

Day 9: Potluck Contest

Oleksandr Kulkov, Tähvend Uustalu

Osijek, Croatia

Problemsetters

- Arul Kolla (Codeforces: flamestorm)
- Kent Huang (Codeforces: ToxicPie9)
- Jeroen Op de Beek (Codeforces: jeroenodb)
- Tähvend Uustalu (Codeforces: -is-this-fft-)
- Oleksandr Kulkov (Codeforces: adamant)

A. Coprime Array

Statement

Given integers s and x up to 10^9.

Find $a_1 + \cdots + a_n = s$ s.t. $\gcd(a_i, x) = 1$ and $n \to \min$.

If $\gcd(s, x) = 1$, then $n = 1$.

If x is odd or s is even, $n = 2$ with $a_1 = d$ and $a_2 = s - d$, where

$$d \not\equiv 0, s \pmod{p}$$

for each prime divisor p of x (find by CRT).

If x is even and s is odd, use $a_3 = 1$ and reduce to the previous case.

B. Square Locator

Statement

For a square $ABCD$, given $|A - O|^2, \ldots, |D - O|^2$.

Find any $A, \ldots, D$ that fit the description.

Rotate the plane by 90° around A.

After the rotation, $|B - O'| = |D - O|$.

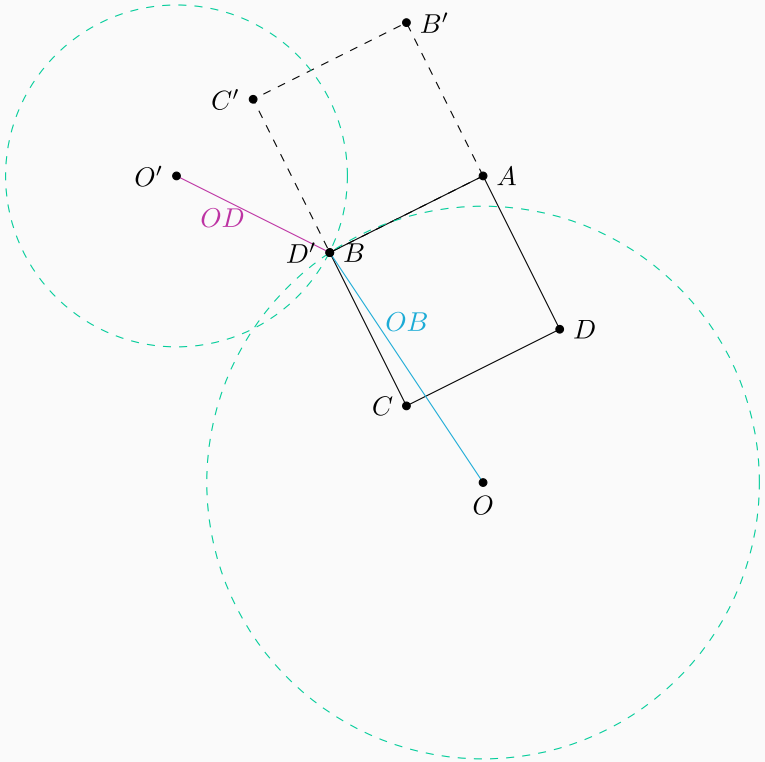
Intersect circles $|r - O'| = |D - O|$ and $|r - O| = |B - O|$ to find B.

B. Square Locator

Statement

For a square $ABCD$, given $|A - O|^2, \ldots, |D - O|^2$.

Find any $A, \ldots, D$ that fit the description.



Rotating the square and O around A

C. -is-this-bitset-

Statement

Given a rooted binary tree and arrays $a_1, \ldots, a_n$ and $b_1, \ldots, b_n$.

You can change at most 5000 elements a_i to anything you want.

For each i, find if there is a subset of ancestors of i that sums up to b_i.

Pick a number H and change a_i of i at height $h \leq H$ to 2^h.

if we're allowed to change k elements, we pick $H \approx \log_2 k$.

Then, the numbers obtainable at deeper i form intervals of length $\geq k$.

Use run-length encoding to take $O(A/k)$ memory instead of $O(A)$.

Maintain rollbacks to use $O(n + A)$ memory and $O(nA/k)$ time.

D. Cycle Game

Statement

A game is played on $n \times m$ white board.

In one turn, a single cell (r, c) is colored black.

Game is over if there is a black cycle with non-empty interior.

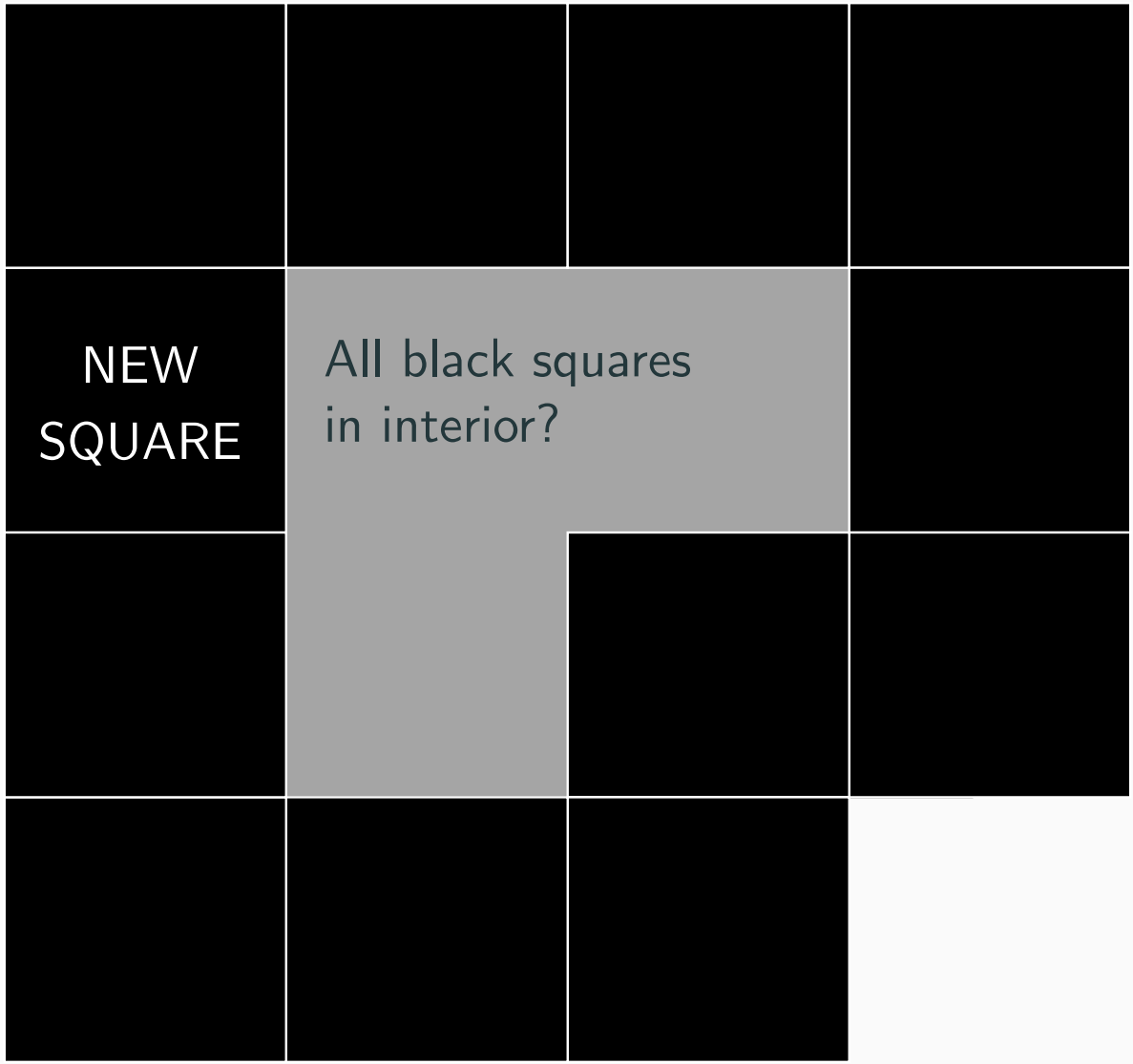
Given sequence of k turns, find which end the game (and skip them).

Consider cycle possibilities after adding a square.

