TopCoder Competitions

SRM 607

Added by [[rng_58]] , last edited by vexorian on Feb 08, 2014 (view change)
Labels: (None) EDIT

Archive
Match Overview
Discuss this match

Single Round Match 607
Monday, February 3rd, 2013

Match summary

SRM 607 was the first contribution of **Wheeler**. A problem set that focused on dynamic programming and clever optimization. The division 1 coders began their journey dealing with a classical string problem. Then they had to face the reality of an complicated problem that could become much simpler through observation. Finally, the beautiful division 1 hard that was a fractal of deeper and deeper algorithmic challenges. This problem wasn't solved by any of the coders, and **lyrically** was the only coder that was able to submit a solution at all. The division winners would be decided by the first two problems. Through marvellous speed at those problems, **tomek** got the first place with a relevant lead over other big names like **K.A.D.R** (2nd place), **Petr** (3rd place) and **tourist** (4th place). Congratulations also to division 2 winner: **enrevol**.

The Problems

BoundingBox | PalindromicSubstringsDiv2 | CombinationLockDiv2 | PalindromicSubstringsDiv1 | CombinationLockDiv1 | PulleyTautLine

CombinationLockDiv1

Rate It | Discuss it

Used as: Division One - Level Two:

Value	475
Submission Rate	173 / 720 (24.03%)
Success Rate	30 / 173 (17.34%)
High Score	**tourist** for 424.38 points (10 mins 3 secs)
Average Score	260.68 (for 30 correct submissions)

Let's use the solution for the division 2 version as a starting point. $O\left(n^3\right)$ is too heavy for this problem $(n \leq 2500)$. But we can optimize it with a few ideas.

Valid number of intervals

In the division 2 solution we use a loop to visit all possibilities of y and consider the ones that set the digit to 0. In reality, though modular arithmetic, it is easy to find a number t such that if we put t, $t+10$, or $t+20$, intervals through the digit the digit will be set to 0.

Closing or opening

Once t is known, we can reach a number of operations modulo t either by opening new intervals or closing them. The key to optimizing this problem is to notice that the number of new intervals or closed intervals should be as small as possible.

- If we are adding intervals, the number of intervals we add should be as small as possible. If adding a intervals is good enough for this digit, and we need more intervals for a future digit, we can always add them when that future digit is reached.
- If we are removing intervals, the number of removed intervals should also be as small as possible. This time it is clear, because if we remove more intervals than necessary for the current digit, a future

digit that needs more intervals would increase the cost without need.

The minimum number of intervals to add or remove, can be found through modular arithmetic and is between 0 and 9. Now each digit has 3 decision: Change direction, add intervals or remove intervals. We need $O(1)$ operations for each state of the $f(p,x,u)$ function. Effectively reducing the complexity to $O\left(n^2\right)$.

Code

Remember that there is a heavy constant factor in the current approach, because the number of open intervals is at most 9*n. We might need to optimize memory - which means we need iterative dynamic programming - and make sure to do crop as many states as possible when doing the dynamic programming.

```
static const int INF = 1000000000;
static const int MAX_N = 2500;
static const int MAX_OP = 2500 * 9;
vector<int> d;
int n;
int dp[2][MAX_OP + 1][2];


int minimumMoves(string s, string t) /* assume we already merged the strings together */
{
    n = s.size();
    d.resize(n);
    for (int i = 0; i < n; i++) {
        if (s[i] >= t[i]) {
            d[i] = s[i] - t[i];
        } else {
            d[i] = s[i] + 10 - t[i];
        }
    }


    for (int p = n; p >= 0; p--) {
        for (int x = 0; x <= p*9; x++) {
            for (int up = 0; up <= 1; up++) {

                // ignore invalid tuples (p,x,up), greatly cuts execution time
                int prev = (  (p == 0) ? 0 : d[p-1] );
                if ( (up == 1) && ( (prev + x) % 10 != 0) ) {
                    continue;
                }
                if ( (up == 0) && ( (prev + 9*x) % 10 != 0) ) {
                    continue;
                }
```

In fact, it appears that the maximum number of moves per digit is around 5000. If you can prove this, it can cut the constant factor enough to allow even memoization and a brute force search for the values of add and rem:

```
static const int INF = 1000000000;
static const int MAX_N = 2500;
static const int MAX_OP = 5500;
vector<int> d;
int n;
int dp[MAX_N + 1][MAX_OP + 1][2];

int rec(int p, int x, int up )
{
    int & res = dp[p][x][up];
    if (res == -1) {
        if (p == n) {
            res = 0;
        } else {
            res = INF;
            if (up == 1) {
                // previous step was up
                // try up:
                // (these search loops for correct add/rem value can be
                //  replaced by just a modular operation)
                for (int add = 0; add <= 9 && x + add <= MAX_OP; add++) {
                    if ( (d[p] + x + add) % 10 == 0) {
                        //we can !
                        res = std::min(res, add + rec(p + 1, x + add, 1) );
                    }
                }
                for (int rem = 0; rem <= 9 && rem <= x; rem++) {
                    if ( (d[p] + x - rem) % 10 == 0) {
                        //we can !
                        res = std::min(res, rec(p + 1, x - rem, 1) );
                    }
                }
                // try down:
                int z = d[p];
                res = std::min(res, z + rec(p + 1, z, 0) );
```

Alternative solutions and additional comments.

Take a look to the forum discussion about this problem, in which we also discuss alternative approaches: forum thread

<Place your comments here>

Next problem: PulleyTautLine

By **vexorian**

TopCoder Member

Editorial feedback	Choose
I liked it.	
I didn't like it.	

18 Comments | Add Comment