2018 Rocky Mountain Regional Programming Contest

Solution Sketches

Credits

- Howard Cheng
- Brandon Fuller
- Darcy Best
- Zachary Friggstad
- Christopher Painter-Wakefield
- Darko Aleksic
- Warren MacEvoy
- Greg Hamerly

- Simple, do what is asked of you.

D - H-Index (36/246)

- Find the maximum h such that there are at least h values that are at least h.
- Sort the array (largest to smallest)
- For each h (from 0 to n), the largest h numbers are at the beginning of the array, so just check if the h-th value is at least h.

- How many intervals contain a “special” house?
- Count the number of bad paths and subtract from total number of intervals.
- Look at the size of the gaps between special houses.
- If the gap has g houses, then the number of bad paths in that gap is $g(g+1)/2$.

H - Small Schedule (35/114)

- Bin packing with two sizes
- Fill with large ones, then with 1's
- Be careful of off-by-one errors

C - Forest for the Trees (14/86)

- Which trees can obstruct the view?
- $(k * x_b/g, k * y_b/g)$ where $g = gcd(x_b, y_b)$ is the next tree in the way.
- Look at the values of k such that the corresponding trees are in the rectangle.
- Can be done in constant time by looking at linear inequalities.

E - Driving Lanes (5/9)

- Dynamic programming
- Define $f(n, L) =$ the shortest distance to drive the first n straightaways and be in lane L.
- For each state, try changing to each of the other lanes (if possible).

I - Mr. Plow King (5/14)

- How to maximize the minimum spanning tree?
- Kruskal's algorithm tells us that we should greedily take the shortest edge if it doesn't create a cycle.
- You *must* include edges 1 and 2. The only way to not include edge 3 is to make a triangle (with 1, 2, 3).
- Then you *must* include edge 4. To not use edge 5 and 6, connect those edges into your connected component.
- Then you *must* include edge 7. Etc.
- At some point, you can use all remaining edges in the MST. Put them in a path.
- In general:
  - Make a giant connected component with lots of small edges.
  - Make a long path of large edges.

J - Rainbow Road Race (3/3)

- Travel through edges to get colors. Return to the start location with all colors with the shortest travel time.
- Create a new graph where the nodes are the pair: (location, set of colors you have).
- For each edge (from location i to j) create an edge from (i, S) to $(j, S \cup \{\text{color of edge}\})$ for all possible color sets S.
- Run a shortest path algorithm from $(1, \{\})$ to $(1, \{R, O, Y, G, B, I, V\})$.

B - Gwen's Gift (2/10)

- There are $(n-1)!$ valid sequences, for each first element there are $(n-2)!$ valid sequences and so on
- Keep track of the sums modulo n.
- For each position, go from 1 to $n-1$ skipping the *invalid* elements until you get to the desired index
- An element is invalid if (a) the prefix sum is 0 or (b) we have already seen the prefix sum (both mod n)
- $20! > 10^{18}$ - we care only about the last 20 elements, the rest are all 1's

- Another way to think about the problem: the partial sums of the array (modulo n) are non-zero and distinct.
- So the partial sums modulo N form a permutation of $1, 2, \ldots, n-1$.
- Generate the kth permutation (but order based on the original sequence).

F - Treasure Spotting (0/23)

- Relatively simple geometry, most cases covered in samples
- Line segment intersection
- Use integer arithmetic (do **not** use floating point)