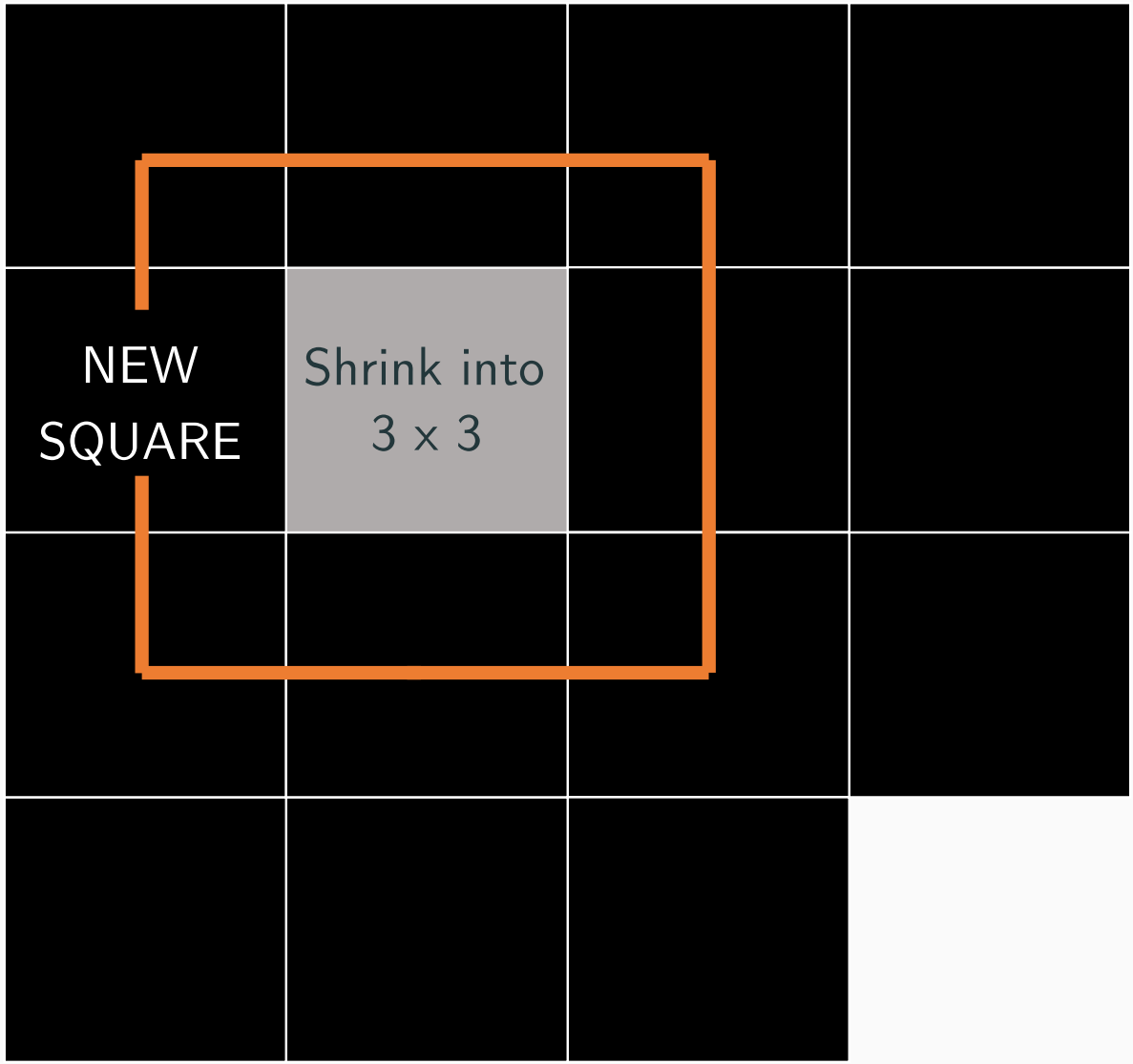
Are there any white squares within the interior?

Maintain black connected components for $O(nm + k)$ times DSU.

