Algolympics 2021

Solution Sketches

Kevin Atienza

Problem F: Virgo Coconut Oil

- Check if a date is between **September 16** and **October 30**, inclusive.
- Most of the info is unnecessary, ignore them!

Kevin Atienza

Problem F: Virgo Coconut Oil

- Solution 1:
  - Enumerate everything
  - Then find the index in the list
- Solution 2:
  - Enumerate only September and October

Kevin Atienza

Problem F: Virgo Coconut Oil

- Solution 3:
  - If month is September, check that day ≥ 16
  - If month is October, check that day ≤ 30
  - Otherwise, NO.

JD Dantes

Problem I: Major Constellation Assembly

- Vectors/lists are enough (e.g., `in` operator of Python).
- Can also use set data structure.
  - Better complexity (faster): Sets can handle “contains” efficiently.
  - You can also use **set difference** to get elements in one set not in the other set. Also, it’s efficient.

JD Dantes

Problem I: Major Constellation Assembly

- Python (**present** and **attendees** are sets)

```
print("STAR STREAM HACKED!!!"
      if present - attendees else
      "START THE MEETING."
      if len(present) * 2 >= len(attendees) else
      "NO STARS IN THE NIGHT SKY.")
```

JD Dantes

Problem I: Major Constellation Assembly

- In some languages (e.g., C++, Java), be careful of **integer division** which would result to **floored** values
- `if (something >= whole / 2)`
  - Be careful if `whole` is an integer. Can multiply instead:

```
if (something * 2 >= whole)
```

  - Alternatively, typecasting may work, but use of floating points is **discouraged** due to precision issues. Better to use the above approach.

```
if ((double) something >= (double) whole / 2)
```

Tim Dumol

Problem D: What a Crabulous Birthday

- For each cell:
  - For each direction:
    - Walk until you find another letter/digit.
    - Must not be adjacent.
- Then remove duplicates, sort, then print.
- Improvement: Only check down and right.
  - No more duplicates

Kevin Atienza

Problem E: Attack of the Cones

- Greedy insights
  - For the same flavor, better assign 1-scoopers before 3-scoopers
  - ...and 3-scoopers before 6-scoopers
  - For the same number of scoops, assign “flavor > 0” before “flavor = 0”.

Kevin Atienza

Problem E: Attack of the Cones

- Greedy insights
  - Better to assign “3-scoopers (flavor=0)” before 6-scoopers
  - ...and “1-scoopers (flavor=0)” before 3-scoopers

Kevin Atienza

Problem E: Attack of the Cones

- Using all insights, we now have determined the correct order to assign people!
- When assigning “flavor=0” people, just reserve enough scoops for them, not letting subsequent steps “take too much”.

Josh Quinto

Problem J: I used to be a musician, then I ...

- Try all possibilities up to 2 rows, then maximize on columns.
  - For each pair, update sums and find top 3 in $O(c)$ instead of $O(rc)$.
- So complexity is $O(r^2)O(c)$, or **$O(r^2c)$**.
- Do the same for columns ➔ rows. **$O(rc^2)$**
- Overall, **$O(rc(r + c))$**.

JD Dantes

Problem M: Crash Landing

- Linearity of expectation:
  - E[Hour1+Hour2+Hour3] = E[Hour1]+E[Hour2]+E[Hour3]
- Hour k is just Hour 1 but with p[i] replaced with $1 - (1 - p[i])^k$
  - 1 - p[i] = probability of component i surviving day 1
  - $(1 - p[i])^k$ = probability of component i surviving day k
  - $1 - (1 - p[i])^k$ = prob. of component i breaking on day k

JD Dantes

Problem M: Crash Landing

- Just need to know how to compute E[Hour1] now
- Bottom-up DP to compute E[Hour1]
- For every subtree, calculate both E[sum(l)] and $E[sum(l)^2]$
- E[Hour1] is then $E[sum(l)^2]$ of the root
- Linear time

Payton Yao

Problem L: The Trolley Problem: Solved

- “Eulerian path”
- There is a tour through all edges iff
  - There are 0 or 2 odd degree nodes.
  - **AND** all edges are connected.
- Gotcha: Ignore nodes without edges!

