Problem Solving Strategies in Computational Geometry

Yingying Wu, *ICPC NAC-NAPC Trainer, yingyingwu.io*

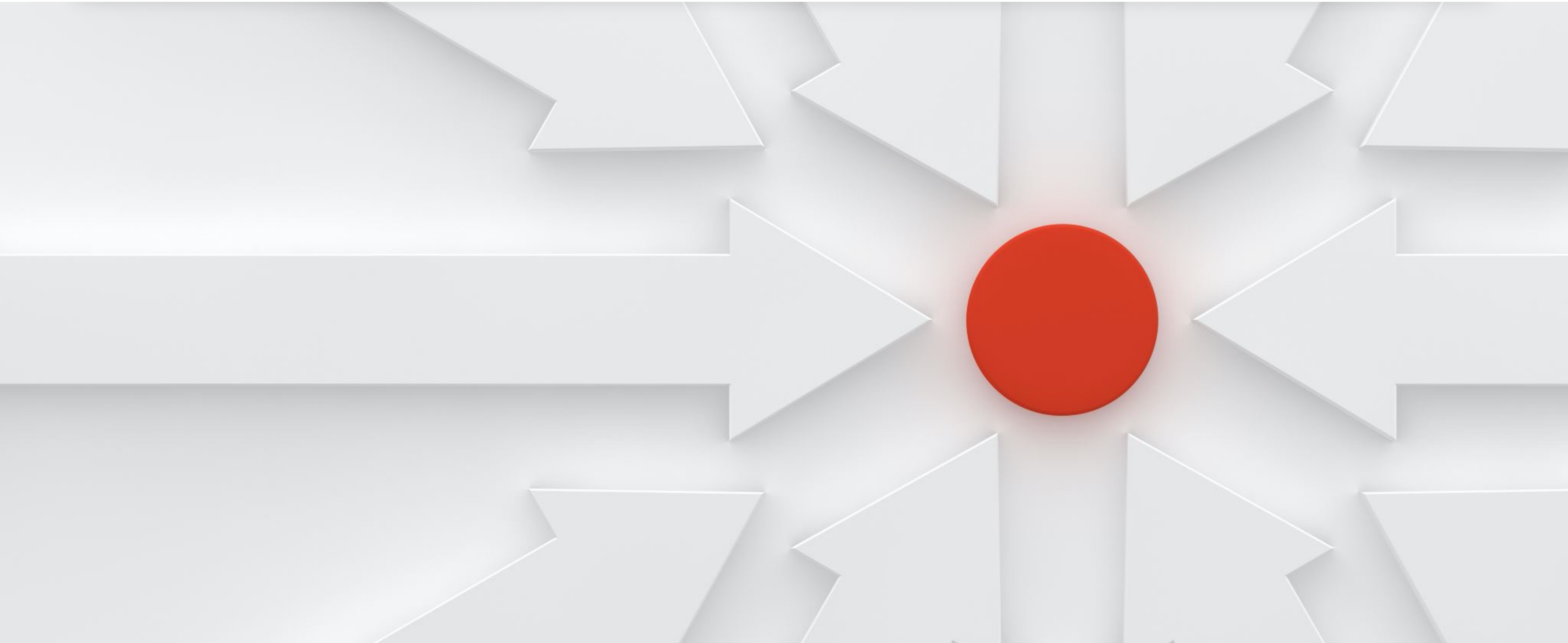
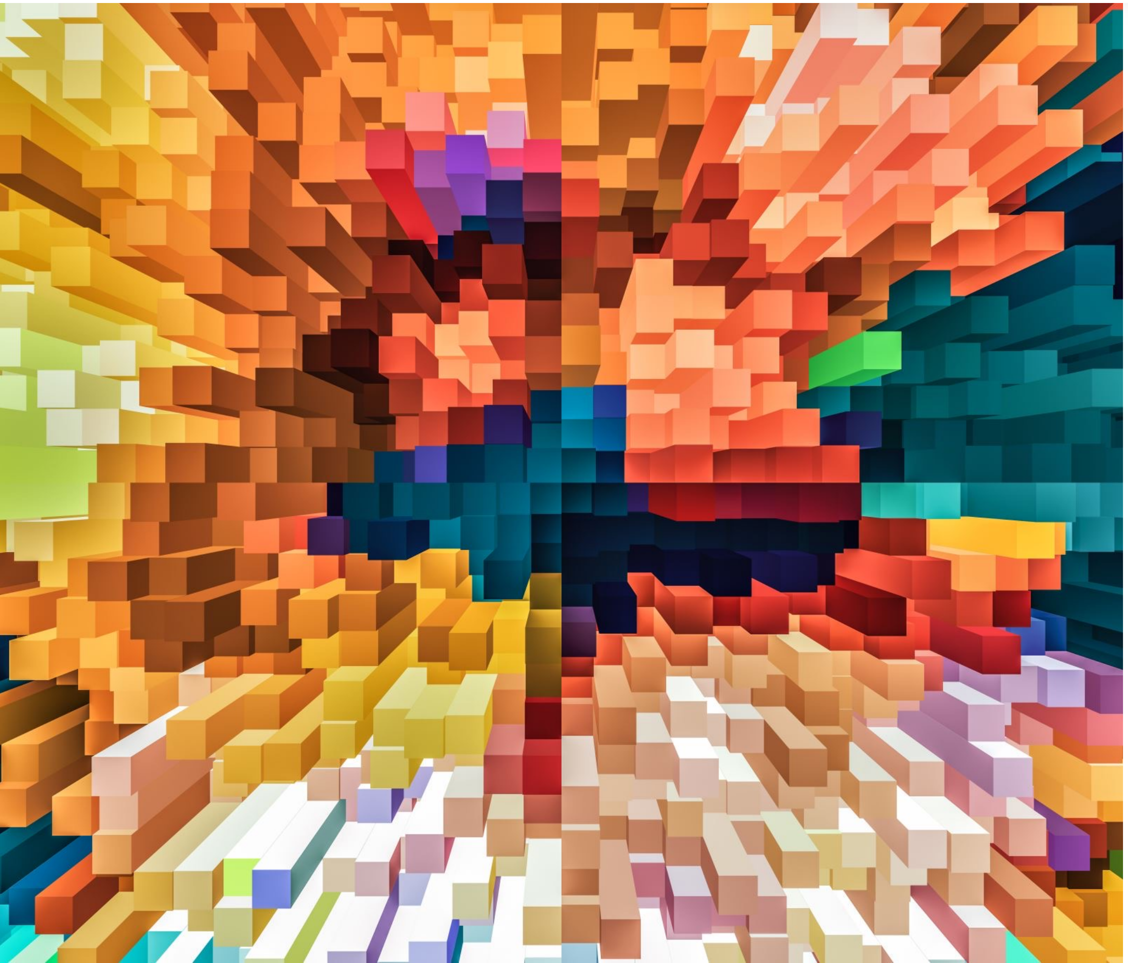
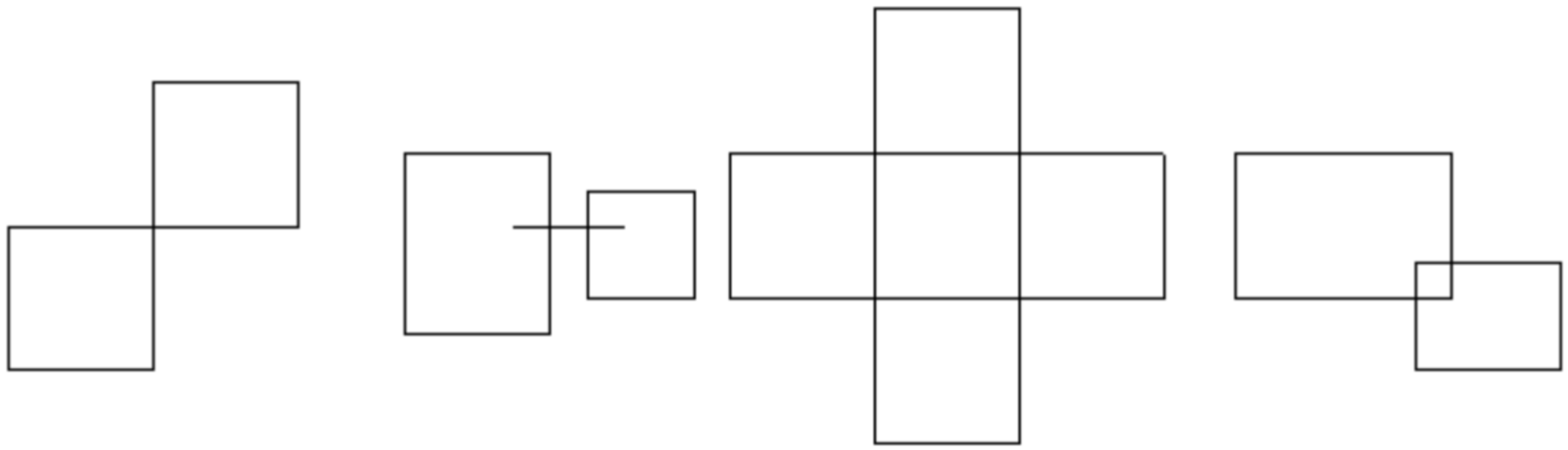


