mBIT Sample Problems

mBIT Organizers

July 1, 2019

These problems are meant to help you decide which division to register for. We have provided problems of varying difficulty within each division to give you a general idea of what to expect, but these don't necessarily represent the full range of difficulty in either direction.

It's fine if you don't know how to solve the harder problems in your division– we intentionally made them difficult to challenge you! Also, keep in mind that there will be several easier problems in the rookie division for students newer to programming.

Another thing to remember is that there are time constraints for these problems. This means that your program must finish running on our test data in less than a few seconds to get credit. While efficiency may not be as relevant for easier problems, most varsity-division problems cannot be solved without significant optimizations to your code.

Contents

Thanks to Evan Chen for letting us use his style file

§1 Rookie

§1.1 Easy: Number Jumping

Problem Statement:

Kevin is on a number line, starting at 0. At each second, Kevin jumps to a number that is two more than the number he was previously at. Kevin will stop jumping if his next jump will result in him being located at a number greater than N. Output all numbers that Kevin will be situated at at any given time in ascending order.

Input Format:

The first line contains an integer N $(1 \leq N \leq 200)$.

Output Format:

Output the numbers Kevin will have visited in ascending order on one line, each number separated by a space.

Sample Input:

```
9
```

Sample Output:

```
0 2 4 6 8
```

§1.2 Medium: Array Division

Problem Statement:

Gabe has been given the job of dividing up an array of numbers. Formally, given an array of size N, with numbers $a_1, \ldots, a_N$, Gabe wants to split the array into two arrays, a left array $a_1, \ldots, a_i$ and a right array $a_{i+1}, \ldots, a_N$, where each array has at least one element. Find the number of ways Gabe can split the array so that the sum of the elements in the left array is strictly less than the sum of the elements in the right array.

Input Format:

The first line will contain an integer N $(1 \leq N \leq 1000)$.

The next line will contain N space separated integers representing $a_1, \ldots, a_N$ $(-1000 \leq a_i \leq 1000)$.

Output Format:

Output a single number representing the number of ways Gabe can split the array to satisfy the conditions given in the problem.

Sample Input:

```
5
5 -4 6 -5 3
```

Sample Output:

```
2
```

Gabe can split in exactly two ways. His options are:

```
5 -4    6 -5 3
5 -4 6 -5    3
```

Each split results in the left side having a sum less than the right side. As a result, the final output is 2.

§1.3 Hard: Bunny Market

It is Farmer Tim's lifelong goal to have exactly B bunnies. He currently has A bunnies. Every night, his current herd of bunnies reproduces, and their number doubles. Every day, Tim can choose whether or not to buy K bunnies from his local market (he must buy exactly 0 or exactly K bunnies, the sellers are very picky). Output the day Tim will first be able to end with exactly B bunnies, if he ever will. Note that Tim starts with A bunnies in the morning on day 1, so he can go to the market before the first nightfall.

Input Format:

The first line will contain three integers: A, B, K $(1 \leq A, B, K \leq 10^9)$.

Output Format:

Output a single line containing the first day that Tim will be able to have B bunnies, or -1 if he will never be able to have B bunnies.

Sample Input:

```
1 12 2
```

Sample Output:

```
3
```

Tim starts with 1 bunny, and wants to get 12. The best strategy is for Tim to buy 2 bunnies on day 1, for a total of 3 bunnies. They double that night so he has 6 by the morning of day 2. He doesn't go to the market on day 2. Once he lets them double overnight, he has 12 bunnies by the morning of day 3. The final output is 3. Note that Tim had the chance to buy bunnies on day 3, and if he had reached his goal by doing so, the answer would still be 3.

§2 Varsity

§2.1 Easy: Log Jumping

Problem Statement:

Elias is jumping across logs on a river that extends infinitely in either direction, which can be thought of as a number line. There is either a log (which he can jump on) or a crocodile (which he can not jump on) at each number. He starts at location 0, and wants to get to location N. Elias is unable to jump backwards, and can only make jumps of certain distances. You are given an array, s, which contains the different size jumps Elias can make. After making a jump of size s_i from location x, Elias will be at location $x + s_i$. Array s is of length M, $1 \leq M \leq 100$.

In K locations, there are crocodiles instead of logs. If Elias jumps on one of these locations, he is eaten by the crocodile and does not get to his desired location. Find the minimum number of jumps it takes for Elias to get from location 0 location N without jumping on a crocodile, or output -1 if he can never get to N. Note that jumping past N without ever jumping *on* N does not count as reaching N.

