Eidgenössische
Technische Hochschule
Zürich

Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Federal Institute of Technology at Zurich

Departement of Computer Science
Markus Püschel, David Steurer
Karel Kubicek, Johannes Lengler

11 November 2019

Algorithms & Data Structures — Programming task 9 — HS 19

The solutions for this sheet need to be submitted to the judge by **Monday, 25 November, 23:59:59**.

Exercise 9.1 *Skip stack* **(2 bonus point)**.

Your task is to implement a slightly modified version of stack for storing integers. That means choosing your representation of the data structure, and implementing the methods `push` and `pop`.

`push` inserts the passed `item` to the stack, to one the two highest positions. If the value `item` is at least as high as the highest item of the stack, then it is inserted on top, otherwise `item` is inserted below the top. In case of an empty stack, `push` works as for a normal stack.

`pop` removes an item from the stack and returns its value. As item for removal, the larger out of the top two items is selected. If the stack contains only a single item, `pop` works the same as for normal stack.

As the stored items are primitive integers, it does not matter which of the top two items you select when their values are the same.

Please find a more detailed description of the task and examples in the `Main.java` file. The methods that you have to implement are denoted by `TODO` and contain a description of the task. The data format is described in the class `ReadAndWrite`. The Eclipse project also contains a set of basic tests, execute them by running `JUnitTest.java`.

You get one point for each passing test set. The test set `small` does not test time complexity. That is done by the test set `large`.

Submission: Submit your `Main.java` at `https://judge.inf.ethz.ch/team/websubmit.php?cid=28784&problem=AD19H9P1`. The enrollment password is "`asymptotic`".

Exercise 9.2 *Shopping list* **(2 bonus point)**.

Your task is to compute the lowest price for the following shopping strategy.

You are neither Coop- nor Migros-child, as your parents could never agree on where to go shopping. You want to support both shops equally, so you buy always half of the products in Coop and half in Migros. However, as a student, you still want to safe money, so you have to select where to buy every particular product.

More formally, your task is to find the lowest price for such an equal split shopping task for $k = 2, 3$ shops. As input, you are getting k, an integer n that denotes the number of products in your shopping list, and then k arrays of length n with prices for each of the shops. Note that n is always divisible by k so that shopping can always be split evenly.

Consider the following example for $k = 2$.

Input (the test sets does not include the annotation):

```
k=2

n=4

p_c=1 2 3 4

p_m=5 7 9 11
```

In this case, the lowest price is $5+7+3+4 = 19$ by shopping the first two items in Migros (`p_m` prices) and the last two in Coop.

Please find a more details about the input in the `Main.java` file. The methods that you have to implement are denoted by `TODO` and contain a description of the task. The data format is described in the class `ReadAndWrite`. The Eclipse project also contains a set of basic tests, execute them by running `JUnitTest.java`.

You get one point for each passing test set. The test set `two` tests the case of $k = 2$, while the test set `three` tests $k = 3$.

Submission: Submit your `Main.java` at `https://judge.inf.ethz.ch/team/websubmit.php?cid=28784&problem=AD19H9P2`. The enrollment password is "`asymptotic`".

Exercise 9.3 *Dyno* **(additional exercise, no bonus points)**.

This task is taken without any modification from Algorithmen und Datenstrukturen 2018, and the authors are Chih-Hung Liu and Stefano Leucci. The programming template was not modified to meet this year's policy. All graded tasks will always satisfy the structure and policy given in this year.

Dyno, the dinosaur of Figure 1, wants to cross a perfectly straight desert. The desert is L meters long and it is split into L segments, indexed from 0 to $L-1$. Each segment can either be *empty* or it can contain one of the C *cacti* that inhabit the desert. When Dyno is in a generic segment i it can either walk or jump forward. Walking allows it to move from segment i to segment $i+1$, provided that segment $i+1$ is empty. Jumping allows it to move from segment i to segment $i+D$, provided that segment $i+D$ is empty (Dyno can jump even if there are cacti between segment $i+1$ and segment $i+D-1$. Dyno starts at segment 0, which is always empty, and your job is to help Dyno reach the farthest possible segment in the desert (i.e., the one with the largest possible index).

Input

The input consists of a set of instances, or *test-cases*, of the previous problem. The first line of the input contains the number T of test-cases. The first line of each test case contains integers L, D and C, separated by spaces. The second line of each test case contains the locations of the C cacti as n integers separated by spaces. The segment numbers of the cacti locations appear in increasing order.

The inputs satisfy $10 \leq L \leq 1\,000\,000$, $0 \leq C \leq 1\,000\,000$, and $2 \leq D \leq 10$.

Output

The output consists of T lines, each containing a single integer. The i-th line is the answer to the i-th test-case, i.e., it contains the the largest index reachable by Dyno.

Grading There are no bonus points for this exercise. The points on Judge are just informatory. Your program should run in time $O(L \log(C+D))$. Submit your `Main.java` at `https://judge.inf.ethz.ch/team/websubmit.php?cid=28784&problem=AD18H9P2`. The enrollment password is "`asymptotic`".

Example

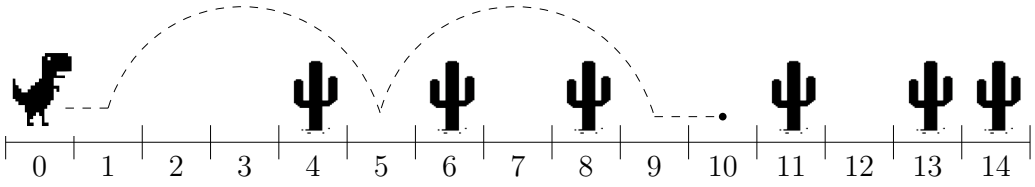

Figure 1: Example input. The dashed line represents the unique optimal solution.[1]

Input (corresponding to Figure 1):

```
1
15 4 6
4 6 8 11 13 14
```

Output:

```
10
```

[1] Dyno is not to scale.