How to solve the 4-7-18 UT Dallas Programming Contest Problems.

A. What Started This?

A simple recurrence relation is given: $x_i = Ax_{i-1} + B$. Given x_i, A, B, and k, find x_0.

The trivial way is to rearrange the equation, $x_{i-1} = (x_i - B)/A$ and iterate this equation k times.

The 'more elegant' way is to solve the recurrence:

$$\begin{aligned} x_i &= Ax_{i-1} + B \\ &= A[Ax_{i-2} + B] + B \\ &= A^2x_{i-2} + AB + B \\ \ldots & \\ &= A^m x_{i-m} + B[1 + A + A^2 \ldots + A^{m-1}] \end{aligned}$$

When $m = k$, $x_{i-m} = x_0$ and $A^m = A^k$. Solving the GP gives:

$$\begin{aligned} x_k &= A^k x_0 + B[1 + A + A^2 \ldots + A^{k-1}] \\ x_k &= A^k x_0 + B\frac{(A^k - 1)}{(A - 1)} \\ x_0 &= \frac{[x_k - B\frac{(A^k-1)}{(A-1)}]}{A^k} \end{aligned}$$

B. Chanukah

This was intended to be the easiest problem:
The answer for each value of n is

$$n + n(n + 1)/2$$

C. Sum Squared Digits

Again, an easy problem:

```
int sumsqdig(int val, int base)
{
      int dig;
      int result = 0;
      while(val > 0) {
            dig = val % base; \\next digit to square and add = remainder of val/base
            result += dig*dig;
                  val = (val - dig)/base;
      }
      return result;
}
```

D. Less Than Me?

This one looks simple, but brute force will give Time Limit Exceeded. There is a clue as to the difficulty of this problem in the time limit of 30 seconds. Note also that there could be 10^6 array elements.

A clue to solving this problem is in the promise of uniform randomness of the input values. If we insert those values into a binary search tree (not necessarily a balanced tree), then the randomness of the data will tend to create balance in the tree.

Write your own tree class. It will only need a constructor and an insert function. You will enter values into the tree from the input array, taking array elements from left to right. The tree will respond to each input request with the final result for that cell. These are stored in a second array that is accessed to satisfy the queries.

The entire code in the main section just inserts values from the input array and records the results returned by the tree's insert function in the second array.

A tree node keeps track of the number of values inserted to its left in the tree.

Finally the queries are read and the answers are found in the second array at the index corresponding to the query value..

The runtime to insert N values is $O(N\,log\,N)$. That is the runtime to process each dataset.

E Uncle Rohi

Just one look at the sample outputs should make you wonder about the merits of brute force.

Consider a simple example where the start value is 4 and seven is to be added over and over until the total exceeds the range and overflows, then passes through the negative values, overflows again, etc.

Consider a simple version of the problem where the total is kept in a 4-bit nibble. The range of the nibble is $[-2^3, 2^3 - 1] = [-8, 7]$. A cycle starting at -8 and returning to that value takes $2^4 = 16$ steps. Starting at 4 and adding 7 twice gives a result of 2: (4 + 14) mod 16 = 2. Then (2+14) mod 16 = 0, Then (0+21) mod 16 = 5, then ((5+14) mod 16 = 3. Bingo!, The sequence of values was 4, 2, 0, 5, 3, four cycles. Thats 4*16 steps.

On each of these simple iterations we add the smallest number of sevens in order to make the sum complete one cycle and return to a non-negative value. We take the modulus, cycle length, of the value holding the result.

Note that each value in the sequence is 2 lower than the previous value, when the values are constrained in [0,6]. A little more math provides a simple equation for the number of cycles to get to the value 3.

The details are left to the reader.

F. Don't Eat The Dead Fish

A clear head and some trials on paper yields the fact that the player about to make a move will lose if the current total is a multiple of $(M+1)$. For example, if $N = 10$ and $M = 5$ this is a losing state for the player about to make a move. If $N = 9$ and $M = 5$, that player will take 4 M&Ms, leaving her opponent in a losing state.

G. Almost Pisano

To find a cycle we only have to compare two pairs of values: if $G_i = G_1 \ mod \, M$, $G_{i+1} = G_2 \ mod \, M$, $i > 1$ then $K(m) = i - 1$. But make sure to take the mod of G_1 and G_2 before starting to look for a cycle.

H Chomp

We discovered during the contest that, while the win/loss state of any cell is unique, the "best next move" is not. We therefore only compared the first part of every answer submitted with the first part of the answers in the dataset.

Think of those patterns (values of p, q, r) that are losing states. Start with one square $p, q, r = 2, 0, 0$) and build up from there.

I Counting Cows Again and Again

You might be tempted to use the Regular Expressions feature of Java, but the problem is simple enough to solve with easy coding. The String.split() function will yield an array of Strings, each String containing one 'group' of some number (greater than zero) of characters. If the group is larger than 5 characters the entire test case is rejected. Then simple code checks for illegal characters, and adds the number of correct groups, then adds the number of Is in the final group.

J Freddy's Maze Quest

Breadth First Search is easy to code and is the best way to solve this problem. An array keeps the smallest number of jumps to each cell. That number is fixed on the first occasion that BFS visits that cell. A boolean array keeps track of which cells have been visited. The algorithm does not enqueue cells that contain zeros or cells that have already been visited. It must also enforce the rule about staying in the maze.