Table of Contents



Tin Cutter

https://vjudge.net/problem/UVA-308



two holes two holes one hole one hole

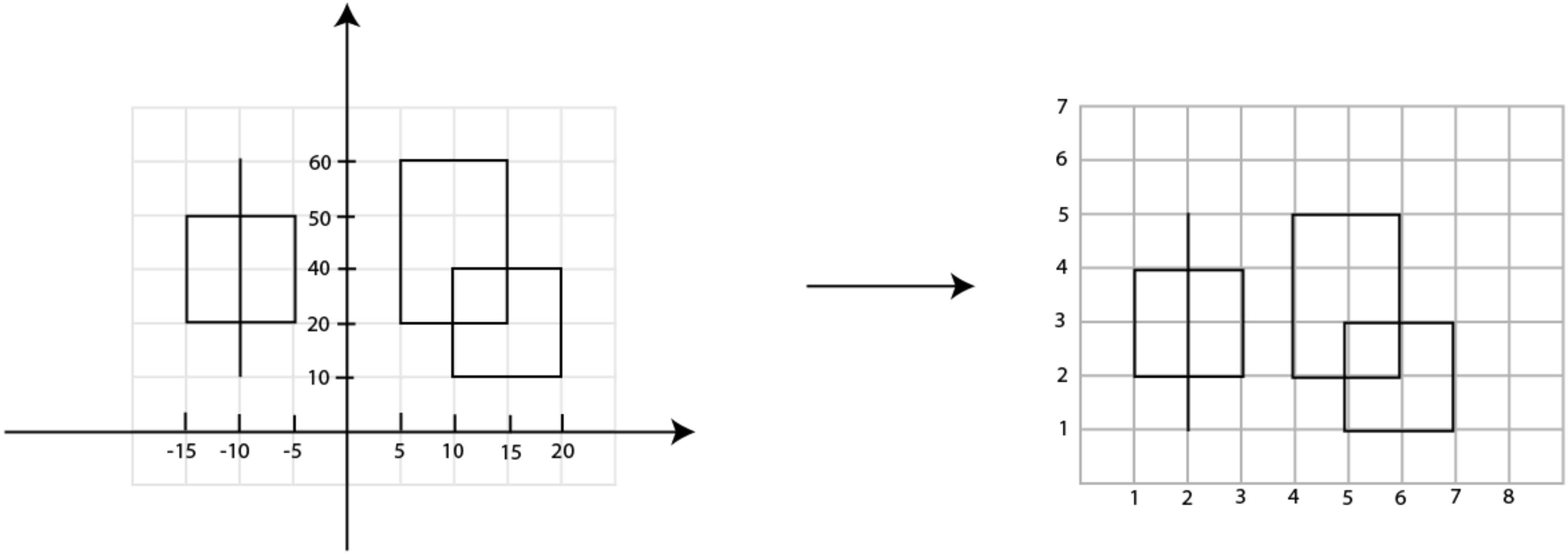
Cuts: horizontal or vertical, integer coordinates.

Each segment cut is given by its endpoints (inside the tin plate).

Some parts of tin plate can fall out and so some holes in the plate can emerge.

Predict the number of holes in the plate.

Single segment cuts are not considered to be holes.

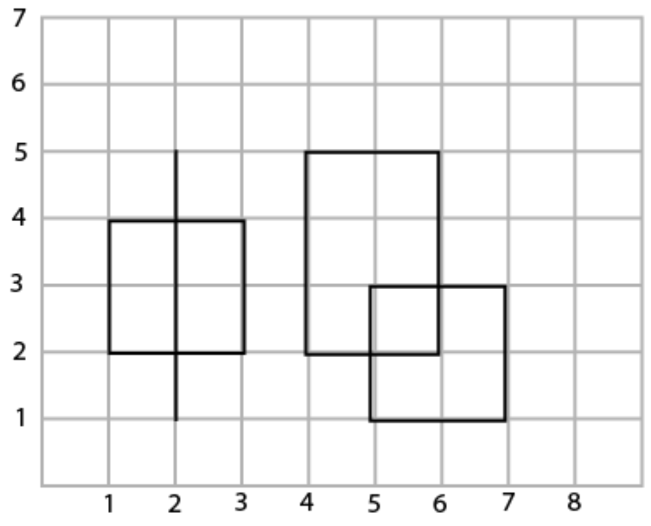


Cuts: <= 1000, Coordinates: [-10^5, 10^5]

[-10^5, 10^5] to large to discretize. 1000 cuts: discretize to 2000 grid.

x-axis (sorted): -15, -10, -5, 5, 10, 15, 20 → 1, 2, 3, 4, 5, 6, 7

y-axis (sorted): 10, 20, 40, 50, 60 → 1, 2, 3, 4, 5



- Binary Insertion Sort or qsort, O(n log n)
- Floodfill from (0,0)
- Count number of holes:
  - While there is unvisited point remaining point (unvisited), Floodfill

```
Flood-fill (node):
 1. If node is not Inside return.
 2. Set the node
 3. Perform Flood-fill one step to the south of node.
 4. Perform Flood-fill one step to the north of node
 5. Perform Flood-fill one step to the west of node
 6. Perform Flood-fill one step to the east of node
 7. Return.
```

Area and Circumference

Discretization with sweeping / alternating trick.

Area

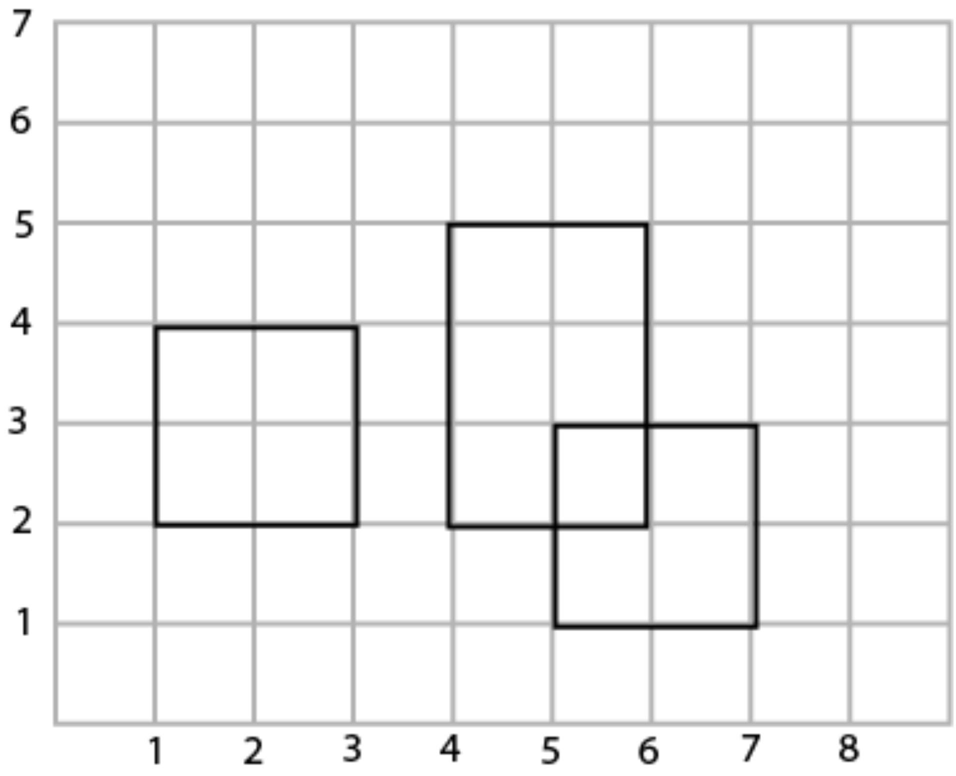
Cuts: horizontal or vertical.

Coordinates: integer.

Each cut is rectangular.

Some parts can fall out and so some holes in the plate can emerge.

What's the total area of the cuts?



Area

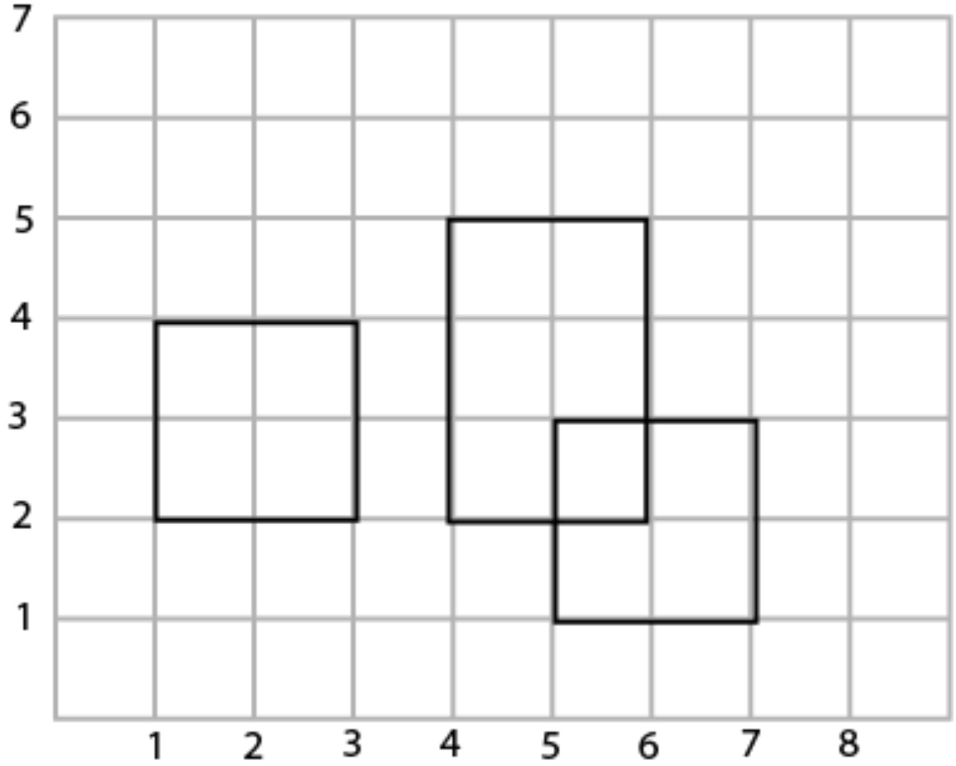
Upper edge counter++, lower edge counter--

Sweep x-axis from left to right.

