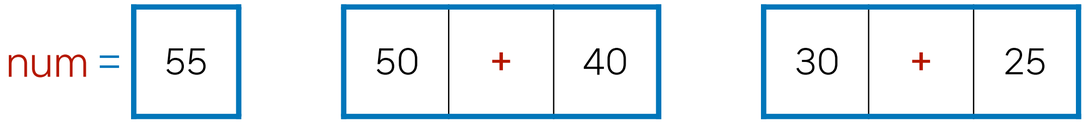
55	-	(50	+	40)	-	(30	+	25)	=	-90

알고리즘 구성

- 입력받은 문자열을 빼기(-) 를 기준으로 분할

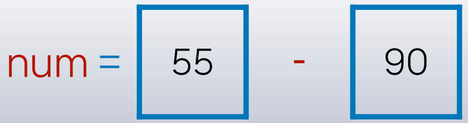


- 첫번째 수를 변수에 저장



- 나머지 수에 대해, 더하기(+)를 기준으로 분할

- 분할된 값을 더해서 num 값에 뺄셈



```java
public class P1541 {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);

        String[] strArr = in.nextLine().split("-");

        int ans = 0;
        for(String s : strArr[0].split("\\+"))
            ans += Integer.parseInt(s);

        for(int i = 1; i < strArr.length; i++) {
            for (String s : strArr[i].split("\\+"))
                ans -= Integer.parseInt(s);
        }

        System.out.println(ans);
    }
}
```

```
const fs = require('fs');
const input = fs.readFileSync("/dev/stdin").toString().trim();

const str = input.split("-");
let ans = str[0].split("+").map(Number).reduce((a, b) => a + b);

for (let i = 1; i < str.length; i++)
  ans -= str[i].split("+").map(Number).reduce((a, b) => a + b);

console.log( data[0]: ans);
```

Python


```python
import sys
input = sys.stdin.readline

str = list(iterable= input().split("-"))
num = sum(iterable= map(func= int, str[0].split(sep= "+")))

for i in range(start= 1, stop= len(obj= str)):
  num -= sum(iterable= map(func= int, str[i].split(sep= "+")))

print(values[0]= num)
```

채점 결과

잃어버린 괄호 : 1541

hyunook02	1541	맞았습니다!!	32412 KB	40 ms	Python 3 / 수정
hyunook02	1541	맞았습니다!!	9340 KB	96 ms	node.js / 수정
hyunook02	1541	맞았습니다!!	17652 KB	172 ms	Java 11 / 수정

소수 구하기 : 1929

문제

M이상 N이하의 소수를 모두 출력하는 프로그램을 작성하시오.

입력

첫째 줄에 자연수 M과 N이 빈 칸을 사이에 두고 주어진다. (1 ≤ M ≤ N ≤ 1,000,000) M이상 N이하의 소수가 하나 이상 있는 입력만 주어진다.

출력

한 줄에 하나씩, 증가하는 순서대로 소수를 출력한다.

예제 입력 1 복사

```
3 16
```

예제 출력 1 복사

```
3
5
7
11
13
```

```java
public static void main(String[] args) {
    Scanner in = new Scanner(source: System.in);
    int m = in.nextInt(), n = in.nextInt();

    boolean[] che = new boolean[n + 1];
    che[1] = true;
    for (int i = 2; i <= Math.sqrt(a: n); i++) {
        if (che[i])
            continue;

        for (int j = i + i; j <= n; j += i)
            che[j] = true;
    }

    for (int i = m; i <= n; i++)
        if (!che[i])
            System.out.println(x: i);
}
```

JavaScript


```
const [m, n] = require('fs')
        .readFileSync('/dev/stdin')
        .toString()
        .trim()
        .split( separator: ' ')
        .map( callbackfn: Number);

let che = Array.from({length: n + 1}, () => false);
che[1] = true;

for (let i = 2; i <= Math.sqrt(n); i++) {
  if (che[i])
    continue;

  for (let j = i + i; j <= n; j += i)
    che[j] = true;
}

for (let i = m; i <= n ; i++)
  if (!che[i])
    console.log(i);
```

Python


```python
import math

m, n = map(func= int, input().split())
a = [False] * (n + 1)
a[1] = True

for i in range(start= 2, stop= math.floor(x= math.sqrt(x= n)) + 1):
    if a[i]:
        continue
    for j in range(start= i + i, stop= n + 1, step= i):
        a[j] = True

for i in range(start= m, stop= n + 1):
    if not a[i]:
        print(values[0]= i)
```

채점 결과

소수 구하기 : 1929

hyunook02	1929	맞았습니다!!	42212 KB	280 ms	Python 3 / 수정
hyunook02	1929	맞았습니다!!	33440 KB	776 ms	Java 11 / 수정
hyunook02	1929	맞았습니다!!	28384 KB	2596 ms	node.js / 수정

GCD(n, k) = 1 : 11689

문제

자연수 n이 주어졌을 때, GCD(n, k) = 1을 만족하는 자연수 $1 \le k \le n$ 의 개수를 구하는 프로그램을 작성하시오.

