Solve the Equation

Solve a given equation and return the value of `x` in the form of string "x=#value". The equation contains only '+', '-' operation, the variable `x` and its coefficient.

If there is no solution for the equation, return "No solution".

If there are infinite solutions for the equation, return "Infinite solutions".

If there is exactly one solution for the equation, we ensure that the value of `x` is an integer.

Example 1:

```
Input: "x+5-3+x=6+x-2"
Output: "x=2"
```

Example 2:

```
Input: "x=x"
Output: "Infinite solutions"
```

Example 3:

```
Input: "2x=x"
Output: "x=0"
```

Example 4:

```
Input: "2x+3x-6x=x+2"
Output: "x=-1"
```

Example 5:

```
Input: "x=x+2"
Output: "No solution"
```

Solution 1

```
public String solveEquation(String equation) {
    int[] res = evaluateExpression(equation.split("=")[0]),
     res2 = evaluateExpression(equation.split("=")[1]);
    res[0] -= res2[0];
    res[1] = res2[1] - res[1];
    if (res[0] == 0 && res[1] == 0) return "Infinite solutions";
    if (res[0] == 0) return "No solution";
    return "x=" + res[1]/res[0];
}

public int[] evaluateExpression(String exp) {
    String[] tokens = exp.split("(?=[-+])");
    int[] res =  new int[2];
    for (String token : tokens) {
        if (token.equals("+x") || token.equals("x")) res[0] += 1;
 else if (token.equals("-x")) res[0] -= 1;
 else if (token.contains("x")) res[0] += Integer.parseInt(token.substring(0, token.i
ndexOf("x")));
 else res[1] += Integer.parseInt(token);
    }
    return res;
}
```

written by compton_scatter original link here

Solution 2

```ruby
def solve_equation(equation)
  a, x = eval('i = 1i;' + equation.gsub('x', 'i').sub('=', '-(') + ')').rect
  "x=#{-a/x}" rescue a != 0 ? 'No solution' : 'Infinite solutions'
end
```

It's easy to change an equation like `2+3x=5x-7` to an expression for a complex number like `2+3i-(5i-7)`. Then I let Ruby evaluate that and it gives me the total of "x numbers" and the total of "non-x numbers" as the imaginary and real part of the complex number.

Same in Python:

```python
def solveEquation(self, equation):
    z = eval(equation.replace('x', 'j').replace('=', '-(') + ')', {'j': 1j})
    a, x = z.real, -z.imag
    return 'x=%d' % (a / x) if x else 'No solution' if a else 'Infinite solutions'
```

And here's a completely different one using a regular expression to parse the tokens:

```python
def solveEquation(self, equation):
    x = a = 0
    side = 1
    for eq, sign, num, isx in re.findall('(=)|([-+]?)(\d*)(x?)', equation):
        if eq:
            side = -1
        elif isx:
            x += side * int(sign + '1') * int(num or 1)
        elif num:
            a -= side * int(sign + num)
    return 'x=%d' % (a / x) if x else 'No solution' if a else 'Infinite solutions'
```

written by StefanPochmann original link here

Solution 3

The idea is to split the equation into 2 parts Left hand side(LHS) and Right hand Side(RHS) and then extract the coefficients and numbers from both the parts.

Coefficients are defined by the following regex
`'-?\d*x'`

`-?` - An optional negative sign
`\d*` - 0 or more digits
`x` - literal x

Regex for number:
`'-?\d*'` Again, this is as explained above.

An example
Before the program, here's a small example for the equation : `x+5-3+x=6+x-2`

LHS: `x+5-3+x`
RHS: `6+x-2`

LHS coefficients: `['x', 'x']`
RHS coefficients: `['x']`

Just the numbers on the LHS `['', '', '5', '-3', '', '', '']`
Just the numbers on the RHS `['6', '', '', '-2', '']`

LHS SUM of the numbers: `2`
RHS SUM of the numbers: `4`

LHS Coeff sum: `2`
RHS Coeff sum: `1`

Final Answer: `x=2`

Now the code

```
def solveEquation(self, equation):
        """
        :type equation: str
        :rtype: str
        """
        # define the patterns
        co_p = re.compile('-?\d*x')
        num_p = re.compile('-?\d*')

        # split the 2 parts of the equation.
        lhs = equation.split('=')[0]
        rhs = equation.split('=')[1]

        # find the list of coefficients with x(in both lhs and rhs)
        lhs_co = co_p.findall(lhs)
        rhs_co = co_p.findall(rhs)

        # find the list of all signed numbers after removing the coefficient
        # Notice that, I replace the coeffiecients with an '*' symbol to extract the
```

```
remaining numbers.
        # And I retrieve the numbers with the sign.
        lhs_num_list = num_p.findall(re.sub('-?\d*x', '*', lhs))
        rhs_num_list = num_p.findall(re.sub('-?\d*x', '*', rhs))

        # Now, simply add the numbers on the LHS and RHS
        lhs_sum = 0
        rhs_sum = 0

        for i in lhs_num_list:
            if i:
                lhs_sum += int(i)

        for i in rhs_num_list:
            if i:
                rhs_sum += int(i)

        # Add the Coefficients on both LHS and RHS.
        lhs_coeff_sum = 0
        rhs_coeff_sum = 0

        # While adding coefficients, 'x' and '-x' wil be replaced by 1 and -1 respec
tively.
        for i in lhs_co:
                i = i.replace('x', '')
                if not i:
                    i = 1
                elif i == '-':
                    i = -1
                lhs_coeff_sum += int(i)

        for i in rhs_co:
                i = i.replace('x', '')
                if not i:
                    i = 1
                elif i == '-':
                    i = -1
                rhs_coeff_sum += int(i)

        # Now, this is simple math and check the conditions to handle the edge cases
.
        if lhs_sum == rhs_sum and lhs_coeff_sum == rhs_coeff_sum:
            return "Infinite solutions"

        if lhs_coeff_sum == rhs_coeff_sum:
            return "No solution"
        result = (rhs_sum - lhs_sum) / (lhs_coeff_sum - rhs_coeff_sum)
        return "x=" + str(result)
```

Do let me know your feedback, Thanks!

written by aosingh original link here