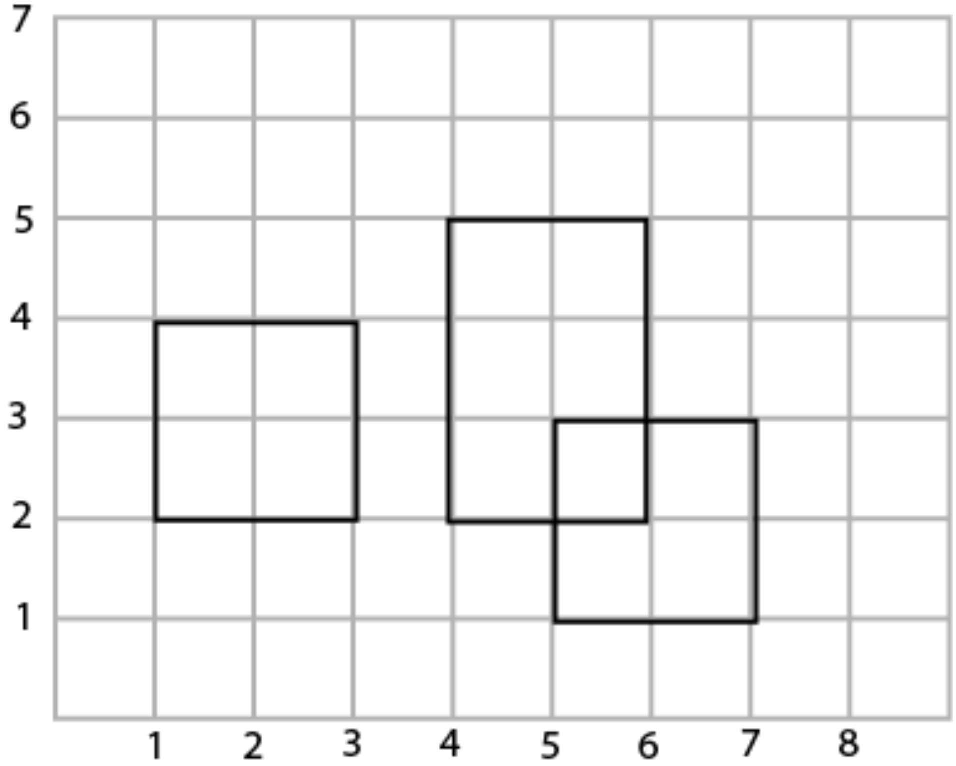
Initialization: counter = 0, area = 0

Accumulate area when counter > 0.

- At x = 1
  - At y = 4 counter = 1, area = 1
  - At y = 3 counter = 1, area = 2
  - At y = 2 counter = 0
- At x = 2
  - At y = 4 counter = 1, area = 3
  - At y = 3 counter = 1, area = 4
  - At y = 2 counter = 0

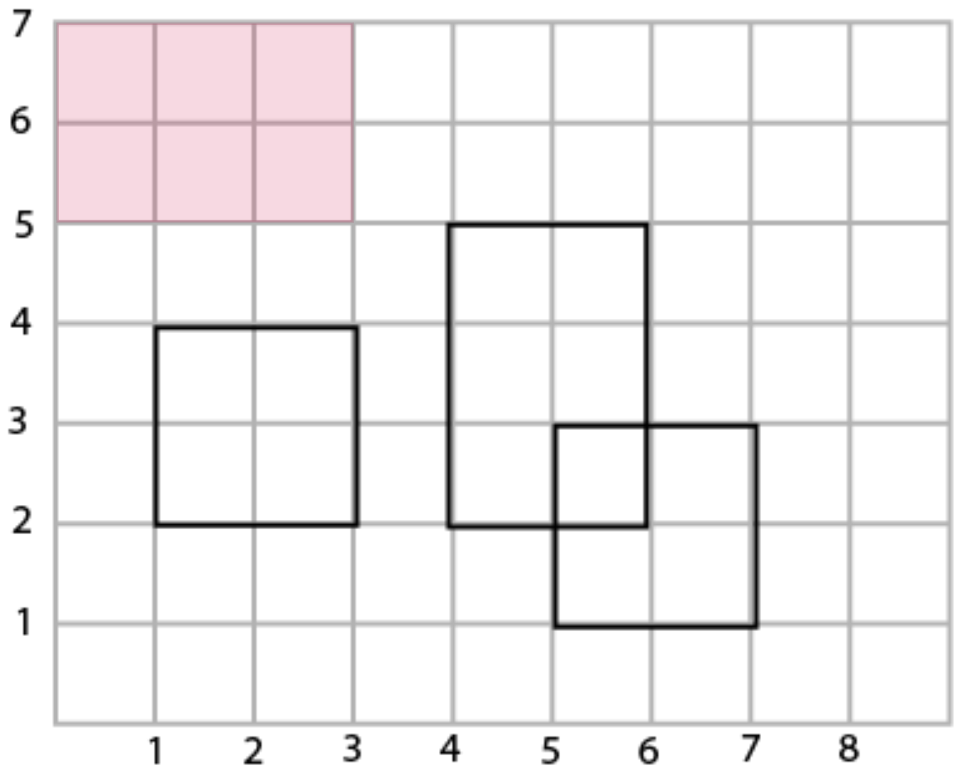


- At x = 4
  - At y = 5 counter=1, area=5
  - At y = 4 counter=1, area=6
  - At y = 3 counter=1, area=7
  - At y = 2 counter=0
- At x = 5
  - At y = 5 counter=1, area=8
  - At y = 4 counter=1, area=9
  - At y = 3 counter=2, area=10
  - At y = 2 counter=1, area = 11
  - At y = 2 counter=0
- At x = 6
  - At y = 3 counter=1, area=12
  - At y = 2 counter=1, area=13
  - At y = 1 counter=0



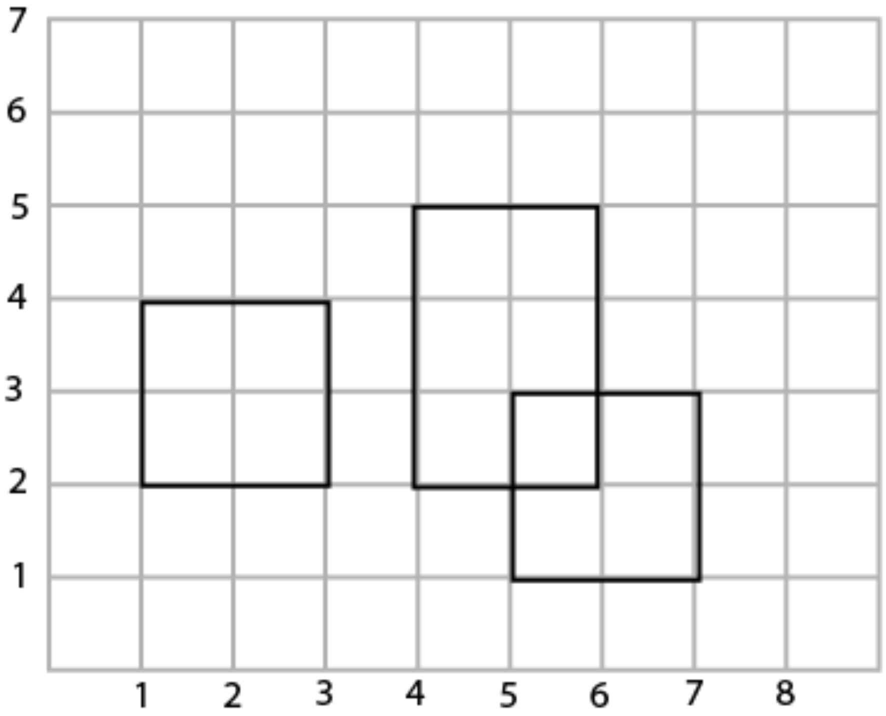
Monster Version: https://cses.fi/problemset/task/1741

- We sweep from left to right over the x-axis. Maintain a Lazy Segment Tree over the y -coordinates.
- When we run into a left boundary of some rectangle with y-coordinates (y_0, y_1), increase by 1.
- When we run into a right boundary of some rectangle with y-coordinates (y_0, y_1), decrease by 1
- Then, for each x, we count the number of non-zero indices (in practice, count the amount of space covered by no rectangles and subtract this amount from the total).



