GeeksforGeeks
A computer science portal for geeks

es

GeeksforGeeks
A computer science portal for geeks

Custom Search

in

Fast I/O in Java in Competitive Programming

Using Java in competitive programming is not something many people would suggest just because of its slow input and output, and well indeed it is slow.

In this article, we have discussed some ways to get around the difficulty and change the verdict from TLE to (in most cases) AC.

For all the Programs below

Input:

```
7 3
1
51
966369
7
9
999996
11
```

Output:

```
4
```

1. **Scanner Class** – (easy, less typing, but not recommended very slow, refer this for reasons of slowness): In most of the cases we get TLE while using scanner class. It uses built-in nextInt(), nextLong(), nextDouble methods to read the desired object after initiating scanner object with input stream.(eg System.in). The following program many a times gets time limit exceeded verdict and therefore not of much use.

```java
// Working program using Scanner
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.util.Scanner;
public class Main
{
    public static void main(String[] args)
    {
        Scanner s = new Scanner(System.in);
        int n = s.nextInt();
        int k = s.nextInt();
        int count = 0;
        while (n-- > 0)
        {
            int x = s.nextInt();
            if (x%k == 0)
                count++;
        }
        System.out.println(count);
    }
}
```

2. **BufferedReader** – (fast, but not recommended as it requires lot of typing): The Java.io.BufferedReader class reads text from a character-input stream, buffering characters so as to provide for the efficient reading of characters, arrays, and lines. With this method we will have to parse the value every time for desired type. Reading multiple words from single line adds to its complexity because of the use of Stringtokenizer and hence this is not recommended. This gets accepted with a running time of approx 0.89 s.but still as you can see it requires a lot of typing all together and therefore method 3 is recommended.

```java
// Working program using BufferedReader
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.StringTokenizer;

public class Main
{
    public static void main(String[] args) throws IOException
    {

        BufferedReader br = new BufferedReader(
                            new InputStreamReader(System.in));

        StringTokenizer st = new StringTokenizer(br.readLine());
        int n = Integer.parseInt(st.nextToken());
        int k = Integer.parseInt(st.nextToken());
        int count = 0;
        while (n-- > 0)
        {
```

```
            int x = Integer.parseInt(br.readLine());
            if (x%k == 0)
                count++;
        }
        System.out.println(count);
    }
}
```

3. **Userdefined FastReader Class-** (which uses bufferedReader and StringTokenizer): This method uses the time advantage of BufferedReader and StringTokenizer and the advantage of user defined methods for less typing and therefore a faster input altogether. This gets accepted with a time of 1.23 s and this method is *very much recommended* as it is easy to remember and is fast enough to meet the needs of most of the question in competitive coding.

```java
// Working program with FastReader
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.Scanner;
import java.util.StringTokenizer;

public class Main
{
    static class FastReader
    {
        BufferedReader br;
        StringTokenizer st;

        public FastReader()
        {
            br = new BufferedReader(new
                    InputStreamReader(System.in));
        }

        String next()
        {
            while (st == null || !st.hasMoreElements())
            {
                try
                {
                    st = new StringTokenizer(br.readLine());
                }
                catch (IOException  e)
                {
                    e.printStackTrace();
                }
            }
            return st.nextToken();
        }

        int nextInt()
        {
            return Integer.parseInt(next());
        }

        long nextLong()
        {
            return Long.parseLong(next());
        }

        double nextDouble()
```

```java
        {
            return Double.parseDouble(next());
        }

        String nextLine()
        {
            String str = "";
            try
            {
                str = br.readLine();
            }
            catch (IOException e)
            {
                e.printStackTrace();
            }
            return str;
        }
    }

    public static void main(String[] args)
    {
        FastReader s=new FastReader();
        int n = s.nextInt();
        int k = s.nextInt();
        int count = 0;
        while (n-- > 0)
        {
            int x = s.nextInt();
            if (x%k == 0)
                count++;
        }
        System.out.println(count);
    }
}
```