Payton Yao

Problem L: The Trolley Problem: Solved

- Each edge turns at most two odd nodes even.
- So if connected and k odd, then minimum is max(0, k/2-1).

Payton Yao

Problem L: The Trolley Problem: Solved

- If disconnected, need to add extra edges to connect them.
- If all components have an odd node, then still possible with max(0, k/2-1).
- Otherwise, for every component without odd node, need to add one more edge.
- Thus, max(0, k/2-1+(#comps without odd nodes))

Payton Yao

Problem L: The Trolley Problem: Solved

- "$i \geq 17$" constraint implies at most 16 components.
- DP on subsets of components.
- $O(3^{\#components})$

Patrick Celon

Problem G: Weighing Scales Heist

- BFS or DP
- Answer must start at S[0]/T[0], end at S[N-1]/T[N-1]
- For each (row,col) there are 2/4 ways to arrive
  - horizontal/vertical or up/down/left/right ➜ orientation
- P(orie, row, col, i)
  - Optimal passphrase starting at *(row, col)* with orientation *orie*
  - Containing subsequence (S[i]/T[i] ... S[N-1]/T[N-1])

Patrick Celon

Problem G: Weighing Scales Heist

- Generate the next level P(orie, row, col, i-1) from P(orie, row, col, i)
    - Check all P(orie, row, col, 0) for the shortest length
    - Naive BFS **O(4rc(r+c))** will cause TLE here
    - Generation can be done in **O(4rc)** per level
- Total complexity is now **O(4rcn)**
- Backtrack to output the optimal path

Patrick Celon

Problem G: Weighing Scales Heist

- Might need to compress data
  - Entire search space is stored for backtracking
  - 4*250*250*250 ≈ **64 million elements**
  - 4 ints (16 bytes) for length, orie, row, col etc.
    - Will use ≈ **1 GB total just for the search space**
  - Compress by using short/char or bitmasking
    - 8 bits for row,col
    - 1 or 2 bits for orientation/direction
    - 10-13 bits is enough for length

Kevin & JD

Problem K: Hot Sus Three

- Only need to assign people that weren’t reported
- For every person, call the person they reported their “parent”
- Start with an unassigned person, then go the parent, then the parent, etc., until you find a person with a fixed location, or you loop around without finding one

Kevin & JD

Problem K: Hot Sus Three

- If you find a fixed person, call that the “root”, and extract the tree of people rooted at that root.
  - Only include people that do not have fixed locations, and their fixed children (if any).

Kevin & JD

Problem K: Hot Sus Three

- Bottom-up DP:
  - **locations**[i] := set of possible locations of i with respect to its subtree.
  - **neighbors**[i] := set of all nodes in **locations**[i] and their neighbors.
  - DP: **locations**[i] = ∩ **neighbors**[j] for all children j of i
  - DP: **neighbors**[i] = ∪ ({k} ∪ **adj**[k]) for all k in **locations**[i]

Kevin & JD

Problem K: Hot Sus Three

- **locations**[i] and **neighbors**[i] can be represented as bitmasks.
- Assignment for this tree is possible iff the fixed location of root is in **locations**[root].
- You can then assign top-down; choosing *any* available node (consistent with parent) is ok.
- $O(cn^2/\text{wordsize})$

Kevin & JD

Problem K: Hot Sus Three

- If you loop without finding a fixed location, then choose a node in the cycle arbitrarily as the “root”, try all n possibilities for its location, and use the previous algorithm. If all fail, impossible.
- $O(cn^2/\text{wordsize}) \times O(n) = O(cn^3/\text{wordsize})$.

Kevin & JD

Problem K: Hot Sus Three

- Implementation tip: Defensive programming.
    - Try assigning with the algorithm above, and then check the validity at the end (even if you're "sure" it works).