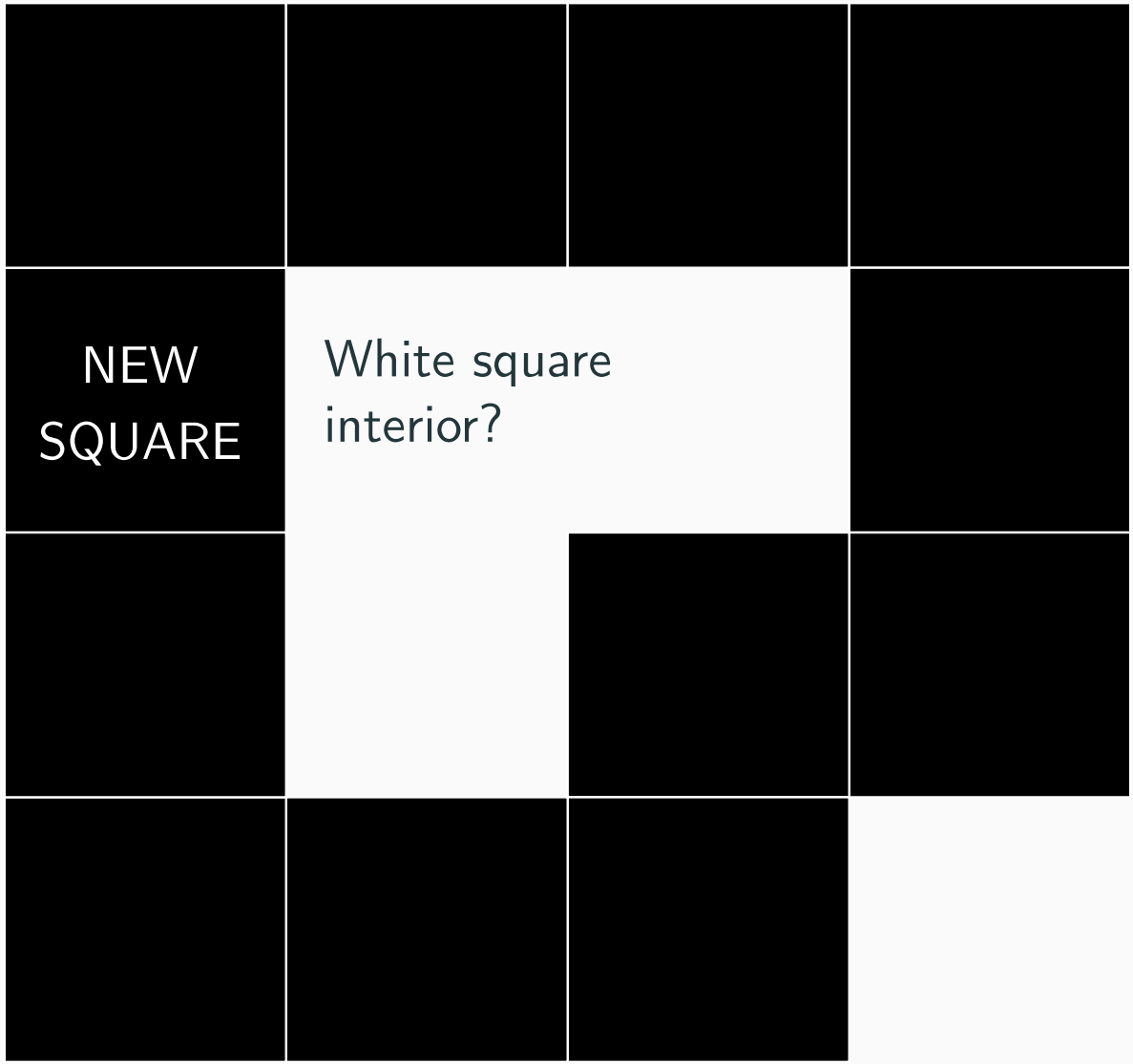
D. Cycle Game



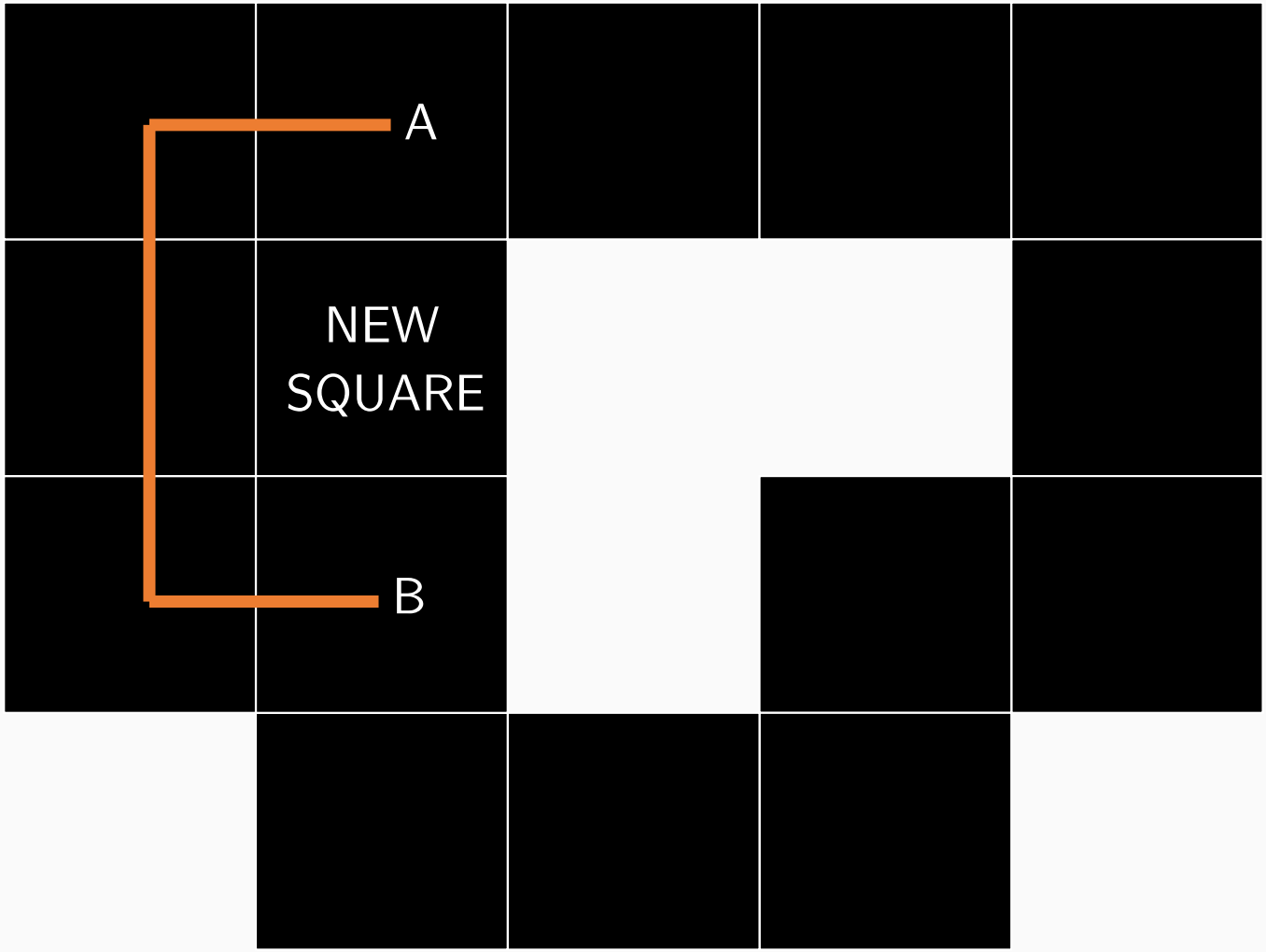
D. Cycle Game



D. Cycle Game

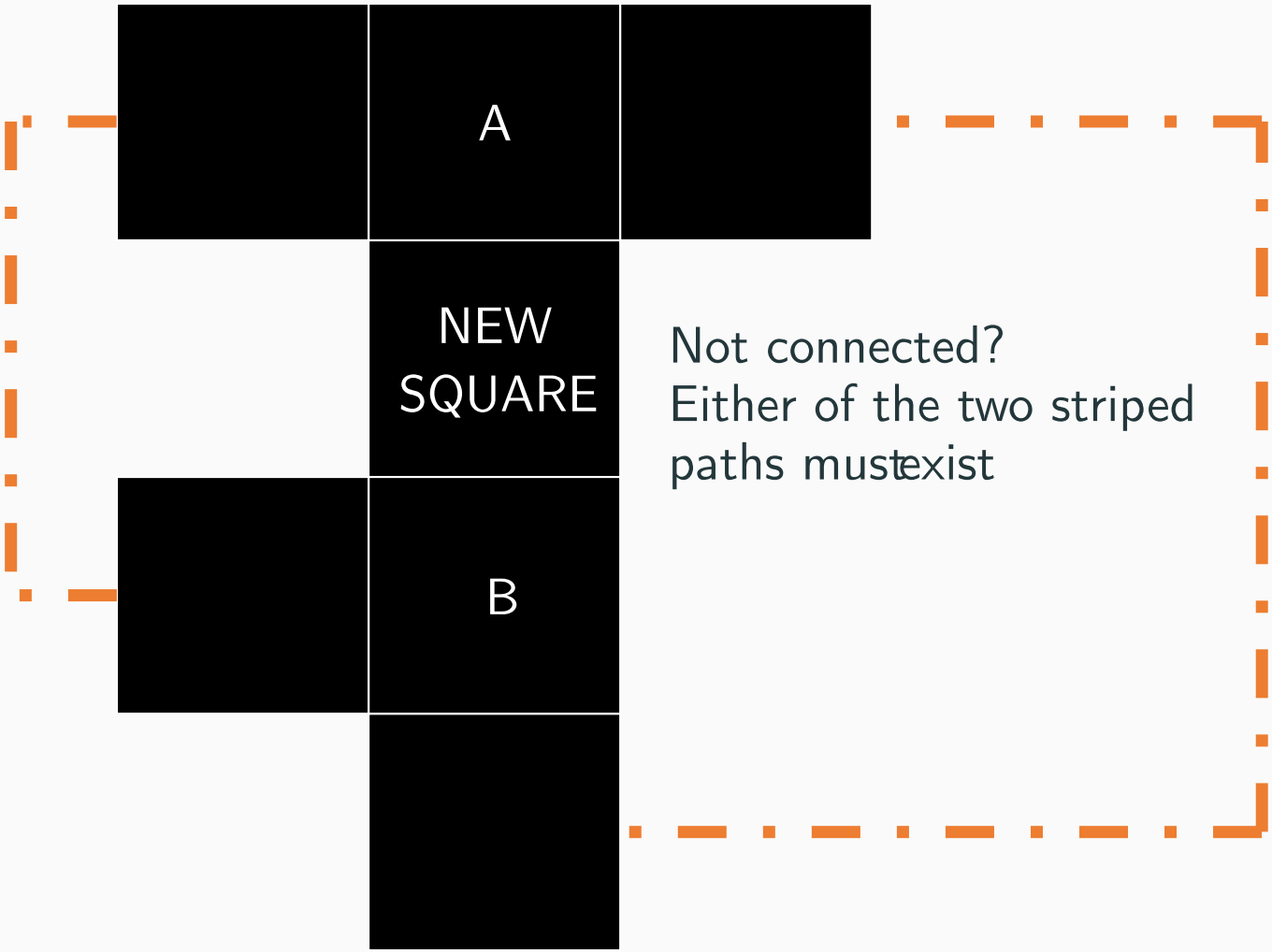


D. Cycle Game



Can go from A to B while "touching" new square?

D. Cycle Game


A
NEW
SQUARE
B
Not connected?
Either of the two striped
paths must exist

E. Sum of Squares

Statement

Given $A(x) = a_0 + \cdots + a_n x^n$.

Find the sum of squares of coefficients of $\prod_{i=1}^{m} A(x_i) \prod_{j=1}^{i-1} (x_i - x_j)$.

Represent it as the Vandermonde determinant:

$$A(x_1) \dots A(x_m) \begin{vmatrix} 1 & x_1 & \cdots & x_1^{m-1} \\ 1 & x_2 & \cdots & x_2^{m-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_m & \cdots & x_m^{m-1} \end{vmatrix}$$

E. Sum of Squares

Statement

Given $A(x) = a_0 + \cdots + a_n x^n$.

Find the sum of squares of coefficients of $\prod_{i=1}^{m} A(x_i) \prod_{j=1}^{i-1} (x_i - x_j)$.

Get $A(x_1), \ldots, A(x_m)$ inside the determinant:

$$\begin{vmatrix} A(x_1) & x_1 A(x_1) & \cdots & x_1^{m-1} A(x_1) \\ A(x_2) & x_2 A(x_2) & \cdots & x_2^{m-1} A(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ A(x_m) & x_m A(x_m) & \cdots & x_m^{m-1} A(x_m) \end{vmatrix}$$

Coefficients correspond to the exterior product:

$$A(x) \wedge xA(x) \wedge \cdots \wedge x^{m-1} A(x)$$

E. Sum of Squares

Statement

Given $A(x) = a_0 + \cdots + a_n x^n$.

Find the sum of squares of all coefficients of $\prod_{i=1}^{m} A(x_i) \prod_{j=1}^{i-1} (x_i - x_j)$.