Input Format:

The first line will contain an integer N ($1 \leq N \leq 10,000$), followed by an integer M ($1 \leq M \leq 100$), and an integer K ($0 \leq K \leq N$), each separated by a single space.

The second line will contain M space-separated integers which represent $s_1, s_2, \ldots s_M$, the distances Elias is able to jump ($1 \leq s_i \leq 10^9$).

The third line will contain K space-separated integers, the positions which contain crocodiles.

Output Format:

Output a single line containing the minimum number of jumps Elias can take to get to location N without jumping on a crocodile.

Sample Input:

```
9 2 1
1 3
3
```

Sample Output:

```
5
```

To reach location 9, Elias can make a jump of size 1, then 2 jumps of size 3, followed by 2 jumps of size 1. This is a total of 5 jumps, which is the minimum number of jumps needed to avoid the crocodile at location 3.

§2.2 Medium: Monster Sums

Problem Statement:

There are two friendly monsters named Angelina and Bert. Each monster has their own array of length N. Angelina's array is $a_1, \ldots, a_N$, and Bert's array is $b_1, \ldots, b_N$. Both monsters want to take a specific prefix sum of their array so that their values are as close as possible. What this means is that Angelina chooses an i between 1 and N and computes $A_i = a_1 + a_2 + \ldots a_i$, and Bert chooses a j between 1 and N and computes $B_j = b_1 + b_2 + \ldots b_j$. They want to work together to choose i, j that minimizes the absolute difference between A_i and B_j. What is this minimum absolute difference?

Input Format:

The first line will contain an integer N $(1 \leq N \leq 100,000)$.

The second line will contain integers representing $a_1, \ldots, a_N$.

The third line will contain integers representing $b_1, \ldots, b_N$. It is guaranteed that all values in the arrays have a magnitude less than 10^9.

Output Format:

Output a single line containing the least possible absolute difference between A_i and B_j.

Sample Input:

```
5
1 2 4 2 -9
-4 3 7 0 2
```

Sample Output:

```
1
```

One way to reach a difference of 1 is when $i = 4$ and $j = 5$.
$A_4 = 1 + 2 + 4 + 2 = 9$ and $B_5 = -4 + 3 + 7 + 0 + 2 = 8$. Thus, $|A_i - B_j| = 1$.

§2.3 Hard: Close Subsequences

Problem Statement:

Samir is very particular about his integer sequences. He starts with a positive integer sequence of length N (denoted as $a_1, \dots a_N$), and chooses a subsequence of this sequence. We define a subsequence to be a sequence formed by deleting several elements of the original sequence a_i. For example, a subsequence of $\{1, 2, 4, 3, 5\}$ could be $\{1, 4, 5\}$, but not $\{1, 3, 4\}$, because the order was changed. Samir likes it when all adjacent integers in his subsequence have an absolute difference of at most K and at most B when taken the difference is taken mod M. He calls subsequences that satisfy these conditions *close* sequences. More formally, Samir is satisfied with a subsequence $b_1, \dots, b_m$ when $|b_i - b_{i+1}| \leq m$ and $|b_i - b_{i+1}| \% M \leq B$ for all $1 \leq i < m$ ($x \% y$ means the remainder when x is divided by y). What is the length of the longest subsequence of $a_1, \dots a_N$ that is close?

Input Format:

The first line will contain N ($1 \leq N \leq 100,000$), K ($0 \leq K \leq 10^9$), M ($2 \leq M \leq 5$), and B ($0 \leq B \leq M$).

The second line will contain integers representing $a_1, \dots, a_N$ ($1 \leq a_i \leq 10^9$).

Output Format:

Output a single line containing the maximum length of a close subsequence of $a_1, \dots a_N$.

Sample Input:

```
6 6 5 1
4 7 2 3 13
```

Sample Output:

```
3
```

Deleting the 4 and the 13 results in a sequence of $7, 2, 3$. This sequence is close because the differences between adjacent elements never exceed $K = 6$ and the differences mod $M = 5$ never exceed $B = 1$. This is the longest such sequence, so the final output is 3.

Note: If you want a special challenge, change the bounds for M to be $2 \leq M \leq N$. This wasn't included in the original problem because we thought it would be too hard, but feel free to try to solve this extension.