Code and Tutorial: https://usaco.guide/adv/count-min?lang=cpp
Monster Version: https://cses.fi/problemset/task/1741

Circumference



Upper edge counter++, lower edge counter--

Sweep x-axis from left to right.

Initialization: Circumference = 0, counter = 0

Accumulate circumference when

- Counter 0 → 1, Counter 1 → 0

(Repeat for vertical edges)

Right edge counter++, left edge counter --

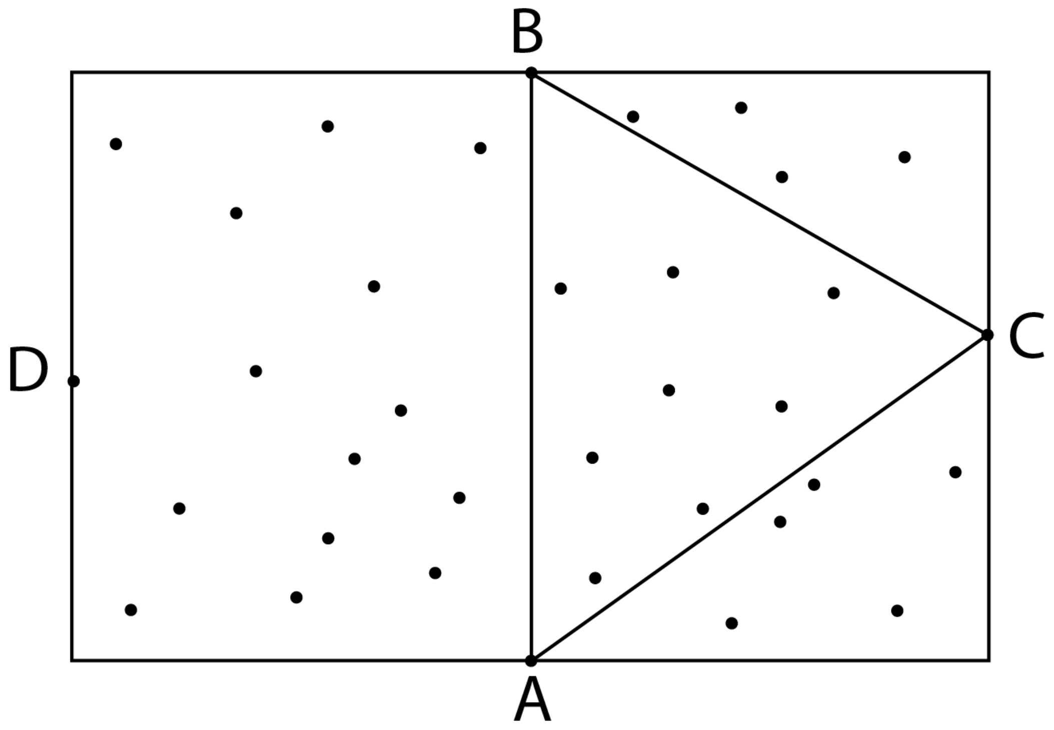
Sweep y-axis from down to up.

Accumulate circumference when

- Counter 0 → 1, Counter 1 → 0

Convex hull with Divide-and-Conquer

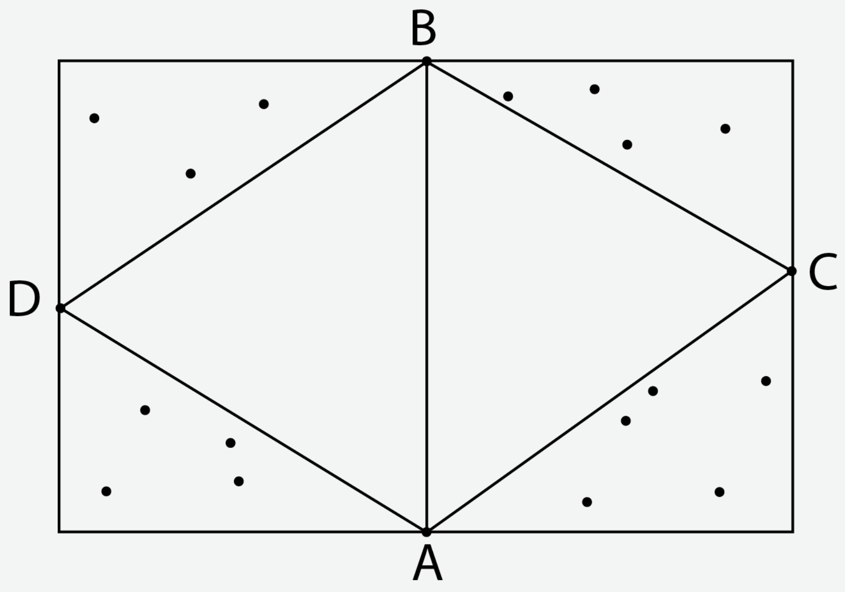
Quick Hull



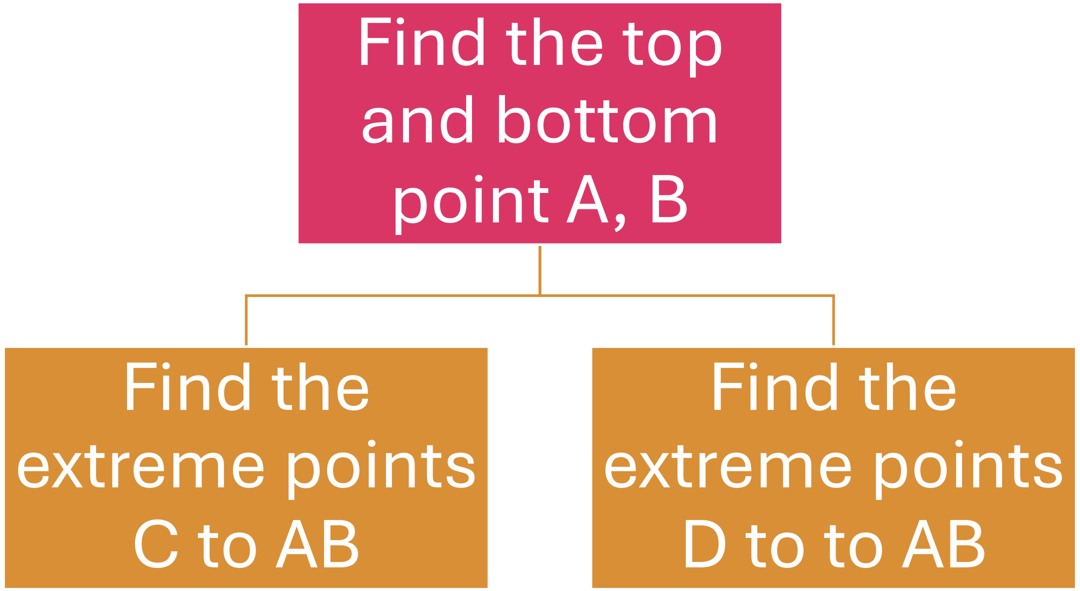
Divide-and-Conquer Strategy for Convex hull

Can we divide a point set into subsets, find convex hull for each subsets, and then combine these convex hulls?

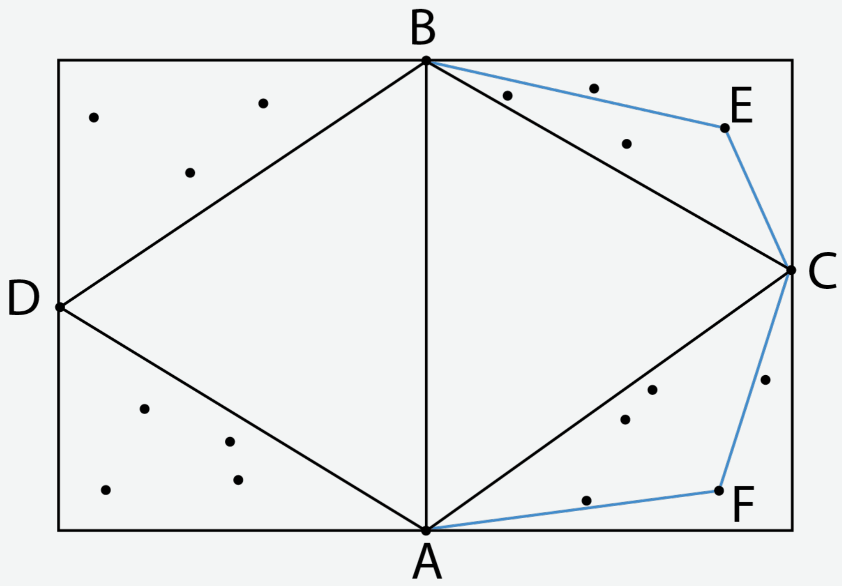
Divide-and-Conquer Strategy for Convex hull



Points inside ABC and ABD are excluded.