Square length of exterior product $\iff$ Gramian determinant:

$$\|A_1 \wedge \cdots \wedge A_m\|^2 = \begin{vmatrix} A_1 \cdot A_1 & A_1 \cdot A_2 & \cdots & A_1 \cdot A_m \\ A_2 \cdot A_1 & A_2 \cdot A_2 & \cdots & A_2 \cdot A_m \\ \vdots & \vdots & \ddots & \vdots \\ A_m \cdot A_1 & A_m \cdot A_2 & \cdots & A_m \cdot A_m \end{vmatrix}$$

Multiply by $m!$ for the sum of squared coeffs.

E. Sum of Squares

Statement

Given $A(x) = a_0 + \cdots + a_n x^n$.

Find the sum of squares of all coefficients of $\prod_{i=1}^{m} A(x_i) \prod_{j=1}^{i-1} (x_i - x_j)$.

Need to find the determinant of $T \in \mathbb{R}^{m \times m}$, where

$$T_{ij} = [x^{i-j}]A(x)A(x^{-1}) = t_{|i-j|}.$$

It is a banded Toeplitz matrix:

$$T = \begin{bmatrix} t_0 & \cdots & t_n & & \\ \vdots & \ddots & & \ddots & \\ t_n & & t_0 & & t_n \\ & \ddots & & \ddots & \vdots \\ & & t_n & \cdots & t_0 \end{bmatrix}$$

E. Sum of Squares

Statement

Given $A(x) = a_0 + \cdots + a_n x^n$.

Find the sum of squares of all coefficients of $\prod_{i=1}^{m} A(x_i) \prod_{j=1}^{i-1} (x_i - x_j)$.

Let x be an eigenvector of T with eigenvalue λ:

$$\begin{bmatrix} t_0 - \lambda & \cdots & t_n & & \\ \vdots & \ddots & & \ddots & \\ t_n & & t_0 - \lambda & & t_n \\ & \ddots & & \ddots & \vdots \\ & & t_n & \cdots & t_0 - \lambda \end{bmatrix} \begin{bmatrix} x_0 \\ \vdots \\ \vdots \\ \vdots \\ x_{m-1} \end{bmatrix} = 0$$

E. Sum of Squares

Embed the system into a larger one:

$$\left[\begin{array}{ccc|ccccc|ccc} t_n & \cdots & & t_0-\lambda & \cdots & t_n & & & & & \\ & \ddots & & \vdots & \ddots & & \ddots & & & & \\ & & & t_n & & t_0-\lambda & & t_n & & & \\ & & & & \ddots & & \ddots & \vdots & \ddots & & \\ & & & & & t_n & \cdots & t_0-\lambda & \cdots & & t_n \end{array}\right] \left[\begin{array}{c} x_{-n} \\ \vdots \\ \hline x_0 \\ \vdots \\ \vdots \\ \vdots \\ x_{m-1} \\ \hline \vdots \\ x_{m+n-1} \end{array}\right] = 0$$

With extra constraint $x_{-n} = \cdots = x_{-1} = x_m = \cdots = x_{m+n-1} = 0$.

It is a linear recurrence of degree $2n$ with $2n$ boundary conditions!

E. Sum of Squares

Statement

Given $A(x) = a_0 + \cdots + a_n x^n$.

Find the sum of squares of all coefficients of $\prod_{i=1}^{m} A(x_i) \prod_{j=1}^{i-1} (x_i - x_j)$.

Let $L : \mathbb{K}[x] \to \mathbb{K}$ be a linear operator s.t. $L(x^k) = x_{k-n}$.

Then, boundary conditions are

$$L(x^0) = \cdots = L(x^{n-1}) = L(x^{m+n}) = \cdots = L(x^{m+2n-1}) = 0$$

Note that $L(P) = L(P \bmod C)$, where

$$C(x) = t_n + t_{n-1}x + \cdots + (t_0 - \lambda)x^n + \cdots + t_{n-1}x^{2n} + t_n x^{2n+1}$$

is the characteristic polynomial of the recurrence.

E. Sum of Squares

Statement

Given $A(x) = a_0 + \cdots + a_n x^n$.

Find the sum of squares of all coefficients of $\prod_{i=1}^{m} A(x_i) \prod_{j=1}^{i-1} (x_i - x_j)$.

x is an eigenvector of T with an eigenvalue λ if and only if

$$\begin{bmatrix} x^0 \bmod C(x) \\ \vdots \\ x^{n-1} \bmod C(x) \\ x^{m+n} \bmod C(x) \\ \vdots \\ x^{m+2n-1} \bmod C(x) \end{bmatrix} \begin{bmatrix} x_{-n} \\ \vdots \\ x_{-1} \\ x_0 \\ \vdots \\ x_{n-1} \end{bmatrix} = 0$$

E. Sum of Squares

Statement

Given $A(x) = a_0 + \cdots + a_n x^n$.

Find the sum of squares of all coefficients of $\prod_{i=1}^{m} A(x_i) \prod_{j=1}^{i-1} (x_i - x_j)$.

The system rewrites in a block form:

$$\begin{bmatrix} I_n & 0 \\ U_\lambda & V_\lambda \end{bmatrix} \mathrm{x} = 0$$

The system has a non-trivial solution if and only if V_λ is singular:

$$\det(T - \lambda I) = 0 \iff \det V_\lambda = 0.$$

In fact, they have the same degree as polynomials in λ:

$$\det(T - \lambda I) = (-1)^{mn} t_n^m \det V_\lambda.$$

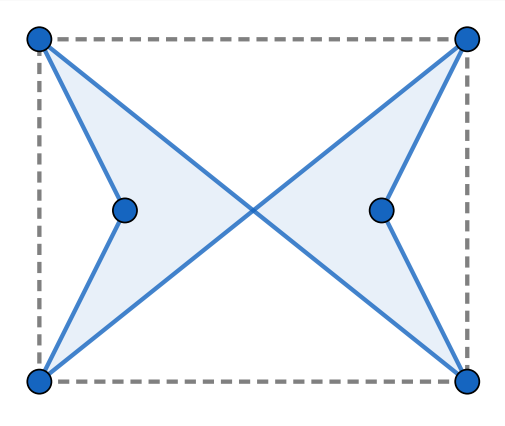
Find $\det T = (-1)^{mn} t_n^m \det V_0$ in $O(n^2 \log m + n^3)$.

F. Alternating Cycle

Statement

You're given n points in general position on the plane.

Find a shortest cycle which angles alternate (counter-)clockwise.



Conjecture

If ≤ 1 points inside the convex hull, impossible.

If $= 2$ points inside the convex hull, the only possible shape is butterfly.

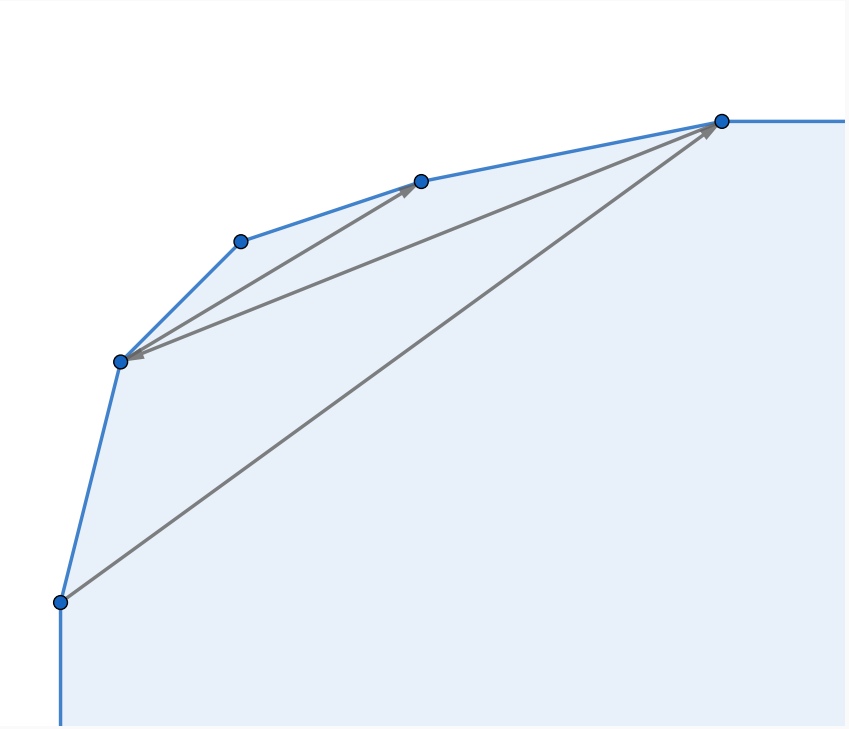
If ≥ 3 points inside the convex hull, there is a cycle of length 6.

