Language: Java

Task 1

A positive integer N is given. The goal is to construct the shortest possible sequence of integers ending with N, using the following rules:

- the first element of the sequence is 1; more specifically: A[0] = 1,
- each of the following elements is generated by multiplying the previous element by 2 or increasing it by 1; more precisely: A[i] = A[i−1] * 2 or A[i] = A[i−1] + 1, for i ≥ 1.

For example, for N = 18, the shortest sequence is:
A[0] = 1
A[1] = 2
A[2] = 4
A[3] = 8
A[4] = 9
A[5] = 18

Write a function:

class Solution { public int solution(int N); }

that, given a positive integer N, returns the length of the shortest possible sequence of integers satisfying the above conditions and ending with N.

For example, given N = 18, the function should return 6, as explained above.

Write an **efficient** algorithm for the following assumptions:

- N is an integer within the range [1..2,147,483,647].

Task 2

You would like to find the sentence containing the largest number of words in some given text. The text is specified as a string S consisting of N characters: letters, spaces, dots (.), question marks (?) and exclamation marks (!).

The text can be divided into sentences by splitting it at dots, question marks and exclamation marks. A sentence can be divided into words by splitting it at spaces. A sentence without words is valid, but a valid word must contain at least one letter.

For example, given S = "We test coders. Give us a try?", there are three sentences: "We test coders", " Give us a try" and "". The first sentence contains three words: "We", "test" and "coders". The second sentence contains four words: "Give", "us", "a" and "try". The third sentence is empty.

Write a function:

class Solution { public int solution(String S); }

that, given a string S consisting of N characters, returns the maximum number of words in a sentence.

For example, given S = "We test coders. Give us a try?", the function should return 4, as explained above.

Given S = "Forget CVs..Save time . x x", the function should return 2, as there are four sentences: "Forget CVs" (2 words), "" (0 words), "Save time " (2 words) and " x x" (2 words).

Assume that:

- the length of S is within the range [1..100];
- string S consists only of letters (a−z, A−Z), spaces, dots (.), question marks (?) and exclamation marks (!). In your solution, focus on correctness. The performance of your solution will not be the focus of the assessment.

In your solution, focus on **correctness**. The performance of your solution will not be the focus of the assessment.

