2019 Rocky Mountain Regional Programming Contest

Solution Sketches

Credits

- Darko Aleksic
- Darcy Best
- Howard Cheng
- Zachary Friggstad
- Brandon Fuller

- The cake is cut into 4 pieces, pick the one with the maximum length for each side:

$$4 \cdot \max(a, n - a) \cdot \max(b, n - b)$$

K - Lost Lineup (67/68)

- $n = 1$, answer is 1
- Otherwise, permutation of numbers between 0 and $n - 2$
- Sort or find position one by one (small n), good enough even if it is $O(n^2)$.

- Too slow to count each pair one at a time
- Equivalence classes: count how many elements have the same quotient
- If there are k elements with the same quotient, then there are $k(k-1)/2$ pairs with the quotient
- You can use a map to count for each quotient, or sort the quotients
- Watch out for overflow!

I - Tired Terry (40/60)

- Sliding window of size p
- Update the count of “sleep” as we slide the window: look at the letter entering the window and leaving the window
- Easier if input string is duplicated to avoid wraparound
- Can be done in linear time.

B - Fantasy Draft (28/47)

- Just simulate one draft pick at a time...
- To do this under the time limit, you cannot afford to search the preference list every time
- Use a queue for each team: its preference with the global ranking appended
- Keep track of whether a player has been selected or not.

- Enumerate all $O(n^3)$ different triples of lines.
  For each triple:
  - Make sure no two are parallel (or coincide).
  - Compute the intersections of any two from the triple.
  - If they are distinct, add the distances between any two of them to get this triangle's perimeter.
- Mostly about getting the geometric details right.

C - Folding a Cube (9/28)

- The specification guarantees the six squares form a “tree”.
- So there is a unique way to try folding them into a cube.
- For any two distinct # squares i, j of the input, consider putting a “test” cube on square i and rolling it along # squares to square j.
  - If this would put the side initially on i face down on j, it is impossible to fold the cube.
  - If this never happens for any i, j pair of # squares, the folding is possible.
- So you have to track a side of the cube as it rolls around.

- Just doing naively it is too slow, the words can be too big. **Solution**: Hashing with polynomials.
- Think of each word $w := c_0 c_1 \dots c_{d-1}$ as a polynomial $w(x) := \sum_i c_i \cdot x^i$ where $c_i \equiv$ ASCII value.
- Pick a random integer $\overline{x}$ and compute each polynomial $w(\overline{x})$ mod p for a large prime p. This is our **hash** of w.
- Store partial sums $w_j(\overline{x}) := \sum_{i \leq j} c_i \cdot \overline{x}^i$ mod p and also the inverse of $\overline{x}$ mod p.
- Using arithmetic tricks, we can then compute the hash of w if we remove any single character c_i in $O(1)$ time.

- **Algorithm**
  - Store the hash of each dictionary word w in an set.
  - Try removing each c_i from each word w in the dictionary, if its hash was one stored in the last step, w is **probably a typo**.
- Since you have to output each typo anyway, you can also spend the time verifying it is indeed a typo (i.e. do the string checking if you see a hit).
- Can prove the **expected** running time is $O(\texttt{input size})$.
- **Why does this work?** Distinct polynomials of degree $< d$ will agree in at most d points even if we work mod p. So the probability of distinct strings of length $\leq d$ hashing to the same value is $\leq d/p$.

- You can do a simulatenous traversal on the two graphs, starting at the entrace at both graphs
- Follow the corresponding edges and keep track of the pair of rooms you are in for each graph
- If we ever arrive at a node such that the first component is the dormitory and the second component is not, the answer is no.
- Any graph traversal (e.g. breadth-first search) algorithm would work.
- Another view: both graphs are finite automaton. Is the language of the first automaton a subset of the other one?

F - Molecules (2/4)

- System of $2n$ equations with $2n$ unknowns: the x and y values of the points that are not fixed equal the average of their neighbours.
- Can prove there is a unique solution, given the assumption the molecule is connected and has at least one fixed point.
- Alternatively, just simulate.
  Place the unfixed points somewhere. Repeatedly, for each point compute the average of its neighbours and move *halfway* there. Converges close enough after a few thousand iterations (can prove this too).

J - Watch Later (1/23)

- Need to determine the order of types of video to watch.
- Once the order is fixed, the number of clicks to watch a particular type is the number of “chunks” of that type
- Use dynamic programming $O(2^n)$ states: what is the subset of types watched so far
- To be fast enough, need to be efficient in determining the number of chunks (can be done in linear time).