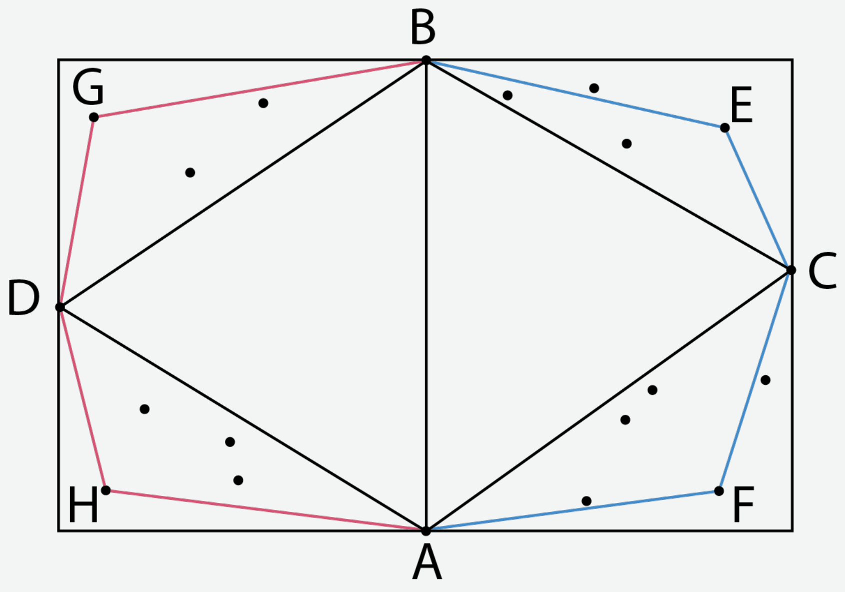


Divide-and-Conquer Strategy for Convex hull



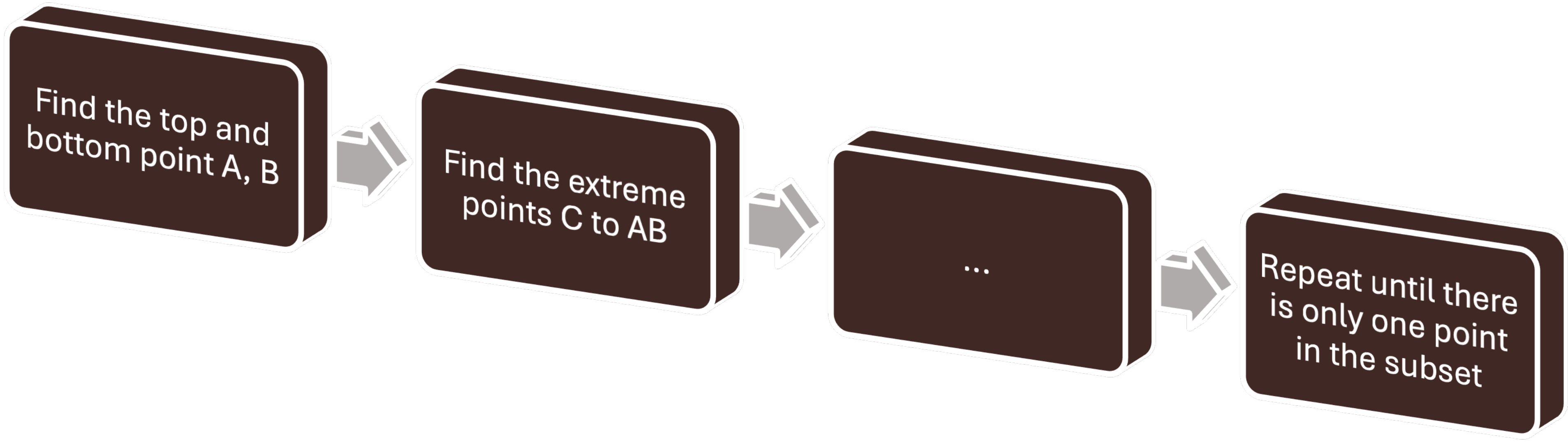
- Find the top and bottom point A, B
  - Find the extreme points C to AB
    - Find the extreme points E to to BC
    - Find the extreme points F to to AC
  - Find the extreme points D to to AB

Divide-and-Conquer Strategy for Convex hull



- Find the top and bottom point A, B
  - Find the extreme points C to AB
    - Find the extreme points E to to BC
    - Find the extreme points F to to AC
  - Find the extreme points D to to AB
    - Find the extreme points G to to DB
    - Find the extreme points H to to AD

Divide-and-Conquer Strategy for Convex hull

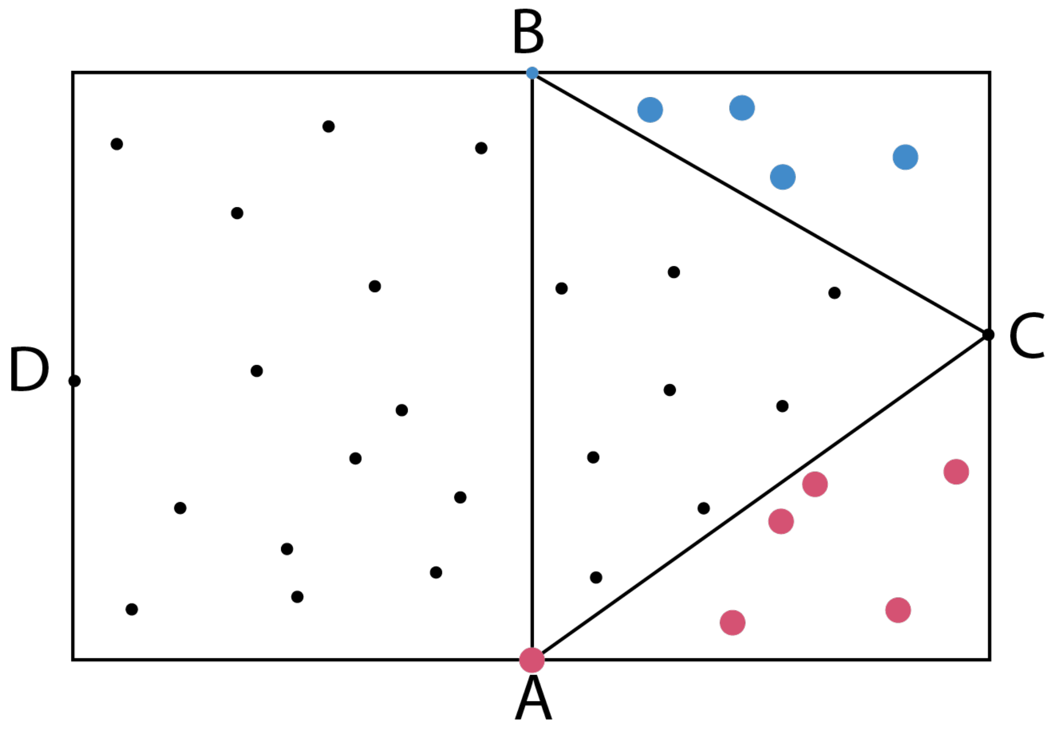


Quick Hull Complexity

Worst case complexity

Average case complexity

Complexity of Quick Hull



If the time computing the convex hull of S is T, then

$$T(S) = T(X) + T(Y) + O(n)$$

- Set X to be the set of points to the lower right of AC, say |X| = p
- Set Y to be the set of points to the upper right of BC, say |Y| = q
- This step takes O(n).
- Say we have n points in S (to the right of AB). The worst case is p + q = n-1.

Worst Case Complexity for Quick Hull

Master Theorem

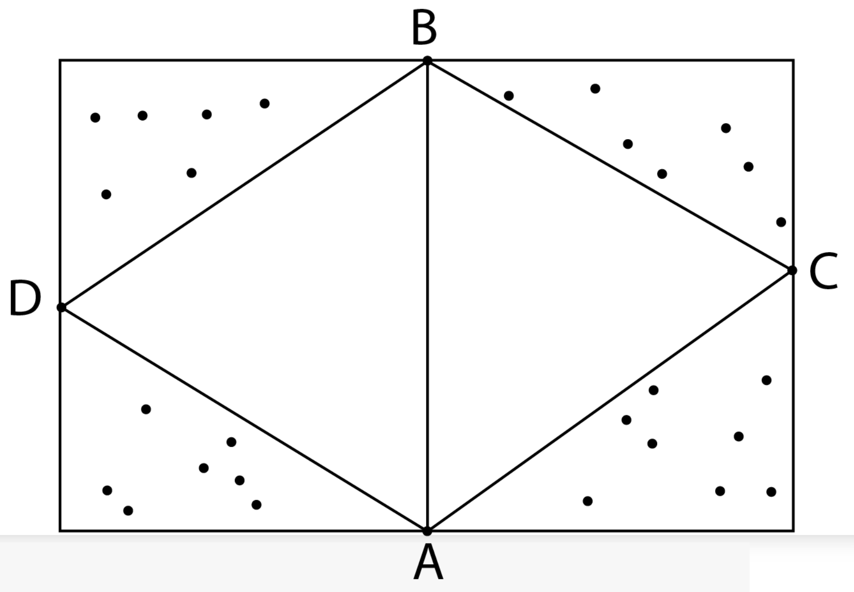
Given a recurrence of the form

$$T(n) = aT\left(\frac{n}{b}\right) + f(n),$$

for constants $a\,(\geq 1)$) and $b\,(> 1)$ with f asymptotically positive, the following statements are true:

- **Case 1.** If $f(n) = O\left(n^{\log_b a - \epsilon}\right)$ for some $\epsilon > 0$, then $T(n) = \Theta\left(n^{\log_b a}\right)$.
- **Case 2.** If $f(n) = \Theta\left(n^{\log_b a}\right)$, then $T(n) = \Theta\left(n^{\log_b a} \log n\right)$.
- **Case 3.** If $f(n) = \Omega\left(n^{\log_b a + \epsilon}\right)$ for some $\epsilon > 0$ $\big($and $af\left(\frac{n}{b}\right) \leq cf(n)$ for some $c < 1$ for all n sufficiently large$\big)$, then $T(n) = \Theta\big(f(n)\big)$.

