2016 Rocky Mountain Regional Programming Contest

Solution Sketches

Credits

- Darko Aleksic
- Howard Cheng
- Zachary Friggstad
- Warren MacEvoy
- Per Austrin
- Sumudu Fernando

A - FizzBuzz (54/54)

- Straightforward - do exactly what you are asked.

I - Stacking Cups (48/50)

- Sorting problem.
- Trickiest part - reading input correctly.

- Optimally, try passwords in order of descending probability.
- Once sorted, result is $\sum_{i=1}^{n} i * p_i$

- As stated - stage j of swather i cannot start before stage j of swather $i - 1$ is completed.
- Solution is $ans_{i,j} = p_{i,j} + max(ans_{i,j-1}, ans_{i-1,j})$
- Easier to implement if we consider $(n+1)X(m+1)$ grid (no boundary checks needed)

F - Bumper-To-Bumper Traffic (18/32)

- Main observation: if $x_1 < x_2$, the only way the first car can catch up is if the second one is not moving.
- Simulate. Implementation may get tricky.
- Alternative - line sweep (yes, this was a geometry problem).

B - Election (15/43)

- Handle sure win/loss separately.
- If there are M remaining votes, and we need at least K votes to win, let $S = \sum_{i=K}^{M} \binom{M}{i}$
- Your candidate wins if $100 * S > W * 2^M$
- Use 64-bit integers.
- Floating point arithmetic may work - you avoid excessive multiplication and division.

H - Nice Numbers (3/8)

- Compress the numbers as much as you can.
- Insert 2 next to each remaining 2.
- Compress and repeat with 4's.
- Add 8's at the end until the sum is a power of 2.
- Hard part - keeping track of insertions relative to the original list.
- Lists of size 1 are already done.

J - Stack Construction (0/16)

- There will be exactly N prints, we have to optimize the number of push and pop operations
- For each character, after we print it, we either pop or push another on top (too slow).
- Dynamic programming.

J - Stack Construction (cont.)

- $a_{i,j}$ - the minimum number of push and pop operations to print the substring $S_{i,j}$
-

$$a_{i,j} = \begin{cases} 0 & \text{if } i > j \\ 2 & \text{if } i = j \\ \min_{i \le k < j}(2 + a_{i+1,k-1} + a_{k+1,j}) & i < j, s[i] = s[k] \end{cases}$$

- Solution: $N + a_{0,N-1}$

E - Studying for Exams (6/9)

- At each point of time study a course that gives you the most value (highest derivative)
- Look at the bounds and precision required and make it into a discrete problem
- Discrete version solvable with a standard dynamic programming algorithm.

E - Studying for Exams (cont.)

- Continous version - in the optimal soution, all non-zero functions will reach the same derivative z
- Binary search on z such that used times add up to T
- Do not go back in time! (Take care of $f'(t) = z : t < 0$)
- General solution - Lagrange multipliers

D - Delivering Goods(0/3)

- Input graph $G(V, E)$
- Calculate all shortest paths from depot and clients
- Build $G'(V', E')$ such that V' contains the depot and clients and $E' = \{(u, v) \in VxV : d(0, u) + d(u, v) = d(0, v)\}$
- G' is a DAG, so the answer is the minimum number of paths from 0 that cover all vertices in G'
- This is equivalent to the number of disjoint paths in G'

D - Delivering Goods (cont.)

- Build a bipartite graph $G''(V'', E'')$ such that each partition contains a copy of V'
- $E'' = \{(u, v) \in E' : u \in V'_{in}, v \in V'_{out}\}$
- König's theorem: G'' has a matching of size m if and only if there exists $n - m$ vertex-disjoint paths that cover each vertex in G', where n is the number of vertices in G'
- Use maximum flow (bipartite matching)