F. Alternating Cycle

Statement

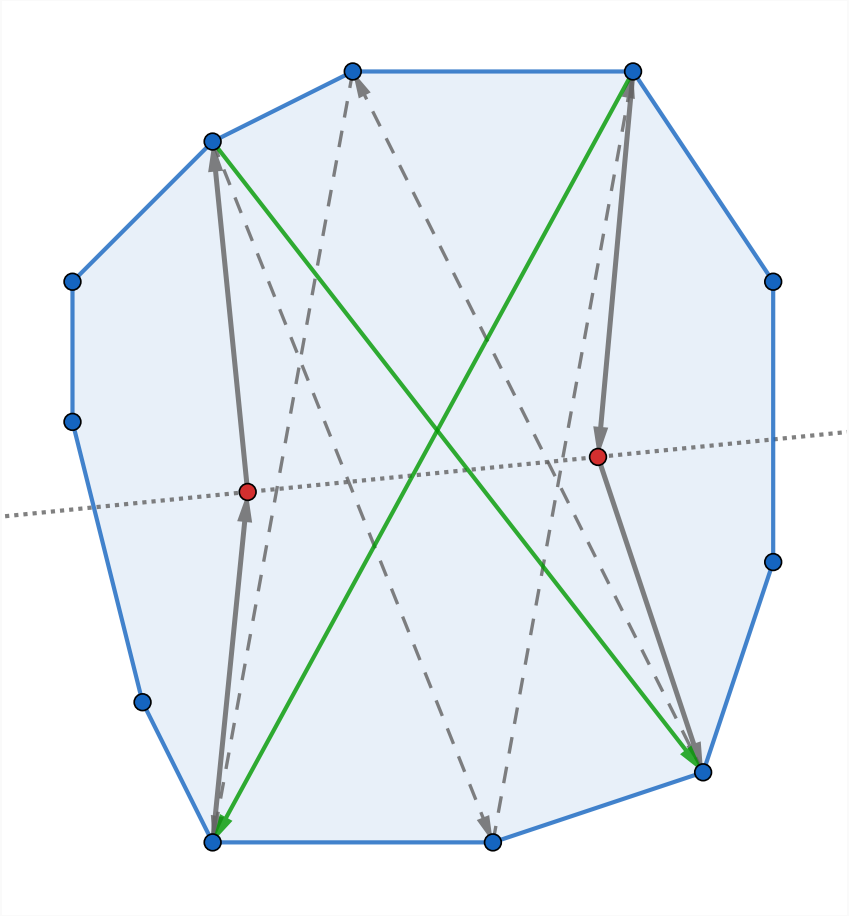
You're given n points in general position on the plane.

Find a shortest cycle which angles alternate (counter-)clockwise.



For ≤ 1 interior points, we get a dead end.

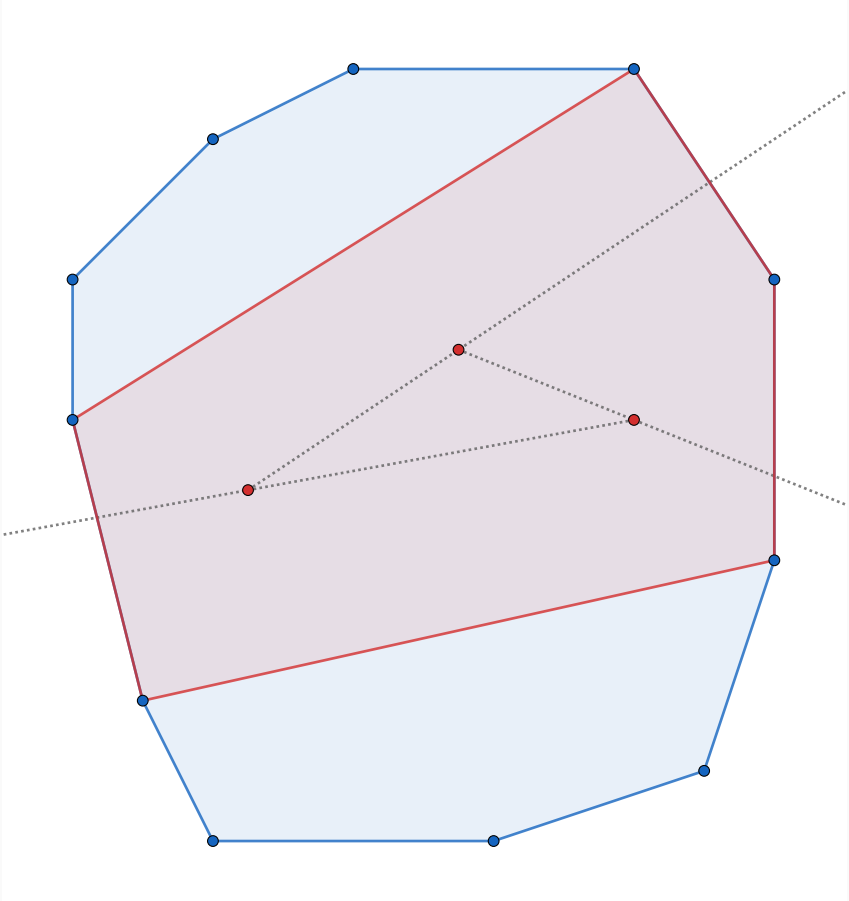
F. Alternating Cycle



For 2 interior points, contract alternating cycle into butterfly.

Find min/max angles with 2 pointers on convex hull.

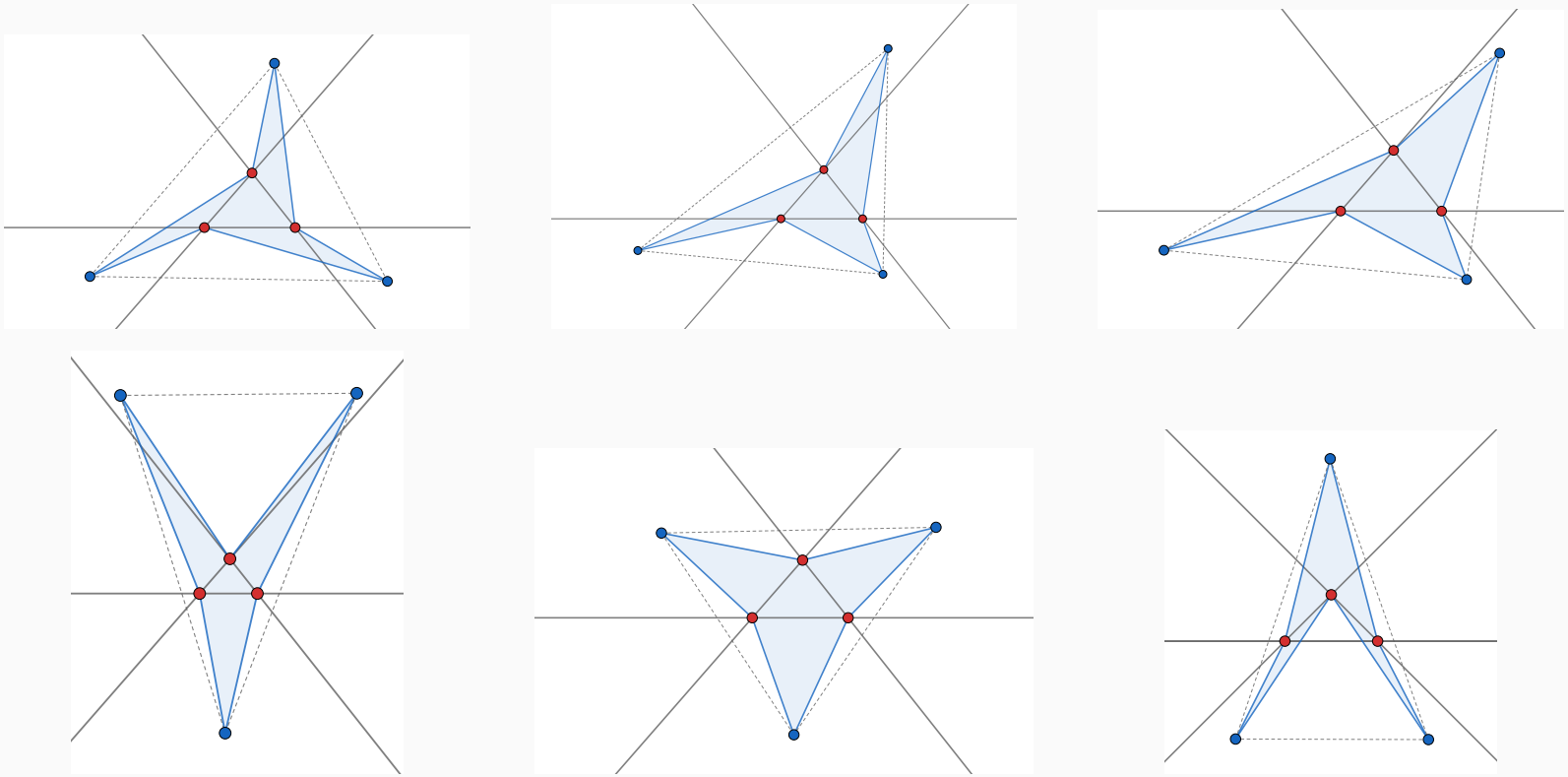
F. Alternating Cycle



For 3 interior points, reduce to at most 6 points in the hull.