Guissmo Asuncion

Problem A: The Complex War

- A(z) = B(z) ➔ A(z) - B(z) = 0
- A(z) - B(z) is a polynomial
- So the z's are common roots of A(z) - B(z) and C(z) - D(z)
- **Theorem**: r is a common root of p and q iff r is a root of gcd(p, q).
    - Proof via Factor thm. + Fundamental Thm. of Algebra

Guissmo Asuncion

Problem A: The Complex War

- So we need to find the gcd of two polynomials.
- **Euclid's algorithm**: Cancel the highest-degree term until one of them becomes the 0 polynomial.
- We have now reduced the problem to: Find the count and sum of roots of a single polynomial p.
- If p = 0, "INFINITE". Otherwise, finite.
  - Proof via Fundamental Theorem of Algebra

Guissmo Asuncion

Problem A: The Complex War

- The sum of roots of $az^n + bz^{n-1} + \ldots$ is **-b/a**.
  - Proof: Factorize to linear terms: $a(z - r_1)(z - r_2)\ldots(z - r_n)$, expand, then equate coefficients of z^{n-1}.
- Issue: -b/a is the sum *counting multiplicities*. We need to remove duplicates.

Guissmo Asuncion

Problem A: The Complex War

- **Theorem**: r is a repeated root of p iff r is a common root of p and p′ (its derivative).
  - Proof (⇒):
    - $p(z) = (z - r)^2 q(z)$
    - $\Rightarrow p'(z) = (z - r)(2q(z) + (z - r)q'(z))$
    - $\Rightarrow p'(r) = (r - r)(\ldots) = 0$.

Guissmo Asuncion

Problem A: The Complex War

- **Theorem**: r is a repeated root of p iff r is a common root of p and p′ (its derivative).
  - Proof (⇐):
    - $p(z) = (z - r)q(z)$ with $q(r) \neq 0$
    - $\Rightarrow p'(z) = q(z) + (z - r)q'(z)$
    - $\Rightarrow p'(r) = q(r) + (r - r)(\ldots) = q(r) \neq 0$.

Guissmo Asuncion

Problem A: The Complex War

- More general **theorem**: r is a root with multiplicity m of p iff r is a root of multiplicity m-1 in both p and p′. Similar proof.
- Thus, we just need to subtract the roots of gcd(p, p′)!

Guissmo Asuncion

Problem A: The Complex War

- *count_distinct_roots*(p) = *degree* p - *degree* gcd(p, p′).
- *sum_distinct_roots*(p) = *sum_roots*(p) - *sum_roots*(gcd(p, p′)).
- Need arbitrary precision integers and fractions. Or python.

Guissmo Asuncion

Problem A: The Complex War

- Also, parsing! Several options.
- Parsing Option 1: Use some infix➔postfix algorithm with a stack
- Parsing Option 2: Use some ad hoc/LL parser (recursion)

Guissmo Asuncion

Problem A: The Complex War

- Parsing Option 3: Python.

```
z = Polynomial(0, 1)

p = eval(input()) - eval(input())

q = eval(input()) - eval(input())

solve(p, q)
```

Kevin Atienza

Problem B: Orion Find

- If collinear, impossible. Otherwise, possible.
- Simplify!
  - Translate so that one point is origin.
  - Rotate so that one point is $(x, 0, 0)$ for $x > 0$.
  - Scale (evenly) so that it becomes $(1, 0, 0)$.
  - Rotate again so that 3rd point is $(x, y, 0)$ with $y > 0$.
  - Finally, rotate again to ensure $x \geq 1/2$.

Kevin Atienza

Problem B: Orion Find

- The three points are now (0, 0, 0), (1, 0, 0), (x, y, 0) with $y > 0$, $x \geq 1/2$.
- There should be a solution (X, Y, Z) with $X \leq 1/2$.

Kevin Atienza

Problem B: Orion Find

- Points: (0, 0, 0), (1, 0, 0), (x, y, 0) with $y > 0$, $x \geq 1/2$.
- Pick some viewpoint (X, Y, 0) with $X \leq 1/2$. Then rotate along x-axis until third point is perceived to be "equidistant" from the first two.
  - While rotating, the first two points stay fixed in the sky, while the third point will traverse a closed curve in the sky.