입력

첫째 줄에 자연수 n ($1 \le n \le 1012$)이 주어진다.

출력

GCD(n, k) = 1을 만족하는 자연수 $1 \le k \le n$ 의 개수를 출력한다.

GCD(n, k) = 1 : 11689

★ 예제

예제 입력 1 복사	예제 출력 1 복사
1	1
예제 입력 2 복사	예제 출력 2 복사
5	4
예제 입력 3 복사	예제 출력 3 복사
10	4
예제 입력 4 복사	예제 출력 4 복사
45	24
예제 입력 5 복사	예제 출력 5 복사
99	60

```java
public static void main(String[] args) throws IOException {
    BufferedReader in = new BufferedReader(in: new InputStreamReader(in: System.in));
    long n = Long.parseLong(s: in.readLine());
    long result = n;

    for (long p = 2; p <= Math.sqrt(a: n); p++) {
        if (n % p == 0) {
            result = result - result / p;
            while (n % p == 0)
                n /= p;
        }
    }

    if (n > 1)
        result = result - result / n;
    System.out.println(x: result);
}
```

JavaScript


```
let n = parseInt( string: require('fs').readFileSync('/dev/stdin').toString());
let result = n;

for (let p = 2; p <= Math.sqrt( x: n); p++) {
  if (n % p === 0) {
    result -= result / p;
    while (n % p === 0)
      n /= p;
  }
}

if (n > 1)
  result -= result / n;
console.log( message: result);
```

Python

GCD(n, k) = 1 : 11689

```python
import math

n = int(x= input())
result = n

for p in range(start= 2, stop= math.floor(x= math.sqrt(x= n)) + 1):
    if n % p == 0:
        result -= result // p
        while n % p == 0:
            n /= p

if n > 1:
    result -= result // n

print(values[0]= int(x= result))
```

채점 결과

GCD(n, k) = 1 : 11689

hyunook02	11689	맞았습니다!!	34536 KB	204 ms	Python 3 / 수정
hyunook02	11689	맞았습니다!!	10296 KB	160 ms	node.js / 수정
hyunook02	11689	맞았습니다!!	14372 KB	108 ms	Java 11 / 수정

최소공배수 : 1934

문제

두 자연수 A와 B에 대해서, A의 배수이면서 B의 배수인 자연수를 A와 B의 공배수라고 한다. 이런 공배수 중에서 가장 작은 수를 최소공배수라고 한다. 예를 들어, 6과 15의 공배수는 30, 60, 90등이 있으며, 최소 공배수는 30이다.

두 자연수 A와 B가 주어졌을 때, A와 B의 최소공배수를 구하는 프로그램을 작성하시오.

입력

첫째 줄에 테스트 케이스의 개수 T(1 ≤ T ≤ 1,000)가 주어진다. 둘째 줄부터 T개의 줄에 걸쳐서 A와 B가 주어진다. (1 ≤ A, B ≤ 45,000)

출력

첫째 줄부터 T개의 줄에 A와 B의 최소공배수를 입력받은 순서대로 한 줄에 하나씩 출력한다.

예제 입력 1 복사

```
3
1 45000
6 10
13 17
```

예제 출력 1 복사

```
45000
30
221
```

```java
public static int gcd(int a, int b) {
    if (b == 0)
        return a;
    return gcd(a: b, b: a % b);
}

Run | Debug
public static void main(String[] args) {
    Scanner in = new Scanner(source: System.in);
    int t = in.nextInt();

    while (t-- > 0) {
        int a = in.nextInt(), b = in.nextInt();
        System.out.println(x: a * b / gcd(undefined: a, undefined: b));
    }
}
```

풀이

JavaScript

최소공배수 : 1934

```
let input = require('fs')
    .readFileSync('/dev/stdin')
    .toString()
    .trim()
    .split( separator: '\n');

const gcd = (a, b) => {
  if (b === 0)
    return a;
  return gcd(b, a % b);
}

const t = parseInt( string: input[0]);
for (let i = 0; i < t; i++) {
  const [a, b] = input[i + 1].split( separator: ' ').map( callbackfn: Number);
  console.log( message: Math.floor(a * b / gcd(a, b)));
}
```

Python


```python
def gcd(x, y):
    if y == 0:
        return x
    return gcd(x= y, y= x % y)

for _ in range(stop= int(x= input())):
    x, y = map(func= int, input().split())
    print(values[0]= x * y // gcd(x, y))
```

채점 결과

최소공배수 : 1934

hyunook02	1934	맞았습니다!!	12056 KB	196 ms	node.js / 수정
hyunook02	1934	맞았습니다!!	32412 KB	88 ms	Python 3 / 수정
hyunook02	1934	맞았습니다!!	21896 KB	280 ms	Java 11 / 수정