# Solution Sketches for Fall 2021 UW Local ICPC Contest

Troy Vasiga

October 3, 2021

# Problem E: Frogger

- Turn-by-turn simulation
- Maintain current state
- Could do some clever/efficient things, but not necessary

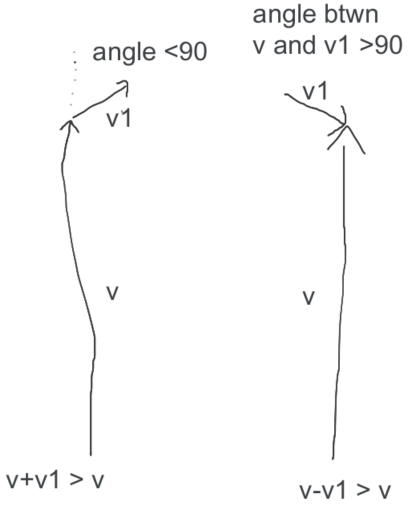
# Problem D: Not Long Enough

- Add the reverse of all the vectors to the set of vectors (i.e., “negate” all vectors)
- Sort the vectors by angles to ensure that they are considered in “roughly same direction” in order
- Add the vectors one at a time in sorted order, keeping track of the total of all vectors so far
- Why does this work? Proof by diagram.

# Problem D: Not Long Enough (cont'd)

- Consider the set of vectors $V$
- Call the maximum vector $v$, formed by vectors in set $M \subseteq V$.
- The algorithm considers vectors sorted by angle, so it will consider the ones whose angle is closer to the maximum vector $v$ together, away from other vectors whose angle is further away (and thus more likely to make $v$ shorter).
- Claim: any vector $m$ that is in $M$ should be included iff $m$ is within 90 degrees of the direction of $v$
- Proof by contradiction

# Problem D: Not long Enough (cont'd)



# Problem D: Not long Enough (cont'd)

- Suppose $v1$ is in $M$ and it's angle is more than 90 away from $v$.
- Removing $v1$ from $M$ will make $v$ even longer and contradict the maximality of $v$.
- Suppose $v2$ is not in $M$ and its angle is less than 90 away frm $v$.
- Adding it to $v$ would make $v$ longer, again contradicting the maximality of $v$.

# Problem C: Bus Connections

- Chinese Remainder Theorem
- Need some bigints
  - Use a reasonable language (i.e., not C++)
  - Build them yourself in C++

# Problem B: Noise

- Looks like a string matching problem, but KMP and suffix{automata/trie/arrays} will not help us (solutions with them will all be $\Omega(n^2)$).
- Instead we will use FFT (which, coincidentally, is also used to solve song-recognition in real life; although in a quite different way).
- Consider the polynomial

$$p(x, y) = (x - y)(x - y + 1)(x - y - 1).$$

  Note that $p(x, y) = 0$ iff $x = y$. We will thus use $p$ as a "comparison".
- Consider two arrays $A$ and $B$ of the same length, and we just want to check if they "match".
- They match iff $A_i \in [B_i - 1, B_i + 1]$ for all $i$, or equivalently if $p(A_i, B_i) = 0$ for all $i$.

# Problem B: Noise (cont'd)

- A first idea could be to check if $[0 = \sum_i p(A_i, B_i)]$, which is almost correct (when $A$ and $B$ "match" this sum is 0, but this sum can also be 0 otherwise). There are at least two ways to fix this:
  1. Consider $p^2$ instead, which is $= 0$ iff $x = y$ and strictly positive otherwise. Hence $[0 = \sum_i (p(A_i, B_i))^2]$ iff offset $x$ works.
  2. Add some random weights. That is we consider $[0 = \sum_i r_i * p(A_i, B_i)]$ where $r_i$ are independent random integers from say $[1, 1e9]$. This works with very high probability $(1 - 1/1e9)$.
- The model solution used (2), as it will in the end use fewer FFT calls.

# Problem B: Noise (cont'd)

- Now, if $A$ is longer than $B$, we want to calculate $[\sum_i r_i * p(A_{i+x}, B_i)]$ for all offsets $x$. Note that this looks like a convolution between $A$ and (a reversed) $B$. If we expand the product in the polynomial $p$, we will see that it suffices to calculate terms of the form:

$$\sum_i r_i A_{i+x}^p B_i^q$$

for some $p, q \leq 3$, and then sum them together.

- To do this we can simply calculate a convolution (with FFT) between $(A_i^p)$ and reversed $(r_i B_i^q)$. We need to do a total of 6 such convolutions (or a bit more for solution (1)). After we perform the 6 convolutions, we can simply sum the results together (with appropriate coefficients), and we have successfully calculated $[\sum_i r_i * p(A_{i+x}, B_i)]$ for all offsets $x$, which can be used to answer the problem.

# Problem B: Noise (cont'd)

- Implementation-wise, numbers get really large (around $(1e6)^4$), and we subtract them, so the solution is not at all numerically precise if we use normal FFT with floating points. But we can do everything in $Z_p$ for a suitable primes $p$ of size $1e9$, and then everything is exact.
- A similar idea can be used to solve "string matching with wildcards" where one uses $p(x, y) = (x - y)xy$ instead, so that $p(x, y) = 0$ iff $x = y$, or one of $x$ or $y = 0$ (0 is the wildcard value).

# Problem A: Mountain Skyline

- Basic trigonometry
- Sorting
- Intersection of a line and cone
  - geometry is full of edge cases
  - 3D geometry is more full of such edge cases
  - tricky since the line is not on a plane that is perpendicular to the axis of the cone
  - therefore, we cannot just project the cone as a triangle
  - need to solve some quadratic equations

# Problem A: Mountain Skyline

Why not just a 2d projection to a triangle?

- Consider cone with radius 2, with observer $2\sqrt{2}$ from base
- Altitude tangents form $2 - 2 - 2\sqrt{2}$ triangle
- Looking up to the cone at altitude 1, which has a circle of radius 1
- The triangle formed by this radius and tangent will have hypotenuse $2\sqrt{2}$ and one edge 1, which cannot be similar to the $2 - 2 - 2\sqrt{2}$ triangle
- Thus the cones "bulge out"
- Icky

# Next contests

- Winter 2022 Local contest: February (probably) 2022
- Spring 2022 Local contest: June (probably) 2022
- East Central North America Regionals: maybe November or maybe not? NADC? NAC?