Kevin Atienza

Problem B: Orion Find

- Four things may happen:
  - You form a “tall” isosceles triangle
    - especially if $X < 0$, $Y \approx 0$
  - You form an equilateral triangle.
  - You form a “wide” isosceles triangle.
  - You can’t make the third point equidistant.

Kevin Atienza

Problem B: Orion Find

- Key: Starting from, say, $(-\varepsilon, 0, 0)$ to, say, $(1/2, 1/2, 0)$, you get those four possibilities in sequence.
- **Bisect** on that line segment!
- Complexity is (complexity of bisection)^2 because rotating the third point requires another bisection.

Kevin Atienza

Problem B: Orion Find

- The solution is not unique. You can probably do other iterative stuff.
  - Just be careful not to make the triangle converge to a degenerate “0 area equilateral triangle”. Note that the grading uses relative error, not absolute.

Kevin Atienza

Problem H: Scorpius Legs Flavor Inversion

- The sequence is bitonic: Increases then decreases (both strict).
- There will be several streaks of consecutive increasing integers (e.g., 5, 6, 7,...) or decreasing (e.g., 9, 8, 7, ...)
    - Can prove that there are at most **O(√n)** streaks per query.

Kevin Atienza

Problem H: Scorpius Legs Flavor Inversion

- Given k streaks, can compute the number of inversions in $O(k)$ time by considering the first and last streaks then recursing.
- Since $k = O(\sqrt{n})$, query time is $O(\sqrt{n})$, which is fast.
- $O(n + q\sqrt{n})$ time overall.

Karl Pilario

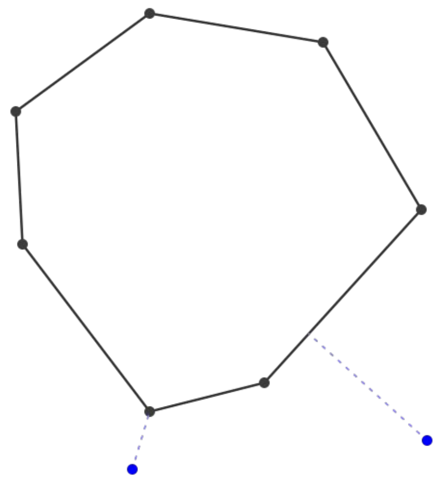
Problem C: Gem in Isaac

- “p is in the vicinity of C” is equivalent to *“p is in the convex hull of C”*
- Compute all c convex hulls.
  - Ignore edges
- “c” is small, so for each query point, just find distance to each hull separately.
- If we can do that quickly, we’re done.

Karl Pilario

Problem C: Gem in Isaac

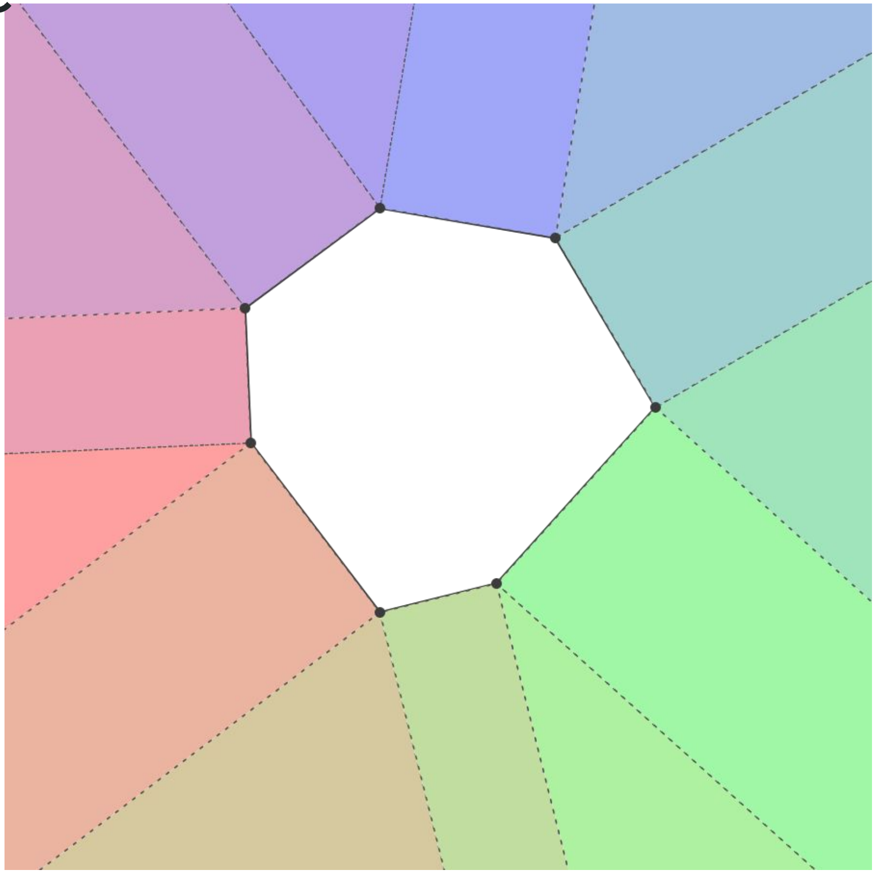
- The closest point to each hull is either a vertex or a side:



Karl Pilario

Problem C: Gem in Isaac

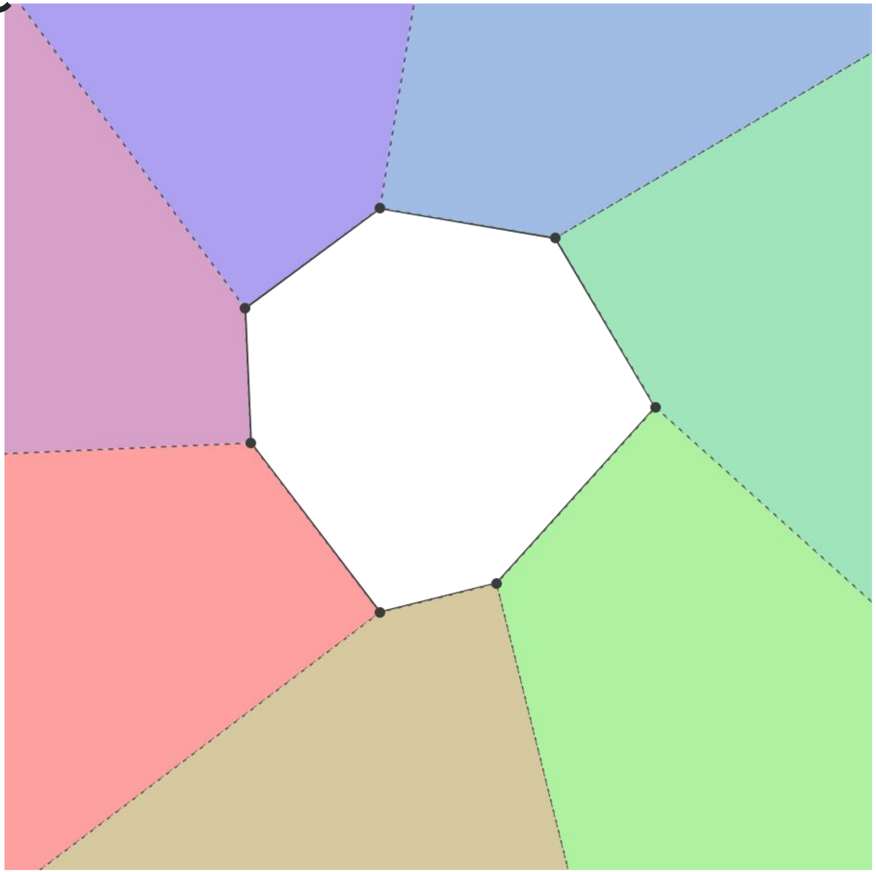
- Here are regions closest to each point and side.