Task 3

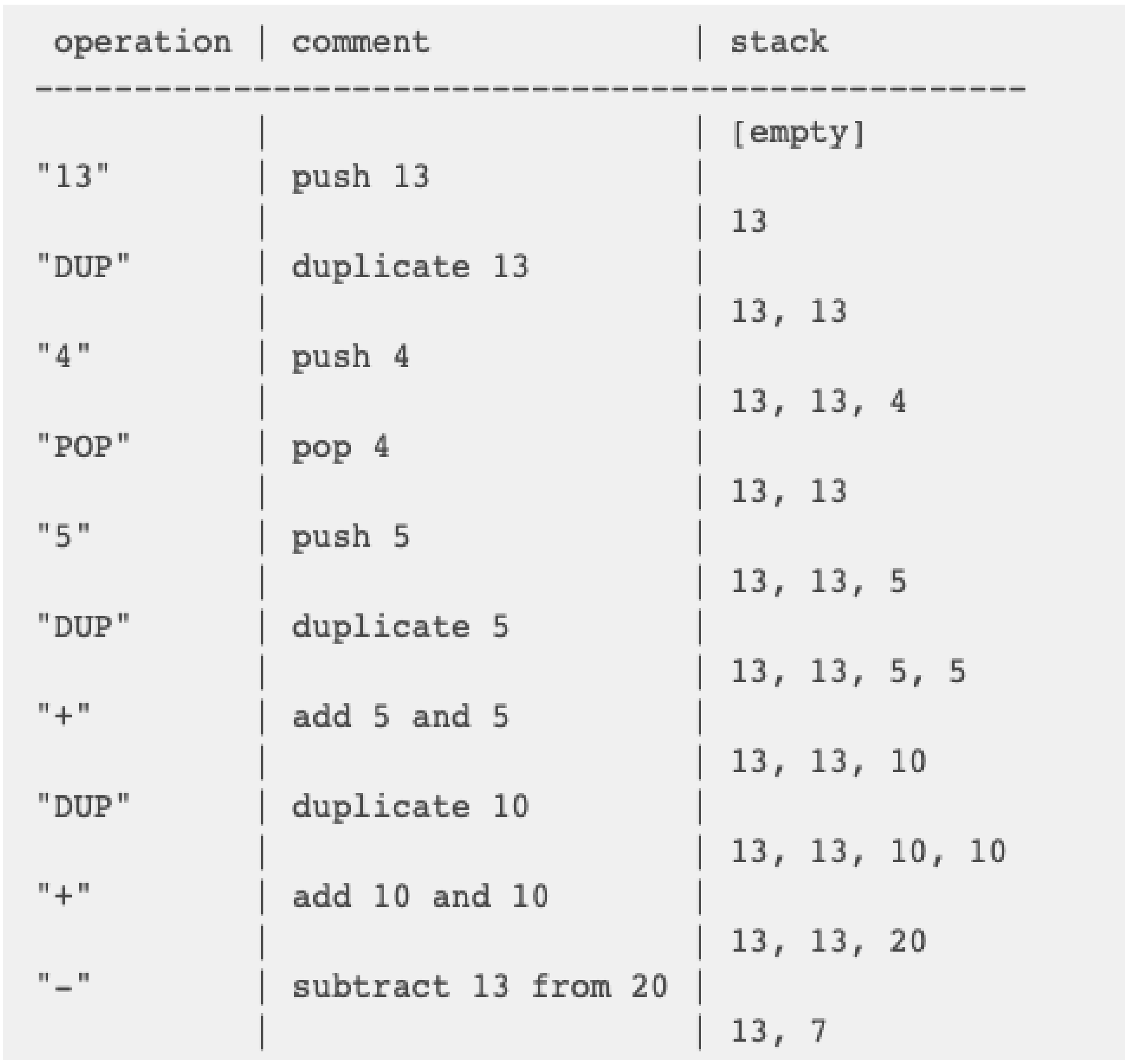
A word machine is a system that performs a sequence of simple operations on a stack of integers. Initially the stack is empty. The sequence of operations is given as a string. Operations are separated by single spaces. The following operations may be specified:

- an integer X (from 0 to 220 − 1): the machine pushes X onto the stack;
- "POP": the machine removes the topmost number from the stack;
- "DUP": the machine pushes a duplicate of the topmost number onto the stack;
- "+": the machine pops the two topmost elements from the stack, adds them together and pushes the sum onto the stack;
- "-": the machine pops the two topmost elements from the stack, subtracts the second one from the first (topmost) one and pushes the difference onto the stack.

After processing all the operations, the machine returns the topmost value from the stack.

The machine processes 20-bit unsigned integers (numbers from 0 to 220 − 1). An overflow in addition or underflow in subtraction causes an error. The machine also reports an error when it tries to perform an operation that expects more numbers on the stack than the stack actually contains. Also, if, after performing all the operations, the stack is empty, the machine reports an error.

For example, given a string "13 DUP 4 POP 5 DUP + DUP + -", the machine performs the following operations:



```
 operation | comment               | stack
--------------------------------------------------
           |                       | [empty]
"13"       | push 13               |
           |                       | 13
"DUP"      | duplicate 13          |
           |                       | 13, 13
"4"        | push 4                |
           |                       | 13, 13, 4
"POP"      | pop 4                 |
           |                       | 13, 13
"5"        | push 5                |
           |                       | 13, 13, 5
"DUP"      | duplicate 5           |
           |                       | 13, 13, 5, 5
"+"        | add 5 and 5           |
           |                       | 13, 13, 10
"DUP"      | duplicate 10          |
           |                       | 13, 13, 10, 10
"+"        | add 10 and 10         |
           |                       | 13, 13, 20
"-"        | subtract 13 from 20   |
           |                       | 13, 7
```

Finally, the machine will return 7.

Given a string "5 6 + -", the machine reports an error, since, after the addition, there is only one number on the stack, but the subtraction operation expects two.

Given a string "3 DUP 5 - -", the machine reports an error, since the second subtraction yields a negative result.

Write a function:

class Solution { public int solution(String S); }

that, given a string S containing a sequence of operations for the word machine, returns the result the machine would return after processing the operations. The function should return −1 if the machine would report an error while processing the operations.