Brute-force to find a 6-cycle. Why is there a 6-cycle?

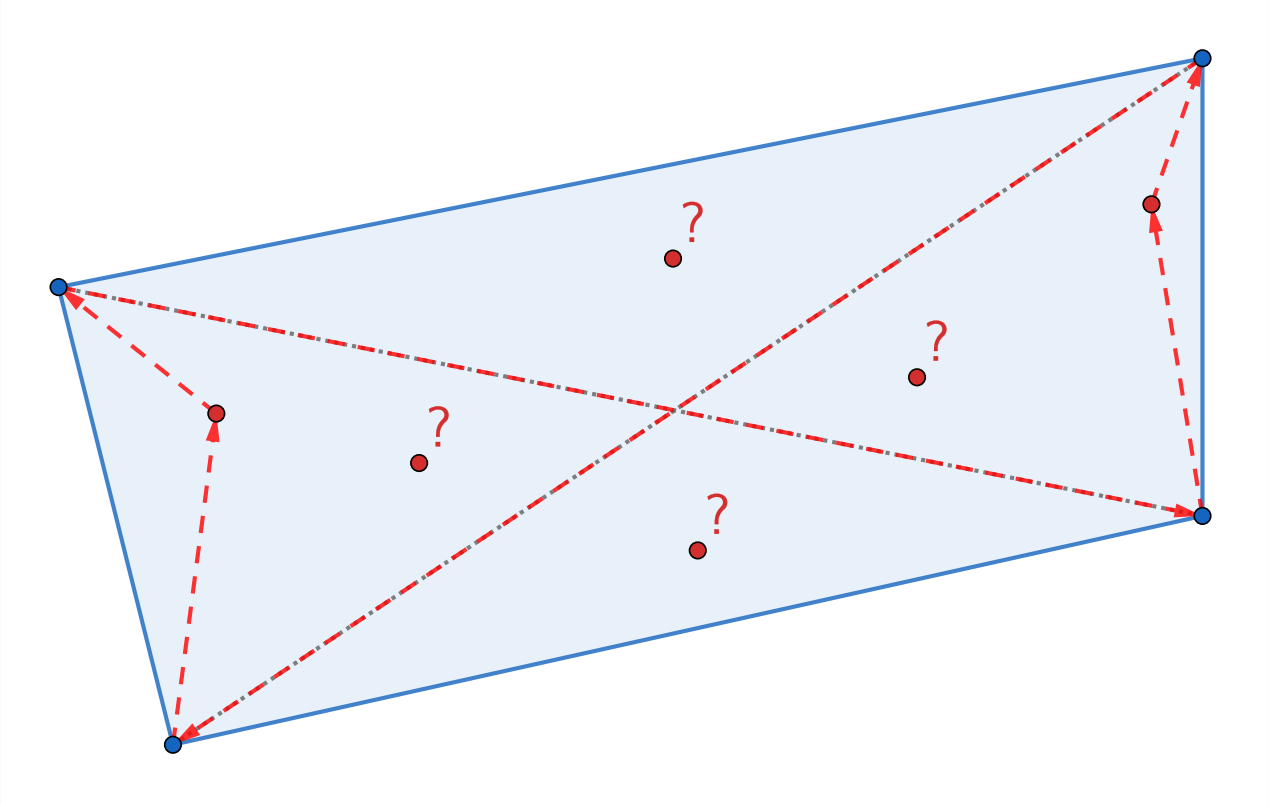
F. Alternating Cycle



Possible configurations for 3 points in the hull

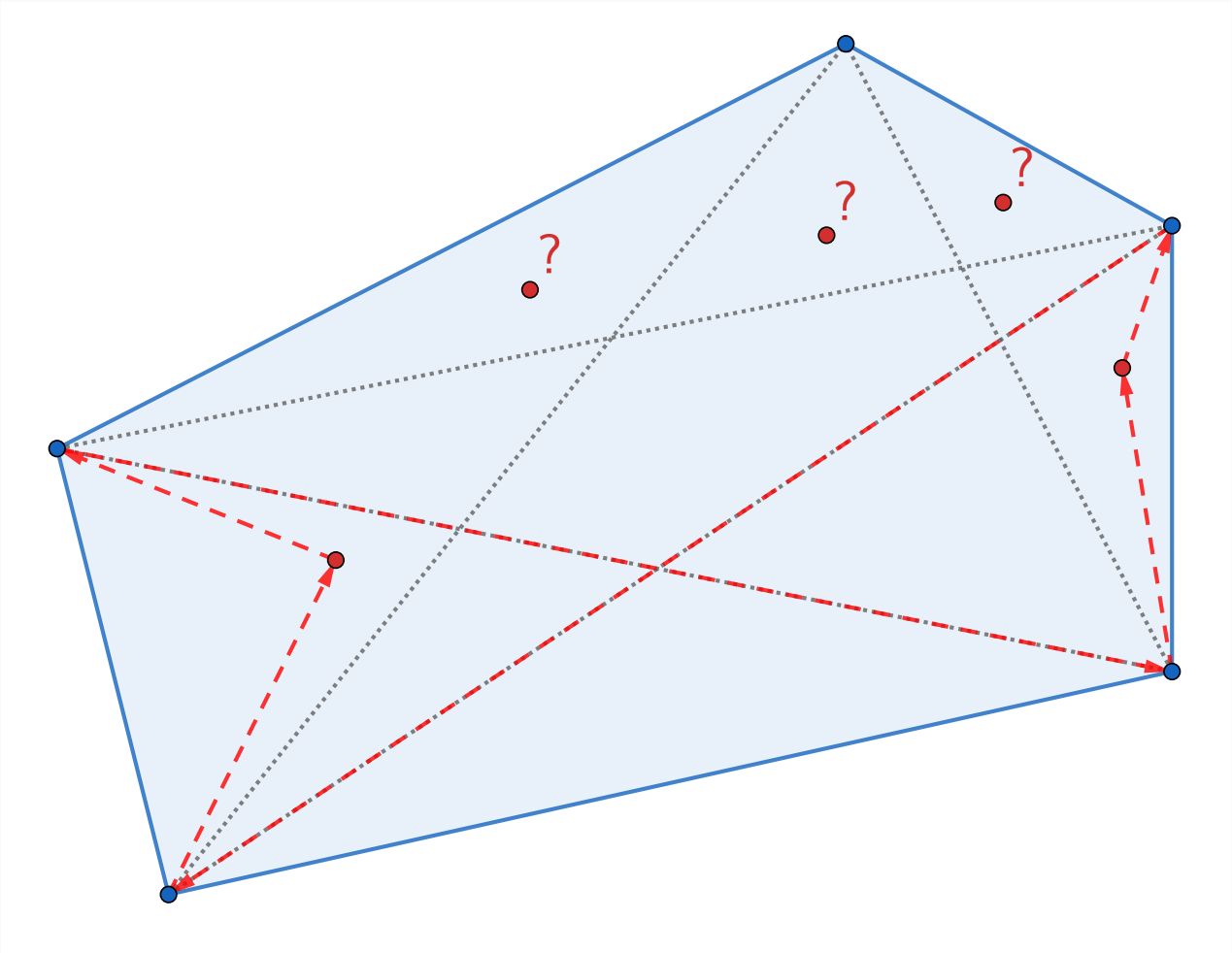
For 4-6 points, find butterfly or remove points.