Karl Pilario

Problem C: Gem in Isaac

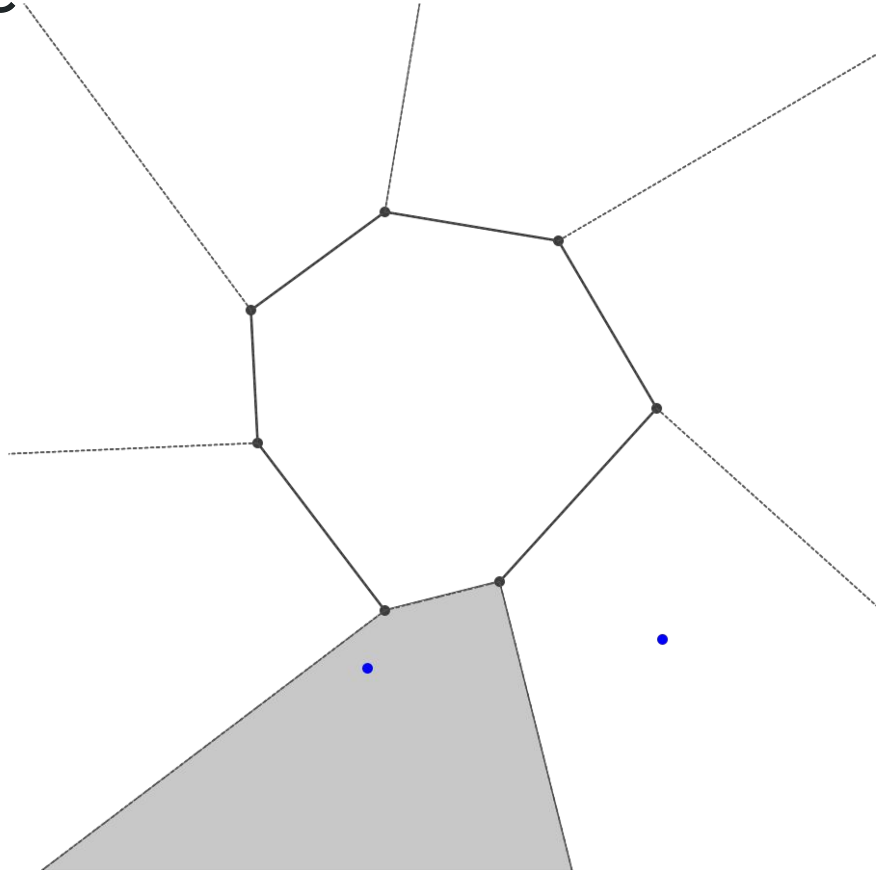
- Simplify by subsuming “closest to vertex” region to the corresponding “closest to side” region:



Karl Pilario

Problem C: Gem in Isaac

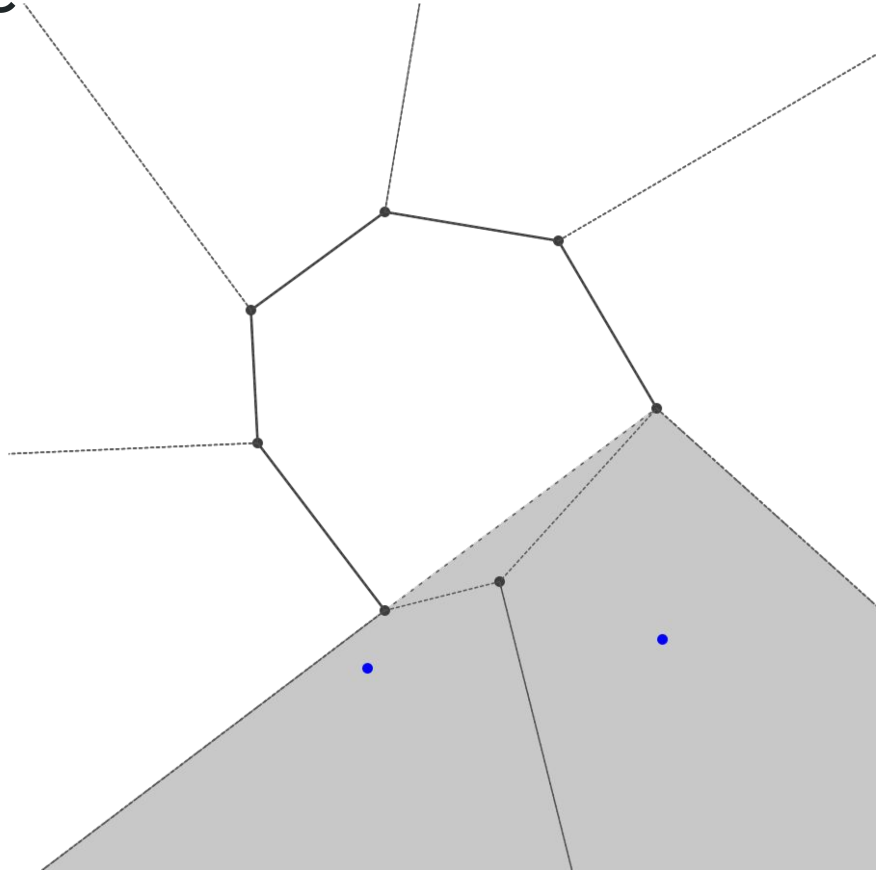
- To find which region a point belongs to, binary search on regions like these:



