S223 Pixel Math Puzzle

Problem idea by **Tony Wong**

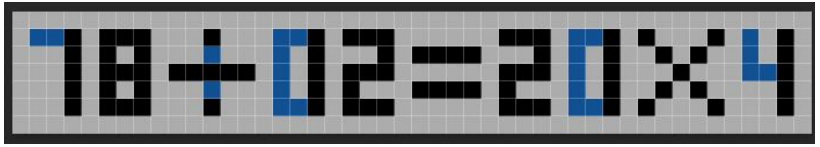
Problem set by **Tony Wong, Kevin Lee**

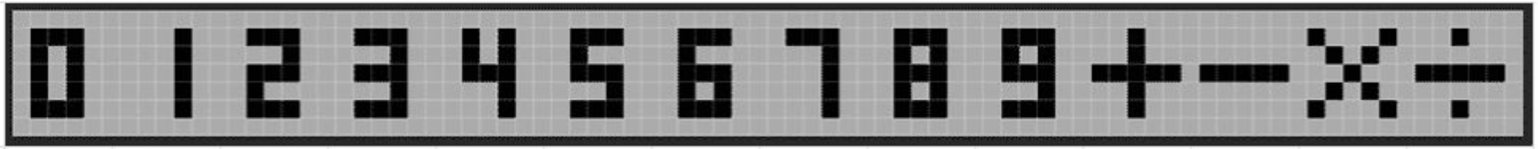
2022-01-29

香港電腦奧林匹克競賽
Hong Kong Olympiad in Informatics

Background

Add black pixels to a given pixel math equation such that the equation is valid



														
Digit/Symbol	0	1	2	3	4	5	6	7	8	9	+	-	*	/
Obtainable digits/symbols	08	0134789	28	389	489	5689	68	03789	8	89	+	+-/	*	+/

Subtasks

	Points	Constraints

1	13	`A` and `B` have exactly 1 digit
		`C` = `D` = `3`
		`o` = `+`
		`p` = `*`
		In other words, the puzzle is in the form of `?+?=3*3`.
		Hint: No valid solution will require changing the right hand side of the equation.

2	19	`A`, `B`, `C` and `D` have exactly 1 digit
3	22	`A`, `B`, `C` and `D` have at most 2 digits `o` and `p` are either `+` or `*`
4	21	`A`, `B`, `C` and `D` have at most 3 digits `o` and `p` are either `+` or `*`
5	25	`A`, `B`, `C` and `D` have at most 3 digits

SAMPLE TESTS

	Input	Output
1	`7+1=3*3`	`9+0=3*3`

This sample satisfies the constraints of subtask 1.
`8+1=3*3` and `0+9=3*3` are the only other acceptable solutions.

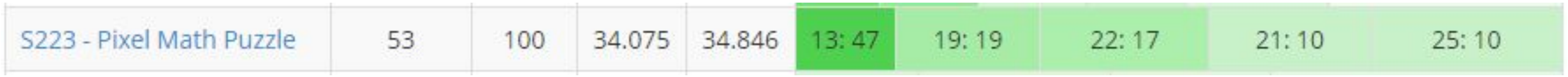
	Input	Output
2	`18/12=21*1`	`78+02=20*4`

Acceptable solutions include `08+12=20*1`, `48/02=24*1`, `18+42=20*3` and many others.
Note that `78+2=20*4` is not an acceptable solution.
This corresponds to the example in the problem statement.

	Input	Output
3	`1-1=80-240`	`1/3=80/240`
4	`20*21=12-18`	`Impossible`

香港電腦奧林匹克競賽
Hong Kong Olympiad in Informatics

Statistics



S223 - Pixel Math Puzzle	53	100	34.075	34.846	13: 47	19: 19	22: 17	21: 10	25: 10

First solved by **mtyeung1** at **0:30**

10 contestants got 100

General Approach

Divide the task into subtasks

- Generate (exhaust) all possible equations
  - `7+1=3*3 -> 9+0=3*3`
- Parse the equation
  - LHS operands: 9, 0 operator: +
  - RHS operands: 3, 3 operator: *
- Evaluate the expressions and compare their values
  - LHS: 9
  - RHS: 9

Generate (exhaust) all possible equations

Create a 2D boolean array `ok[10][10]`. `ok[i][j]` stores whether it's possible convert `i` to `j`

```
for (int i = 0; i <= 9; i++) {
  ok[i][i] = true;
  ok[i][8] = true;
}
ok[1][0] = ok[1][3] = ok[1][4] = ok[1][7] = ok[1][9] = true;
ok[3][9] = ok[4][9] = ok[5][6] = ok[5][9] = ok[7][0] = true;
ok[7][3] = ok[7][9] = true;
```