4. **Using Reader Class**: There is yet another fast way through the problem, I would say the fastest way but is not recommended since it requires very cumbersome methods in its implementation. It uses inputDataStream to read through the stream of data and uses read() method and nextInt() methods for taking inputs. This is by far the fastest ways of taking input but is difficult to remember and is cumbersome in its approach. Below is the sample program using this method. This gets accepted with a surprising time of just 0.28 s. Although this is ultra fast, it is clearly not an easy method to remember.

```java
// Working program using Reader Class
import java.io.DataInputStream;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.Scanner;
import java.util.StringTokenizer;

public class Main
{
    static class Reader
    {
        final private int BUFFER_SIZE = 1 << 16;
        private DataInputStream din;
        private byte[] buffer;
        private int bufferPointer, bytesRead;
```

```java
    public Reader()
    {
        din = new DataInputStream(System.in);
        buffer = new byte[BUFFER_SIZE];
        bufferPointer = bytesRead = 0;
    }

    public Reader(String file_name) throws IOException
    {
        din = new DataInputStream(new FileInputStream(file_name));
        buffer = new byte[BUFFER_SIZE];
        bufferPointer = bytesRead = 0;
    }

    public String readLine() throws IOException
    {
        byte[] buf = new byte[64]; // line length
        int cnt = 0, c;
        while ((c = read()) != -1)
        {
            if (c == '\n')
                break;
            buf[cnt++] = (byte) c;
        }
        return new String(buf, 0, cnt);
    }

    public int nextInt() throws IOException
    {
        int ret = 0;
        byte c = read();
        while (c <= ' ')
            c = read();
        boolean neg = (c == '-');
        if (neg)
            c = read();
        do
        {
            ret = ret * 10 + c - '0';
        } while ((c = read()) >= '0' && c <= '9');

        if (neg)
            return -ret;
        return ret;
    }

    public long nextLong() throws IOException
    {
        long ret = 0;
        byte c = read();
        while (c <= ' ')
            c = read();
        boolean neg = (c == '-');
        if (neg)
            c = read();
        do {
            ret = ret * 10 + c - '0';
        }
        while ((c = read()) >= '0' && c <= '9');
        if (neg)
            return -ret;
        return ret;
    }
```

```java
    public double nextDouble() throws IOException
    {
        double ret = 0, div = 1;
        byte c = read();
        while (c <= ' ')
            c = read();
        boolean neg = (c == '-');
        if (neg)
            c = read();

        do {
            ret = ret * 10 + c - '0';
        }
        while ((c = read()) >= '0' && c <= '9');

        if (c == '.')
        {
            while ((c = read()) >= '0' && c <= '9')
            {
                ret += (c - '0') / (div *= 10);
            }
        }

        if (neg)
            return -ret;
        return ret;
    }

    private void fillBuffer() throws IOException
    {
        bytesRead = din.read(buffer, bufferPointer = 0, BUFFER_SIZE);
        if (bytesRead == -1)
            buffer[0] = -1;
    }

    private byte read() throws IOException
    {
        if (bufferPointer == bytesRead)
            fillBuffer();
        return buffer[bufferPointer++];
    }

    public void close() throws IOException
    {
        if (din == null)
            return;
        din.close();
    }
}

public static void main(String[] args) throws IOException
{
    Reader s=new Reader();
    int n = s.nextInt();
    int k = s.nextInt();
    int count=0;
    while (n-- > 0)
    {
        int x = s.nextInt();
        if (x%k == 0)
            count++;
    }
    System.out.println(count);
```

```
        }
    }
```

Suggested Read: Enormous Input Test designed to check the fast input handling of your language.Timings of all programs are noted from SPOJ.

This article is contributed by **Rishabh Mahrsee**. If you like GeeksforGeeks and would like to contribute, you can also write an article and mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

Recommended Posts:

Fast I/O for Competitive Programming

Tips and Tricks for Competitive Programmers | Set 2 (Language to be used for Competitive Programming)

Java tricks for competitive programming (for Java 8)

Input/Output from external file in C/C++, Java and Python for Competitive Programming | Set 2

Input/Output from external file in C/C++, Java and Python for Competitive Programming

How can competitive programming help you get a job?

Bit Tricks for Competitive Programming

A Better Way To Approach Competitive Programming

How to become a master in competitive programming?

C++ tricks for competitive programming (for C++ 11)

What is Competitive Programming and How to Prepare for It?

Python in Competitive Programming

What Are The Best Resources For Competitive Programming?

How to begin with Competitive Programming?

Knowing the complexity in competitive programming

Article Tags :   Competitive Programming   Java

Practice Tags : Java

👍

30

3.5

☐ To-do ☐ Done

Based on **73** vote(s)

(Feedback/ Suggest Improvement) (Add Notes) (Improve Article)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Load Comments

A computer science portal for geeks

5th Floor, A-118,
Sector-136, Noida, Uttar Pradesh - 201305
feedback@geeksforgeeks.org

COMPANY

About Us
Careers
Privacy Policy
Contact Us

LEARN

Algorithms
Data Structures
Languages
CS Subjects
Video Tutorials

PRACTICE

Courses
Company-wise
Topic-wise
How to begin?

CONTRIBUTE

Write an Article
Write Interview Experience
Internships
Videos