Karl Pilario

Problem C: Gem in Isaac

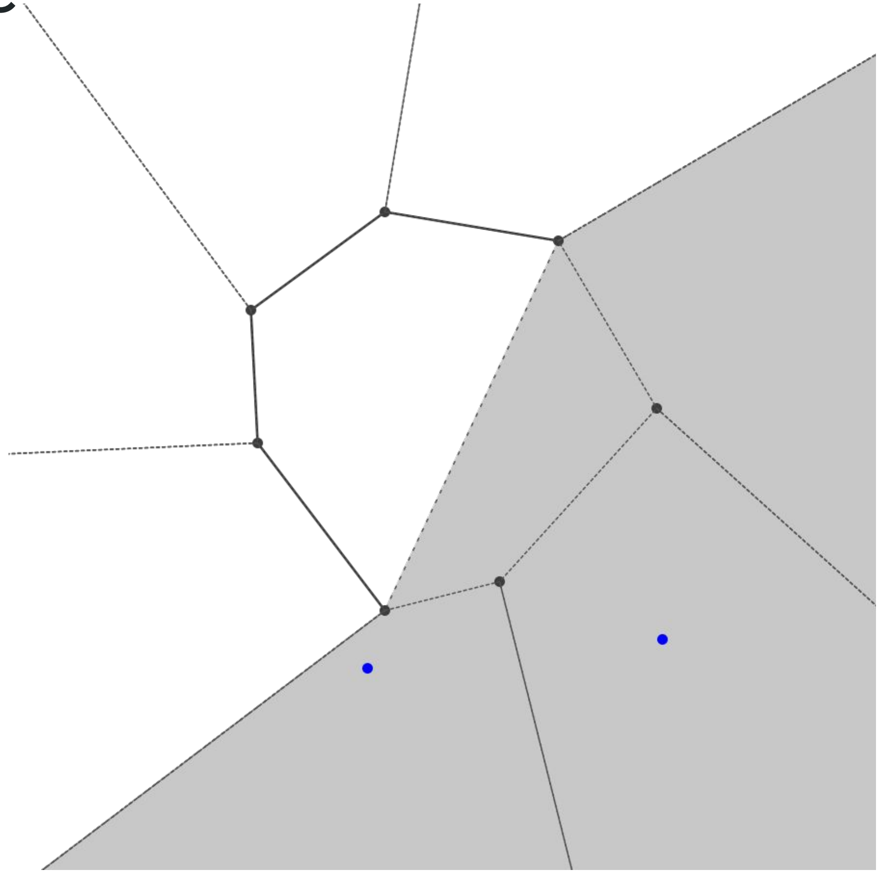
- To find which region a point belongs to, binary search on regions like these:



Karl Pilario

Problem C: Gem in Isaac

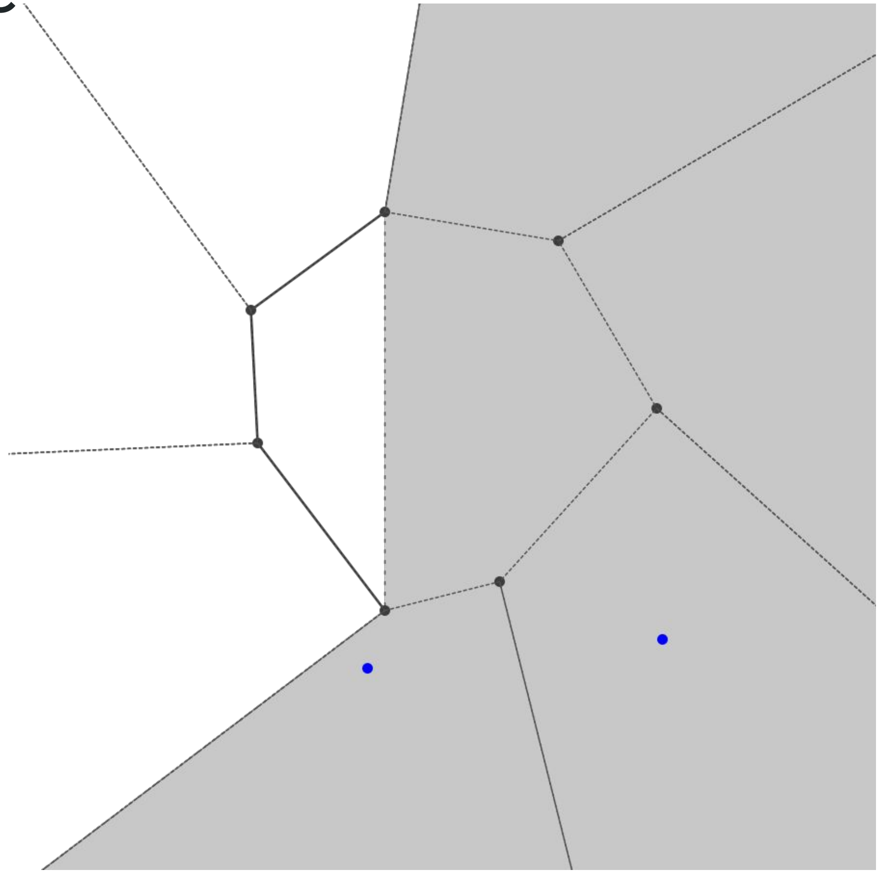
- To find which region a point belongs to, binary search on regions like these:



Karl Pilario

Problem C: Gem in Isaac

- To find which region a point belongs to, binary search on regions like these:



Karl Pilario

Problem C: Gem in Isaac

- Each such region is bounded by O(1) rays and segments, so can be checked against in O(1).
- Therefore, we can find the closest side in **O(log n)**!
- Overall O(n log n) to compute hulls and O(qc log n) to find distances.

Thank you!

- **Kevin Charles Atienza**
- **Joseph Daniel Dantes**
- **Marc Patrick Celon**
- **Rene Josiah Quinto**
- **Payton Robin Yao**
- **Tim Joseph Dumol**
- **Karl Ezra Pilario**
- **Jared Guissmo Asuncion**

- **A: The Complex War** - Asuncion
- **B: Orion Find** - Atienza
- **C: Gem in Isaac** - Pilario
- **D: What a Crabulous Birthday** - Dumol
- **E: Attack of the Cones** - Atienza
- **F: Virgo Coconut Oil** - Atienza
- **G: Weighing Scales Heist** - Celon
- **H: Scorpius Legs Flavor Inversion** - Atienza
- **I: Major Constellation Assembly** - Dantes
- **J: ...took an arrow to the knee** - Quinto
- **K: Hot Sus Three** - Atienza, Dantes
- **L: The Trolley Problem: Solved** - Yao
- **M: Crash Landing** - Dantes