F. Alternating Cycle



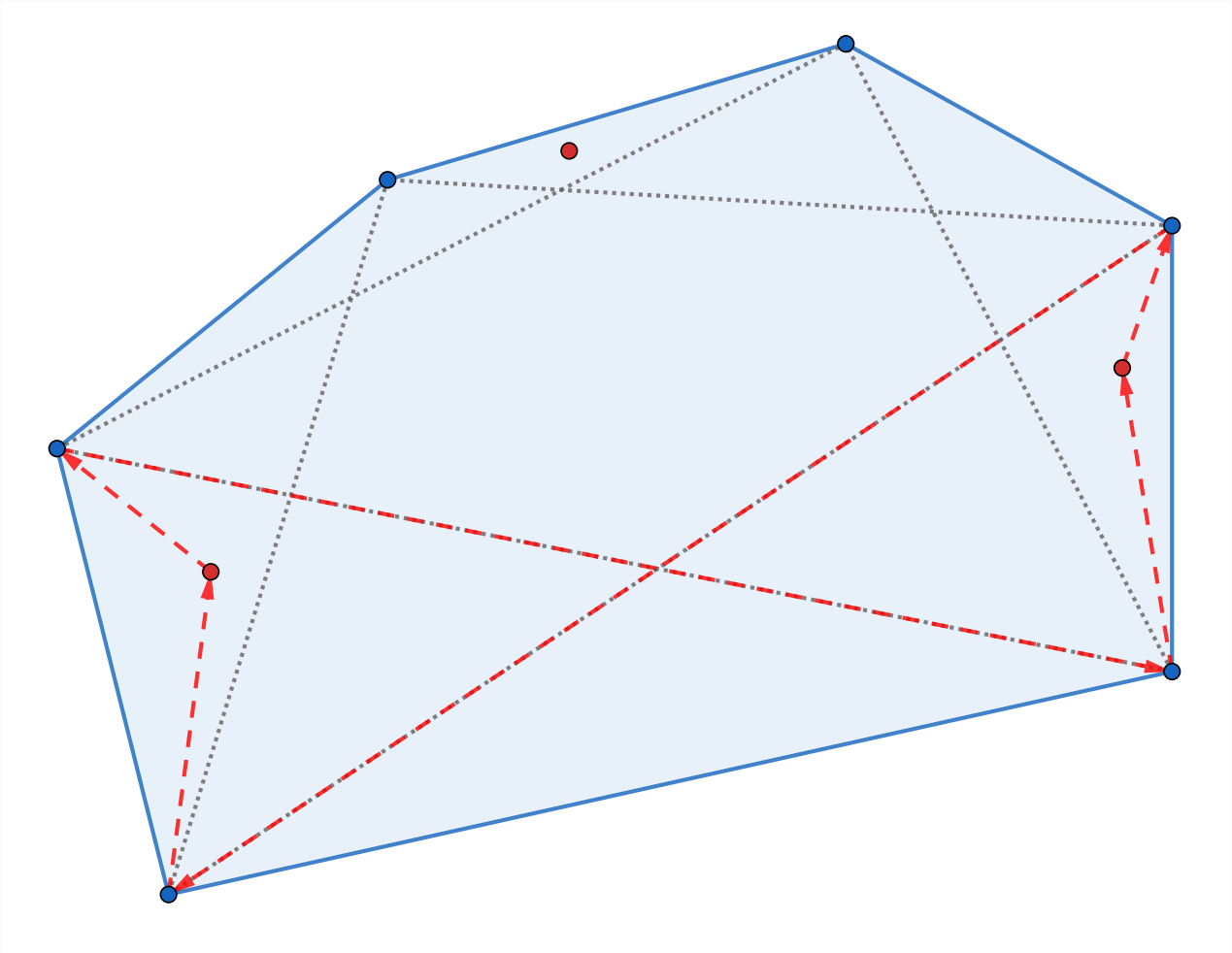
There is a butterfly if can't remove any of 4 hull points.

F. Alternating Cycle



There is a butterfly if can't remove any of 5 hull points.

F. Alternating Cycle



There is a butterfly if can't remove any of 6 hull points.

G. Touching Grass

Statement

Given n points. Answer m queries: given a segment, find a point that is "above" this segment. $n \leq 8 \cdot 10^5, m \leq 3 \cdot 10^5$. Low memory.

G. Touching Grass

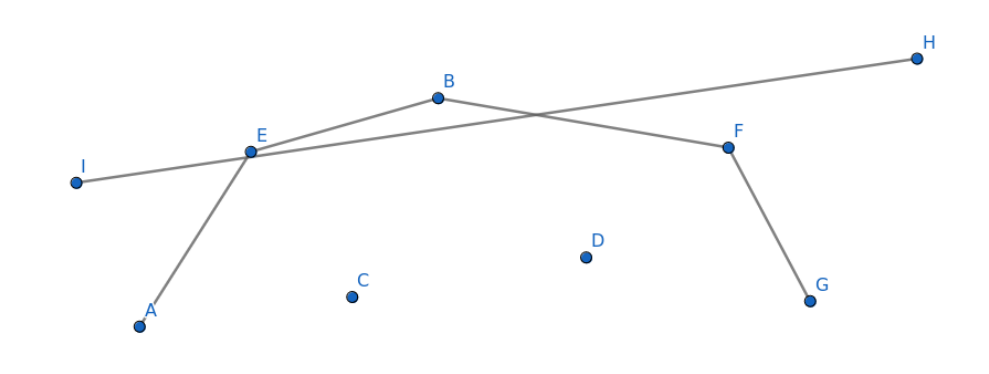
Statement

Given n points. Answer m queries: given a segment, find a point that is "above" this segment. $n \leq 8 \cdot 10^5, m \leq 3 \cdot 10^5$. Low memory.

Focus on a single query $(x_l, y_l) - (x_r, y_r)$.

Only points with $x_l \leq x \leq x_r$ can be the answer.

Of those, points not on the convex hull can't be the only answer.



(x, y) is "above the segment" only if $ax + by \geq c$ for some a, b, c.

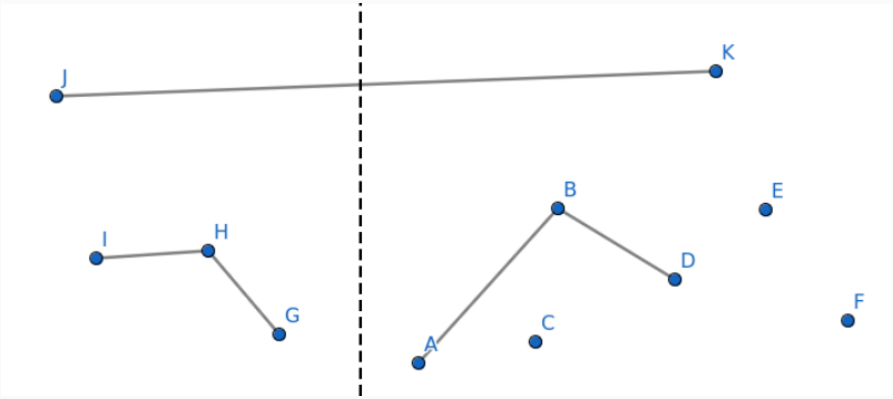
Hull points are in unimodal order w.r.t. this functional: binary search.

G. Touching Grass

Statement

Given n points. Answer m queries: given a segment, find a point that is "above" this segment. $n \leq 8 \cdot 10^5, m \leq 3 \cdot 10^5$. Low memory.

Divide and conquer. Split in the middle, handle queries that cross the middle.



Build hull of right side from left to right, at each prefix search for a match for appropriate queries.

G. Touching Grass

Statement

Given n points. Answer m queries: given a segment, find a point that is "above" this segment. $n \leq 8 \cdot 10^5, m \leq 3 \cdot 10^5$. Low memory.

How to know which queries to search on?

Build an array of all x-coordinates (endpoints of segments and given points).

Sort it and D&C on that.

I.e. when building a hull, if you see a given point, add it, otherwise if the other endpoint is over the split-line, do the search.

H. Game Design

Statement

Given a set $S \subseteq [n]$. Count the number of permutations p such that for each i, $p_i < i$ or $p_i \in S \wedge p_i \neq i$. $n \leq 5000$.

H. Game Design

Statement

Given a set $S \subseteq [n]$. Count the number of permutations p such that for each i, $p_i < i$ or $p_i \in S \wedge p_i \neq i$. $n \leq 5000$.

If $p_i = i$ were allowed for $p_i \in S$:

Build p left-to-right, maintain a pool of legal elements to set p_i to. Initially, pool is S.

- Pop any element of the pool, set p_i to that.
- Add i to the pool if $i \notin S$.

Answer: product of pool sizes throughout the process.

H. Game Design

Statement

Given a set $S \subseteq [n]$. Count the number of permutations p such that for each i, $p_i < i$ or $p_i \in S \wedge p_i \neq i$. $n \leq 5000$.

How to avoid $p_i = i$?

Inclusion-exclusion principle.

Naively: for each subset $T \subseteq S$, do the same as before but set $p_i = i$ for $i \in T$. Add to the answer if $|T|$ even, subtract otherwise.