For example, given string S = "13 DUP 4 POP 5 DUP + DUP + -" the function should return 7, as explained in the example above. Given string S = "5 6 + -" or S = "3 DUP 5 - -" the function should return −1.

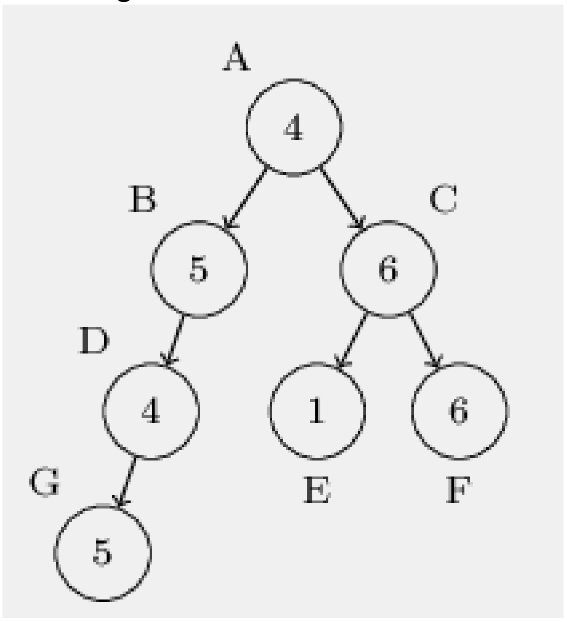
Assume that:

- the length of S is within the range [0..2,000];
- S is a valid sequence of word machine operations.

In your solution, focus on **correctness**. The performance of your solution will not be the focus of the assessment.

Task 4

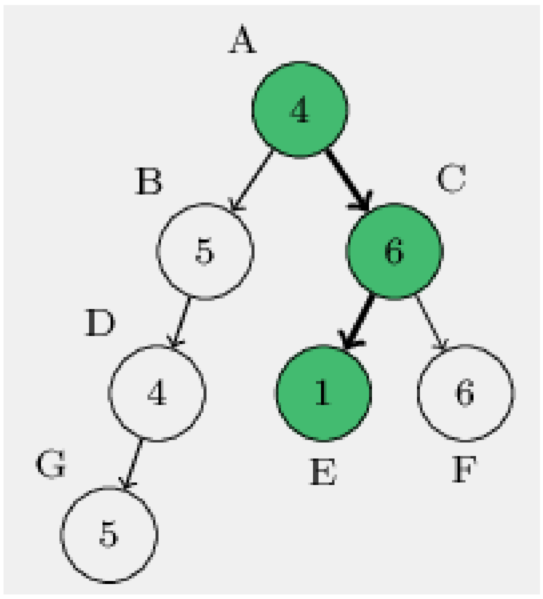
In this problem we consider binary trees. The figure below shows an example binary tree consisting of seven nodes.



A *binary tree* is either an empty tree or a node (called the *root*) containing a single integer value and linked to two further binary trees. We are interested in paths (sequences of linked adjacent nodes) that start at the root and follow the tree edges (marked as arrows in the figure above). For example, the sequence of nodes A, B, D is a valid path, but the sequence A, B, G is not.

Problem

We would like to find the maximum number of distinct values that appear on a path starting at the root of the tree. For example, on the path consisting of nodes A, B, D, G there are two distinct values (4 and 5). On the path A, C, E there are three distinct values (1, 4 and 6). There is no path that contains four or more distinct values.



Write a function:

class Solution { public int solution(Tree T); }

that, given a binary tree T consisting of N nodes, returns the maximum number of distinct values that appear on a path starting at the root of tree T. For example, given the tree shown above, the function should return 3.

Technical details

A binary tree is given using a pointer data structure. Assume that the following declarations are given:

```
class Tree {
  public int x;
  public Tree l;
  public Tree r;
}
```

An empty tree is represented by an empty pointer (denoted by null). A non-empty tree is represented by a pointer to an object representing its root. The attribute x holds the integer contained in the root, whereas attributes l and r hold the left and right subtrees of the binary tree, respectively.

Assumptions

Write an **efficient** algorithm for the following assumptions:

- N is an integer within the range [1..50,000];
- the height of tree T (number of edges on the longest path from root to leaf) is within the range [0..3,500];
- each value in tree T is an integer within the range [1..N].