Worst Case Best Scenario



Given a recurrence of the form

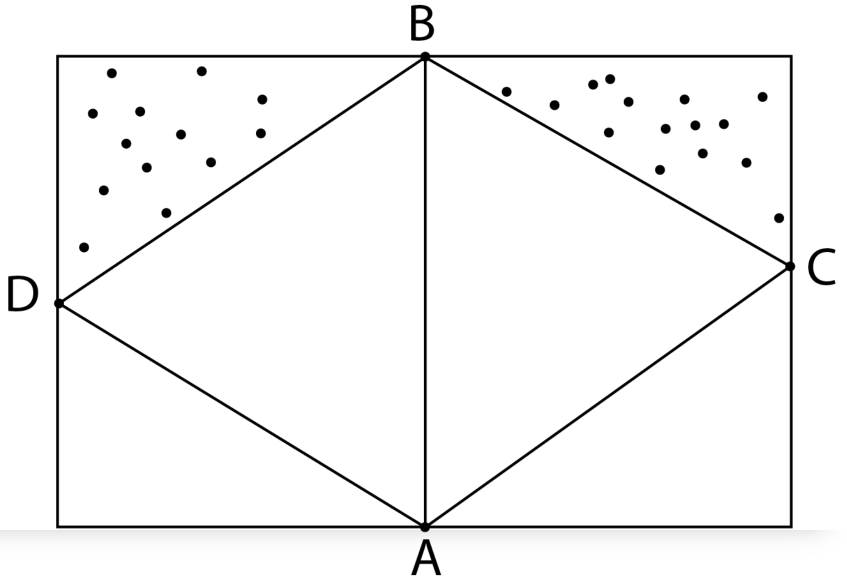
$$T(n) = aT\left(\frac{n}{b}\right) + f(n),$$

for constants $a\ (\geq 1)$) and $b\ (> 1)$ with f asymptotically positive, the following statements are true:

- **Case 1.** If $f(n) = O\left(n^{\log_b a - \epsilon}\right)$ for some $\epsilon > 0$, then $T(n) = \Theta\left(n^{\log_b a}\right)$.
- **Case 2.** If $f(n) = \Theta\left(n^{\log_b a}\right)$, then $T(n) = \Theta\left(n^{\log_b a} \log n\right)$.
- **Case 3.** If $f(n) = \Omega\left(n^{\log_b a + \epsilon}\right)$ for some $\epsilon > 0$ $\left(\text{and } af\left(\frac{n}{b}\right) \leq cf(n) \text{ for some } c < 1 \text{ for all } n \text{ sufficiently large}\right)$, then $T(n) = \Theta(f(n))$.

- |X| = |Y| = n/2, then $T(n) = 2T\left(\frac{n}{2}\right) + O(n) \Rightarrow T(n) = O(n \log n)$
- $f(n) = \Theta(n)$ since every point has to be traversed. So Case 2.

Worst Case Worst Scenario



- |X| = n - 1, then $T(n) = T(n-1) + O(n) \quad \sim \quad T(n) = O(n^2)$

Master Theorem

Given a recurrence of the form

$$T(n) = aT\left(\frac{n}{b}\right) + f(n),$$

for constants $a\,(\geq 1)$) and $\boxed{b\,(> 1)}$ with f asymptotically positive, the following statements are true:

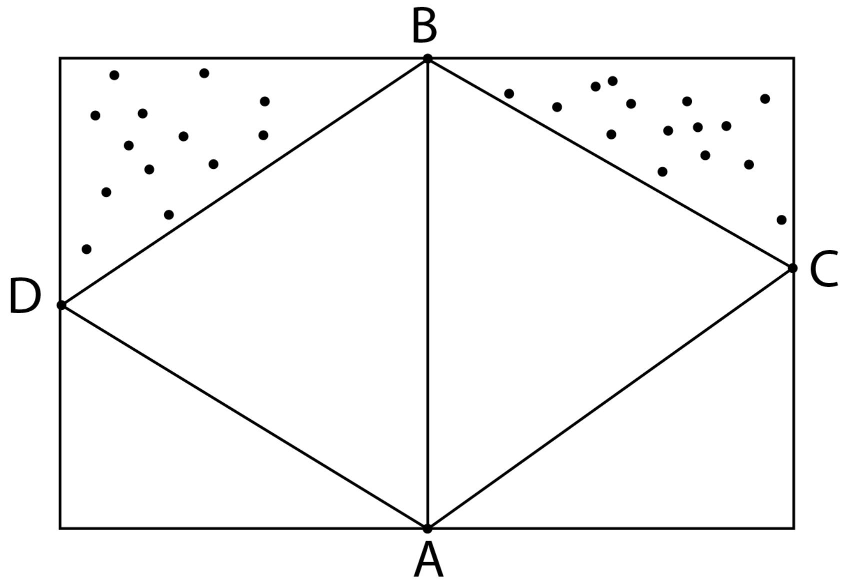
- **Case 1.** If $f(n) = O\left(n^{\log_b a - \epsilon}\right)$ for some $\epsilon > 0$, then $T(n) = \Theta\left(n^{\log_b a}\right)$.
- **Case 2.** If $f(n) = \Theta\left(n^{\log_b a}\right)$, then $T(n) = \Theta\left(n^{\log_b a} \log n\right)$.
- **Case 3.** If $f(n) = \Omega\left(n^{\log_b a + \epsilon}\right)$ for some $\epsilon > 0$ $\big($and $af\left(\frac{n}{b}\right) \leq cf(n)$ for some $c < 1$ for all n sufficiently large$\big)$, then $T(n) = \Theta\big(f(n)\big)$.

- In this case, b = 1, so Master Theorem does not apply.

Worst Case Worst Scenario

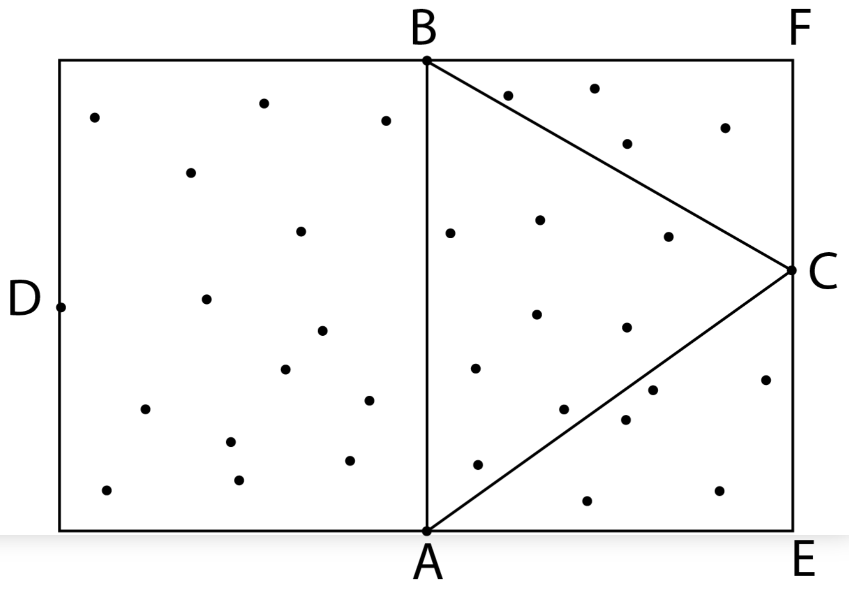
- |X| = n - 1 , $T(n) = T(n-1) + O(n) \quad \sim \quad T(n) = O(n^2)$

$$\begin{aligned} T(n) &= T(n-1) + O(n) \\ &= T(n-2) + O(n) \\ &\vdots \\ &= nO(n) + T(1) \\ &= O(n^2) \end{aligned}$$



Average Case Complexity

Best case scenario: O(n)



- In the average case, points inside of ABC = points outside of ABC, i.e.,

  Area(AEFB) = 2Area(ABC)

