code jam

System.out.println("hello, world!");

Practice Mode

Round 1B 2009

Contest Analysis

Questions asked 2

Submissions

Decision Tree

10pt	Not attempted **1512/1752 users** correct (86%)
11pt	Not attempted **1266/1544 users** correct (82%)

The Next Number

9pt	Not attempted **2559/3329 users** correct (77%)
26pt	Not attempted **1890/2557 users** correct (74%)

Square Math

12pt	Not attempted **157/422 users** correct (37%)
32pt	Not attempted **69/168 users** correct (41%)

Top Scores

ACRush	100
ftc	100
bmerry	100
andrewzta	100
ipknHama	100
halyavin	100
mystic	100
Yarin	100
Khuc.Anh.Tuan	100

Contest Analysis

Overview | Problem A | Problem B | Problem C

The problem would be a lot easier if there were only plus signs used. With the presence of the negative signs, a valid expression might get to a large value in the middle, then decrease back to the value we are looking for. Will that be the shortest answer for a certain query at all?

The best path cannot be long

First of all, let us denote a non-zero digit in the square *pos* if it has at least one neighbor that is a plus sign; call it *neg* if it has one minus sign as its neighbor. A digit can be both pos and neg. We assume there are both pos digits and neg digits in the square. Let **q** be the value we are looking for.
Let g be the gcd (greatest common divisor) of all the digits in the square. If g is not 1, in order to find a solution for q, it must be a multiple of g. Dividing everything by g, we may assume from now on that the gcd of all the numbers in the square is 1. The situation is further simplified when all our numbers are between 0 and 9 -- in this case, the above implies that there must be two digits, **a** and **b**, in the square, such that gcd(a, b) = 1.

Case 1. **a** is pos and **b** is neg. Take any shortest path **P** from **a** to **b** in the square. Let **q'** be the value of this path. q' is between [-200, 200]. Let t = q - q'. We prove the case t ≥ 0; the other case is similar and left to the readers.
Since gcd(a, b)=1, one of the numbers t, t+b, t+2b, ..., t+(a-1)b is a multiple of **a**. This means we can find non-negative numbers x and y such that ax - by = t, where y < a (therefore x < t/a + b). We start from **a**, since it is a positive digit, we use the plus sign to repeat at **a** x times, then follow **P**. After reaching **b**, we use the minus sign to repeat y times. The path just described evaluates to the query **q**.

Case 2. Both **a** and **b** are pos, there is a neg digit **c**. If **c** is co-prime with either **a** or **b**, we handle it as Case 1. Otherwise c has to be 6.
Pick any shortest path **P** that connects **a**, **b**, and **c**. Suppose it evaluates to q'. Pick a non-negative z such that q - q' + 6z ≥ ab - a - b + 1. We use a basic fact in number theory here

> If positive integers a and b are co-prime, then for any t ≥ ab - a - b + 1, there exist non-negative integers x and y such that ax + by = t.

To get an answer for query q, we use the path **P**, repeat **a** x times with plus sign, **b** y times with plus sign, and **c** z times with minus sign.

dgozman	100

Case 3. Both **a** and **b** are neg. This is similar to Case 2, and we leave it to the readers.

Thus we proved that for any solvable query, there is a path that is not long, as well as any partial sum on the path cannot be too big. The rough estimate shows that any of the paths cannot take more than 1000 steps. One can get a better bound by doing more careful analysis.

The algorithm

Our solution is a BFS (breadth first search). The search space consists all the tuples (r, c, v), where (r, c) is the position of a digit, and denote A(r, c, v) to be the best path that evaluates to v, and ends at the position (r, c). We know there are 100 such (r, c)'s, and at most (much less than) 20000 such v's from the bound we get.

The only difference with a standard BFS is that, because of the lexicographical requirement, we may need to update the answer on a node. But we never need to re-push it into the queue, since it is not yet expanded.