

# Advanced Algorithms and Datastructures

Challenges and Solutions

Last modified:  
12th February 2018

# Contents

<b>1</b>	<b>Introduction &amp; Rules</b>	<b>2</b>
<b>I</b>	<b>Challenges</b>	<b>3</b>
<b>2</b>	<b>Aku (AKU) - 40 points</b>	<b>4</b>
<b>3</b>	<b>Multiplication Tables (TABLE) - 40 points</b>	<b>6</b>
<b>II</b>	<b>Reference Solutions</b>	<b>8</b>

# 1 | Introduction & Rules

Over the duration of this course challenges will continually become available. Solving these challenges will allow you to accumulate points. These points will, in the end, translate directly into your end grade. There are however some rules, which will be explained below.

**Submitting:** For submitting your solutions you will have to upload your programs to Themis. Each submission should be made individually and before the deadline, as the submission tool will close afterwards.

Your submission should contain three things. Namely, the best performing algorithm you could think of that solves the problem, a short time complexity analysis, and a short memory complexity analysis.

**Grading:** You will obtain points for the amount of test cases you manage to pass. The points will be distributed equally over all available test cases for a challenge. I.e. if a challenge has 10 test cases and a maximum reward of 40 points, then each passed test case rewards you 4 points.

Each point directly translates into .01 grade point. Meaning that a challenge of 40 points may increase your grade by .4, potentially. Although, you will be able to accumulate more than 600 points through all the challenges, only the first 600 points will influence your end grade.

**Libraries & Plagiarism:** We will never ask you to reinvent the wheel or ignite a burning candle. You are advised to make use of existing libraries where you see fit. This only applies to the standard libraries, though. Third-party dependencies are prohibited. Additionally, your submissions will be checked for plagiarism, so make sure your submission contains your own work only.

**Penalties:** A maximum of 25 % of the maximum points assigned to a challenge may be withheld in case of a missing or incorrect complexity analysis. This holds for both the time complexity and the memory complexity. Furthermore, double that amount (50 % of max points) may be withheld when your solution is suboptimal.

**Bonuses:** In case you manage to solve a challenge in two or more different languages the maximum amount of points you may obtain from that challenge will increase with 10 %. This is under the condition that in all languages all test cases are passed and no penalties have been applied.

Additionally, you may choose to implement the data structure on which your algorithm relies yourself. So you will not be using the pre-implemented libraries for your solution. If done correctly the maximum amount of points you can obtain for that challenge will increase with 25 %. This bonus stacks with the multiple languages bonus, and there is no need to implement the data structure in both languages in order to be eligible for that bonus. Also, this bonus is only awarded once per data structure across all challenge. In other words, once for a stack, once for a heap, etc. This bonus comes with a downside, as you will be eligible for more penalties.

Furthermore, points are awarded for completing HackerRank challenges. The amount of points awarded is one-tenth the amount the challenge is worth. Through this bonus you may obtain a maximum of 25 points.

# I | Challenges

## 2 | Aku (AKU) - 40 points

Your friend Samurai Jack, a young prince, has received a magic katana in order to defeat and imprison the supernatural shape-shifting demon Aku. The demon Aku is so supernatural that now he has transformed itself into a natural number! Jack has to very quickly perform a series of complicated moves in order to defeat Aku.



Initially, Jack sees on the field (which is a sequence of integers) only the number  $n$  — the number in which Aku hides. Then he has to battle in the following way: each move, Jack uses his katana to split any number  $x > 1$  on the field, into the numbers  $\lfloor \frac{x}{2} \rfloor, x \bmod 2, \lfloor \frac{x}{2} \rfloor$  which appear sequentially in the same position. In this way, Jack has to dissect each part of Aku until all the numbers on the field are either 0 or 1. Notice that the final state of the field is invariant under the order of the operations performed.

Now, in order to make the final move, he needs to count the total number of 1s in the range of  $l$  to  $r$  (1-indexed). Unfortunately Jack cannot do it sufficiently fast and asks you for help.

### 2.1 | Input

The first line contains three integers  $n, l, r$  ( $0 \leq n < 2^{50}$ ,  $0 \leq r - l \leq 10^5$ ,  $r \geq 1$ ,  $l \geq 1$ ) — the initial number of Aku and the range from  $l$  to  $r$ . It is guaranteed that  $r$  is not greater than the length of the final list.

### 2.2 | Output

Print the number of 1s in the given range on final state of the field.

### 2.3 | Example

Input	Output
7 2 5	4
10 3 10	5

## 2.4 | Note

Consider the first example:

$[7] \rightarrow [3, 1, 3] \rightarrow [1, 1, 1, 1, 3] \rightarrow [1, 1, 1, 1, 1, 1, 1]$  The elements on the positions from 2 to 5 are  $[1, 1, 1, 1]$  so the answer is 4.

$[10] \rightarrow \dots \rightarrow [1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1]$  The elements on the positions from 3 to 10 are  $[1, 1, 1, 0, 1, 0, 1, 0]$  so the answer is 5.

## 2.5 | Restrictions

Remember that the **BufferedReader** in Java is faster than the **Scanner**, this may mean the difference between reaching the time limit and passing the tests.

### 3 | Multiplication Tables (TABLE) - 40 points

While you were learning the multiplication table in the primary school, Jaap was having some fun on his own.



Jaap draw a  $n \times m$  multiplication table, where the element on the intersection of the  $i$ -th row and  $j$ -th column equals  $i \cdot j$ , given that the rows and the columns are 1-indexed. Jaap's PhD supervisor then asked him: what number in the table is the  $k$ -th largest number? Jaap answered correctly within two seconds. Can you do it faster?

Consider the given multiplication table. If you write out all the  $n \cdot m$  numbers from the table in non-decreasing order, then the  $k$ -th number you write out is called the  $k$ -th largest number on the table.

#### 3.1 | Input

The single line in the input contains three integers:  $n$ ,  $m$  and  $k$  ( $1 \leq n, m \leq 10^5$ ;  $1 \leq k \leq n \cdot m$ ).

#### 3.2 | Output

Print the  $k$ -th largest number in a  $n \times m$  multiplication table.

#### 3.3 | Example

Input	Output
2 2 2	2
2 3 4	3
1 10 5	5

#### 3.4 | Note

The  $2 \times 3$  multiplication table looks like this:

$$\begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 6 \end{bmatrix}$$

## 3.5 | Restrictions

Remember that the `BufferedReader` in Java is faster than the `Scanner`, this may mean the difference between reaching the time limit and passing the tests.

## II | Reference Solutions