H. Game Design

Statement

Given a set $S \subseteq [n]$. Count the number of permutations p such that for each i, $p_i < i$ or $p_i \in S \wedge p_i \neq i$. $n \leq 5000$.

How to avoid $p_i = i$?

Inclusion-exclusion principle.

Naively: for each subset $T \subseteq S$, do the same as before but set $p_i = i$ for $i \in T$. Add to the answer if $|T|$ even, subtract otherwise.

Calculate the sum of all these products simultaneously.

$dp[i][j]$ — i first positions considered, j elements in the pool.

Statement

Given a polygon, consider ray-casting to check whether it contains the origin: count the parity of edges and points that intersect $\overrightarrow{Ox}$.

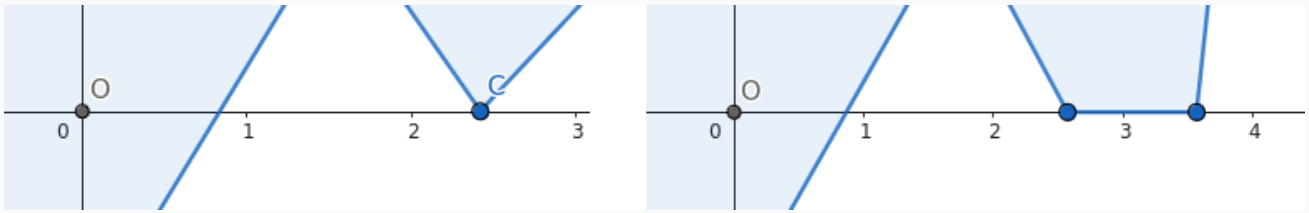
Order all integer polygons where this algorithm gives a false negative, by area. Print the first 1000.

Statement

Given a polygon, consider ray-casting to check whether it contains the origin: count the parity of edges and points that intersect $\overrightarrow{Ox}$.

Order all integer polygons where this algorithm gives a false negative, by area. Print the first 1000.

The algorithm fails to consider these edge cases:



The algorithm never gives a false negative for convex polygons.

I. Geometry Hacking

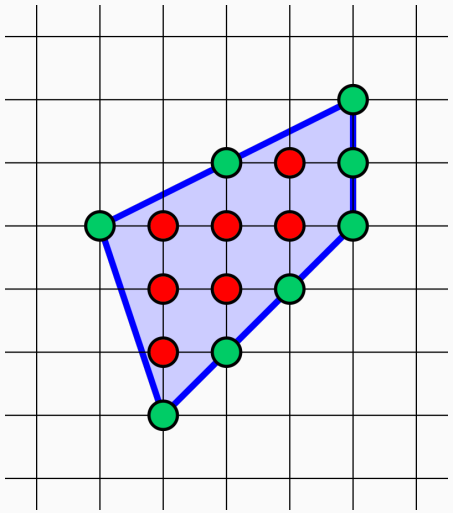
Statement

Given a polygon, consider ray-casting to check whether it contains the origin: count the parity of edges and points that intersect $\overrightarrow{Ox}$.

Order all integer polygons where this algorithm gives a false negative, by area. Print the first 1000.

What is the minimum area for counterexamples?

Pick's theorem: $A = i + b/2 - 1 \geq 1 + n/2 - 1$. Triangles are convex, so $A \geq 2$.



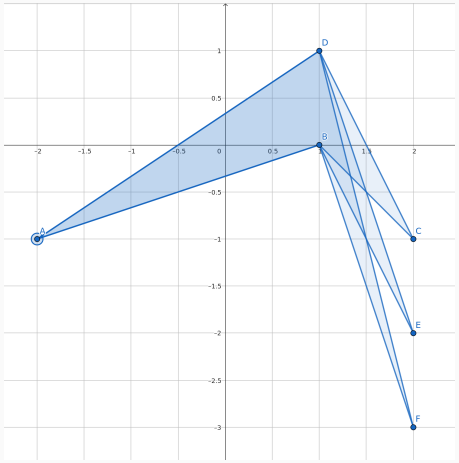
I. Geometry Hacking

Statement

Given a polygon, consider ray-casting to check whether it contains the origin: count the parity of edges and points that intersect $\overrightarrow{Ox}$.

Order all integer polygons where this algorithm gives a false negative, by area. Print the first 1000.

How to get a false positive for $A = 2$?



$$\{((-2,-1),(1,0),(2,-k),(1,1)) \mid k \in \mathbb{N}\}$$

J. Non-Interactive Nim

Statement

Given a losing state of Nim, print a sequence of moves, ending with a loss, such that the opponent has only one optimal move every time.

J. Non-Interactive Nim

Statement

Given a losing state of Nim, print a sequence of moves, ending with a loss, such that the opponent has only one optimal move every time.

Nim is losing iff $a_1 \oplus a_2 \oplus \cdots \oplus a_n = 0$.

In a winning state, the opponent must set $a_i \leftarrow a_i \oplus x$, where x is the total XOR. This is valid iff the highest set bit in x is set in a_i.

Pick a bit k that appears only in a_i and a_j, set $a_i \leftarrow a_i \oplus x$. Opponent will $a_j \leftarrow a_j \oplus x$, where highest set bit of x is k.

J. Non-Interactive Nim

Statement

Given a losing state of Nim, print a sequence of moves, ending with a loss, such that the opponent has only one optimal move every time.

Nim is losing iff $a_1 \oplus a_2 \oplus \cdots \oplus a_n = 0$.

In a winning state, the opponent must set $a_i \leftarrow a_i \oplus x$, where x is the total XOR. This is valid iff the highest set bit in x is set in a_i.

Pick a bit k that appears only in a_i and a_j, set $a_i \leftarrow a_i \oplus x$. Opponent will $a_j \leftarrow a_j \oplus x$, where highest set bit of x is k.

For each $k = 59, \ldots, 0$:

- If k is not set anywhere, do nothing.
- If k is set in only a_i, a_j, do $a_i \leftarrow a_i \oplus a_j$, $a_j \leftarrow 0$.
- Otherwise, report impossible.

J. Non-Interactive Nim

Statement

Given a losing state of Nim, print a sequence of moves, ending with a loss, such that the opponent has only one optimal move every time.

Can prove: there is a solution iff this process doesn't fail:

Make a DSU on n vertices. For each $k = 59, 58, \ldots, 0$, consider the connected components that have bit k set an odd number of times.

- If there are no such components, do nothing.
- If there are exactly two, join them.
- Otherwise, fail.

K. String and Nails

Statement

Given n points. Remove them in an order such that after each move, the convex hull changes.

Remove points in the lexicographical order.

L. All-You-Can-Eat

Statement

You are given a sequence of items, online. Solve knapsack. You are allowed to throw away items you've already picked. At the end, your solution must be at least 60% of the optimum.

L. All-You-Can-Eat

$[0, 400)$ – small, $[400, 600)$ – large, $[600, 1000]$ – massive.

1. Sum in bag ≥ 600: ignore.
2. i massive: take i, discard everything else.
3. i can be added without discarding anything: take i.

L. All-You-Can-Eat

$[0, 400)$ – small, $[400, 600)$ – large, $[600, 1000]$ – massive.

1. Sum in bag ≥ 600: ignore.
2. i massive: take i, discard everything else.
3. i can be added without discarding anything: take i.
4. No large items in bag: take i, discard until fits.

L. All-You-Can-Eat

$[0, 400)$ – small, $[400, 600)$ – large, $[600, 1000]$ – massive.

1. Sum in bag ≥ 600: ignore.
2. i massive: take i, discard everything else.
3. i can be added without discarding anything: take i.
4. No large items in bag: take i, discard until fits.
5. If $\ell + a_i \leq 1000$: take i, discard all except ℓ.
6. If $a_i < \ell$: take i, discard ℓ.
7. Otherwise: ignore.

2, 4, 5 occur: instant win.

L. All-You-Can-Eat

$[0, 400)$ – small, $[400, 600)$ – large, $[600, 1000]$ – massive.

1. Sum in bag ≥ 600: ignore.
2. i massive: take i, discard everything else.
3. i can be added without discarding anything: take i.
4. No large items in bag: take i, discard until fits.
5. If $\ell + a_i \leq 1000$: take i, discard all except ℓ.
6. If $a_i < \ell$: take i, discard ℓ.
7. Otherwise: ignore.

2, 4, 5 occur: instant win.

Only 3, 6, 7: we take all smalls and the smallest large. Optimum takes one large and some smalls.

$$\frac{\text{all small} + \min \ell_i}{\text{some small} + \ell_j} \geq \frac{\text{all small} + \min \ell_i}{\text{all small} + \ell_j} \geq \frac{\min \ell_i}{\ell_j} \geq 0.6.$$