- That implies

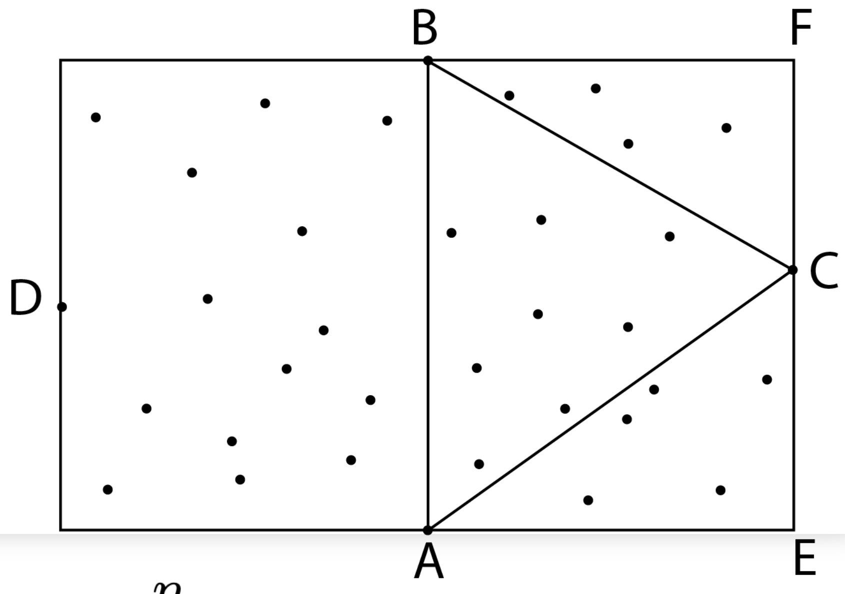
$$T(n) = O(n) + T(p) + T(q), p + q = \frac{n}{2}$$

- Average case best scenario

$$T(n) = O(n) + 2T\left(\frac{n}{4}\right), p = q = \frac{n}{4}$$

Average Case Complexity

Best scenario: O(n)

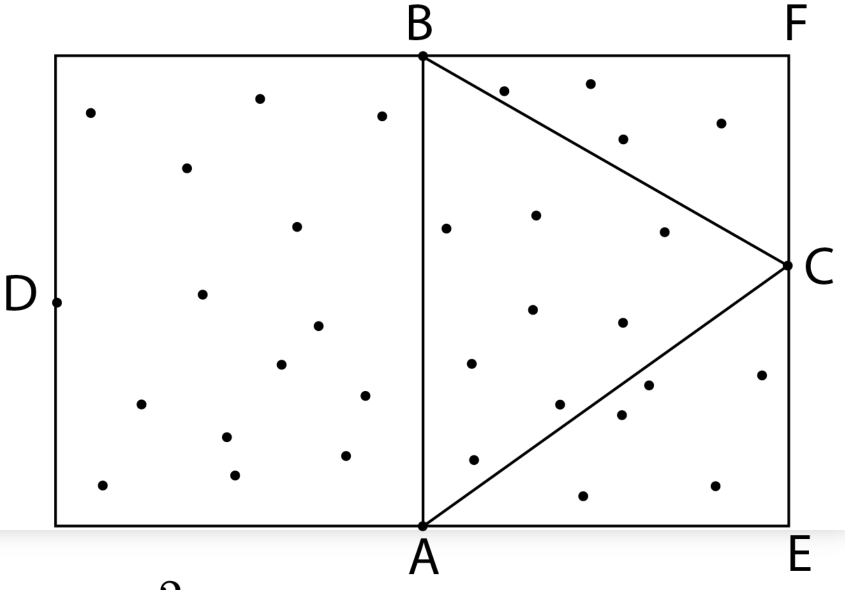


- Average case best scenario $T(n) = O(n) + 2T\left(\frac{n}{4}\right), p = q = \frac{n}{4}$
- So we can not apply Master Theorem directly.
- If f and g are both required to be functions from some unbounded subset of the positive integers to the nonnegative real numbers; then f(x) = O(g(x)) if there exist positive integer numbers M and n_0 such that $f(n) \leq Mg(n)$ for all $n \geq n_0$.[1]
- Given $f(n) = O(n) \leq Mn$, we denote $f'(n) = Mn$. Then

$$T(n) = O(n) + 2T\left(\frac{n}{4}\right) \leq 2T\left(\frac{n}{4}\right) + f'(n) = T'(n)$$

[1]Michael Sipser (1997). Introduction to the Theory of Computation. Boston/MA: PWS Publishing Co. Def.7.2, p.227

Average Case Complexity



Best scenario: O(n)

$$\text{Given } f'(n) = Mn \geq n^{\log_4^2 + \epsilon} \text{ and } 2f'\left(\frac{n}{4}\right) = \frac{Mn}{2} \leq cf'(n) = cMn \text{ for } c = \frac{2}{3}$$

This shows f'(n) satisfies Case 3 in Master Theorem,

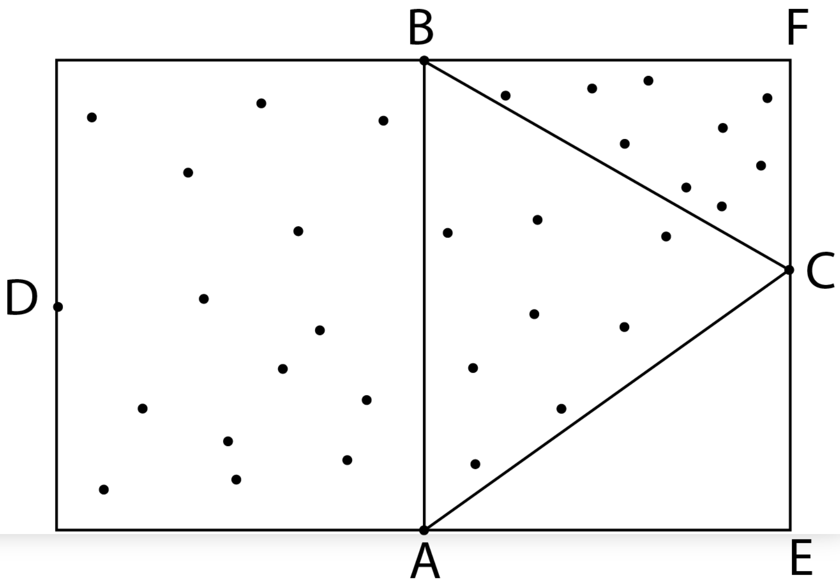
- **Case 3.** If $f(n) = \Omega\left(n^{\log_b a+\epsilon}\right)$ for some $\epsilon > 0$ $\left(\text{and } af\left(\frac{n}{b}\right) \leq cf(n) \text{ for some } c < 1 \text{ for all } n \text{ sufficiently large}\right)$, then $T(n) = \Theta\big(f(n)\big)$.

$$\text{Recall } T'(n) = 2T\left(\frac{n}{4}\right) + f'(n), \text{ so } T'(n) = \Theta(f'(n)) = \Theta(Mn)$$

Therefore, $T(n) \leq T'(n), T(n) = O(n)$

Average Case Complexity

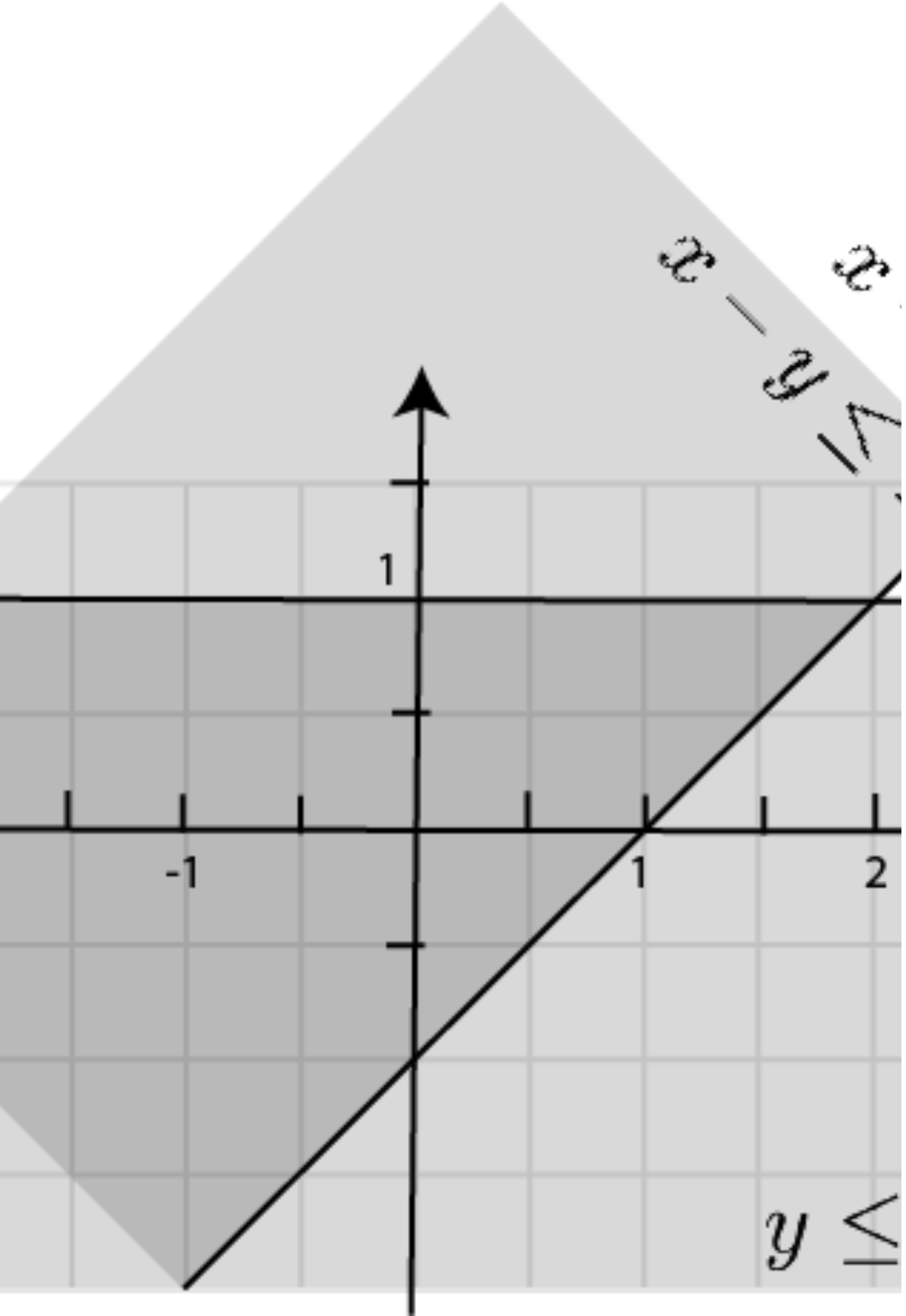
Worst scenario: O(n)



- Worst case Scenario:

$$T(n) = O(n) + T(m) + T(n) = T\left(\frac{n}{2}\right) + O(n), \quad m = \frac{n}{2}, n = 0$$

- Each time fold by half gives complexity O(log n), together O(n).