You can also map the operators `+-*/` to index 10-13. e.g. `ok[11][10]=true`

For subtask 1 and 2, you can use nested for-loops

Once the length of the equation is variable, recursion is needed

Using recursion to generate all possible equations

```
global input

global generated

function recur(int position) {
    if (position == input length) {     (base case)
        evaluate generated
    } else {
        for each letter l that can be converted from input[position] {
            append l to generated
            recur(position + 1)
            remove last letter from generated
        }
    }
}
```

Using recursion to generate all possible equations

Input: `123+456=78*90`

Generated: 12

```
recur(pos=2) {
    append 3 (generated=123) -> recur(pos=3) -> pop
    append 8 (generated=128) -> recur(pos=3) -> pop
    append 9 (generated=129) -> recur(pos=3) -> pop
}
```

Evaluate the equation

Please refer to

D108 - Simple Calculator

Time complexity

In case a digit can be converted to 10 other digit, and the total length of the string is N, then there could be 10^N possible expressions to evaluate

Here, the digit 1 has the most (7) digits to convert to

There are also two operators in the equation . – can be converted to 3 operators

Subtask 3:	A, B, C and D have at most 2 digits, o and p are either + or *	7^8 possible equations
Subtask 4:	A, B, C and D have at most 3 digits, o and p are either + or *	7^12 possible equations
Subtask 5:	A, B, C and D have at most 3 digits	7^12 x3x3 possible equations

Time complexity

Can we evaluate 7^8 equations in 0.5 seconds?
Yes -> Solves subtask 3

Can we evaluate 7^12 equations in 0.5 seconds?
No

Meet in the middle

Imagine that the input is 111-111=111*111

The first few equations we will evaluate are

```
000-000=000*000
000-000=000*001
000-000=000*003
```

You can see that the LHS is evaluated many times,
and they have the same value until the next one `000-001`

Meet in the middle

Strategy: generate possible expressions for LHS and RHS separately and evaluate their values

Check whether the set of possible values overlap.

Example: 8+5=9*2
LHS possible values: 13, 14, **16**, 17
RHS possible values: **16**, 18, 64, 72

Note 1: We need some way to store/restore which expression gives 16.
Note 2: If there are multiple expression that gives the same value, storing 1 of them is sufficient.

Meet in the middle

There can be up to 7^6x3 values for each side.

How to find common values efficiently?

Sort both arrays and match like merge sort (D805 Merging sub-arrays)

LHS possible values: 13, 14, **16**, 17
RHS possible values: **16**, 18, 64, 72

Handling division

Sample 3 has been provided to demonstrate that the result of the expression may not be an integer.

Input: `1-1=80-240`

Output: `1/3=80/240`

Division is much more difficult than the rest of the operators

- Fractional numbers
- Division by 0

Fractional Numbers

Approach 1:

Treat all the numbers as real number (e.g. `double`)

Precision error may occur: 2/5 != 4/10

- Whether this will actually happen depends on various factors such as CPU, OS and compiler.
- We have a solution and test case that fails on Codeforces, but it passes on HKOJ.

Fractional Numbers

Approach 2 (recommended):

Store a number using 2 integers p and q.

p/q where p and q are co-prime

Divide both numerator and denominator by their GCD

For example, 4/10 and 2/5 are both stored as p=2, q=5

Fractional Numbers

Approach 2 (recommended):

Store a number using 2 integers p and q.

p/q where p and q are co-prime

If we want to sort the possible values in order to find the common ones, we need to compare 2 fraction numbers

Method 1: p1/q1 < p2/q2 => p1*q2 < q1*p2

Method 2 that you can use for only this task: Compare p first, then q

Division by 0

When an integer divides by 0, you will get Runtime Error

How to reliably make contestants program crash?

- No special input required
- As long as there are no solutions and 0 is a possible number in the denominator, the program will consider 0 and tries to divide by 0.
- e.g. `8*8=0/0`

Division by 0

What about floating point data types?

1.0 / 0.0 will not cause runtime error, but the result is nan

But nan == nan and therefore

Input:	`8/0=88/0`
Expected output:	`Impossible`
Wrong output:	`8/0=88/0`

Other tricks

Don’t know recursion?

Randomly change the characters in the input until a solution is found.

Getting Time Limit Exceeded?

During exhaustion, check the run time regularly and
output Impossible when 0.49 seconds has been used.

Other notes

Cases like 1111-1111=1111-1111 are not practical, because there exist obvious solutions 8888+8888=8888+8888

In general, if the operands have the same length and both operators are one of +-/, then one solution is to convert all digits to 8 and all operators to +.

Therefore, to design cases without this obvious solution, we need to at least:

- Fix one of the operators to * (3x fewer combinations)
- Reduce the length of one operator (7x fewer combinations)

Also, if there are many solutions, then the previous random approach will be able to find any of them easily.

Overall, it is hard to find cases that have a lot of 1s but not many solutions.

Example: 611+915=111+111