Incremental Algorithm

- Convex hull again
- Linear programming in 2D
- Kernel of polygon
- Randomized Incremental Algorithm

Incremental Algorithm for Convex Hull

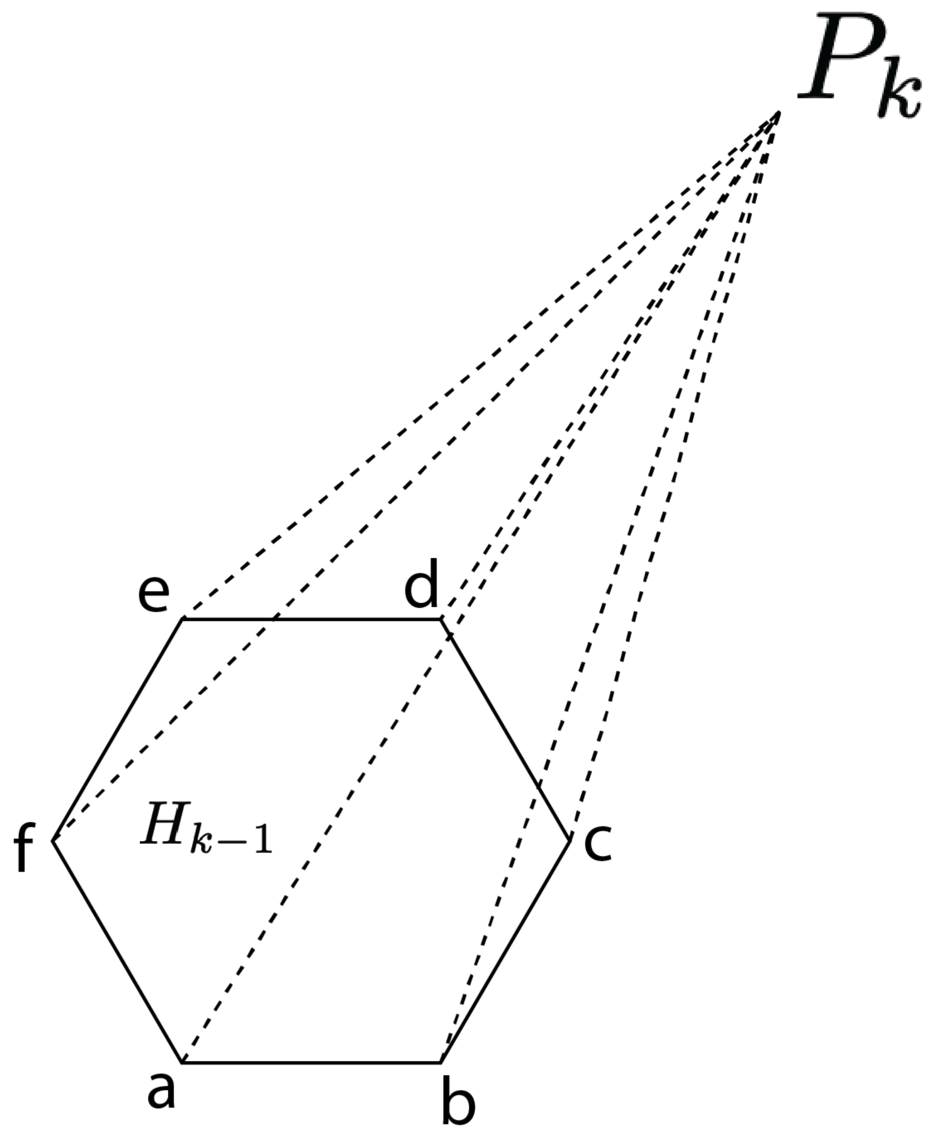
- Starting with three points (first three points in the input).
- These three points form a convex hull.
- Iterate through new points
  - If already in the convex hull: ignore.
  - If not in the convex hull: expand the convex hull to include the new point.

How to update the convex hull?

- $\overrightarrow{bc}$ and $\overrightarrow{cd}$ pointing to opposite direction relate to P_k
- $\overrightarrow{de}$ and $\overrightarrow{ef}$ pointing to opposite direction relate to P_k

So

- Delete points between b and e
- Connect bP_k and eP_k



Linear programming (LP): a method to achieve the best outcome in a mathematical model whose requirements are represented by linear relationships.

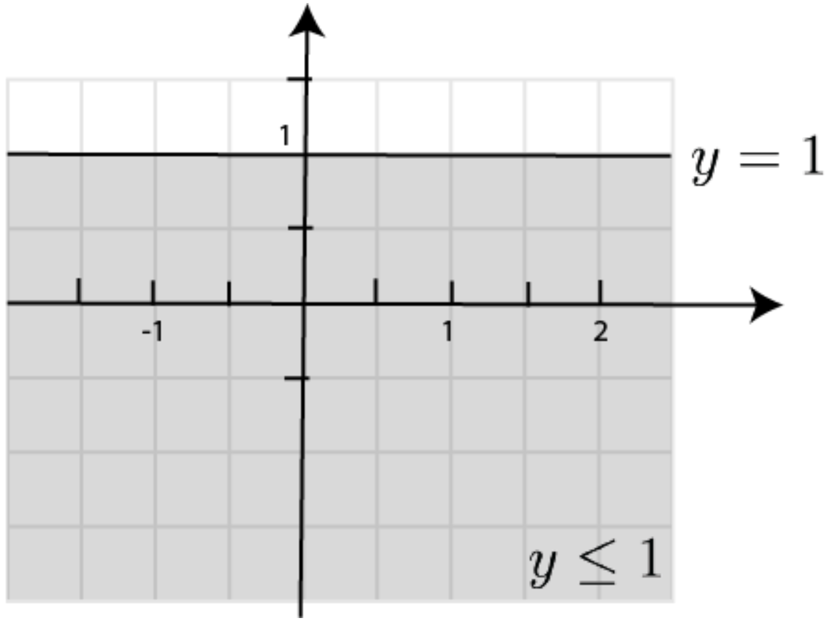
Liner Programming and Reformulation

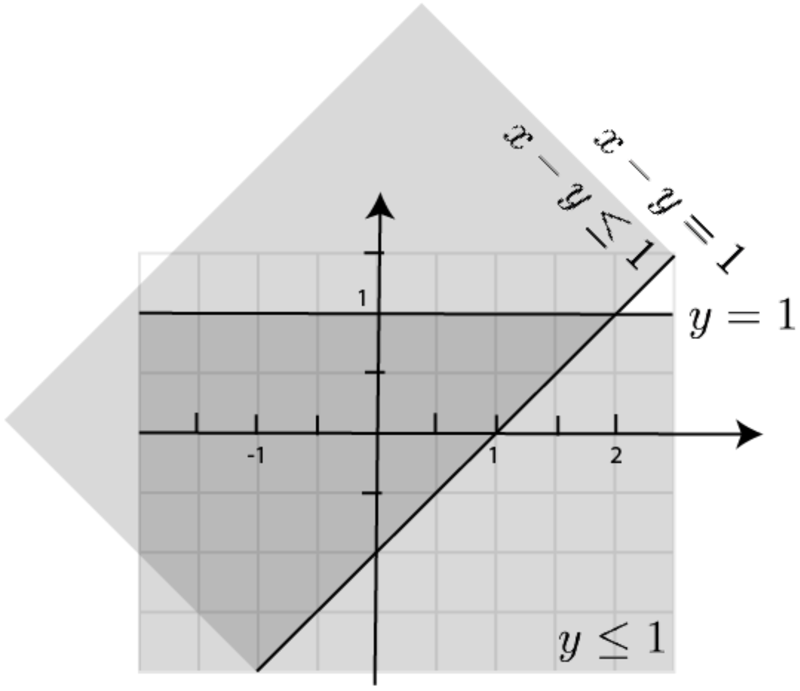
$$\begin{aligned} \text{Maximize } & c_1x_1 + c_2x_2 \\ \text{Constraints } & a_{1,1}x_1 + a_{1,2}x_2 \leq b_1 \\ & a_{2,1}x_1 + a_{2,2}x_2 \leq b_2 \\ & \vdots \\ & a_{n,1}x_1 + a_{n,2}x_2 \leq b_n \end{aligned}$$

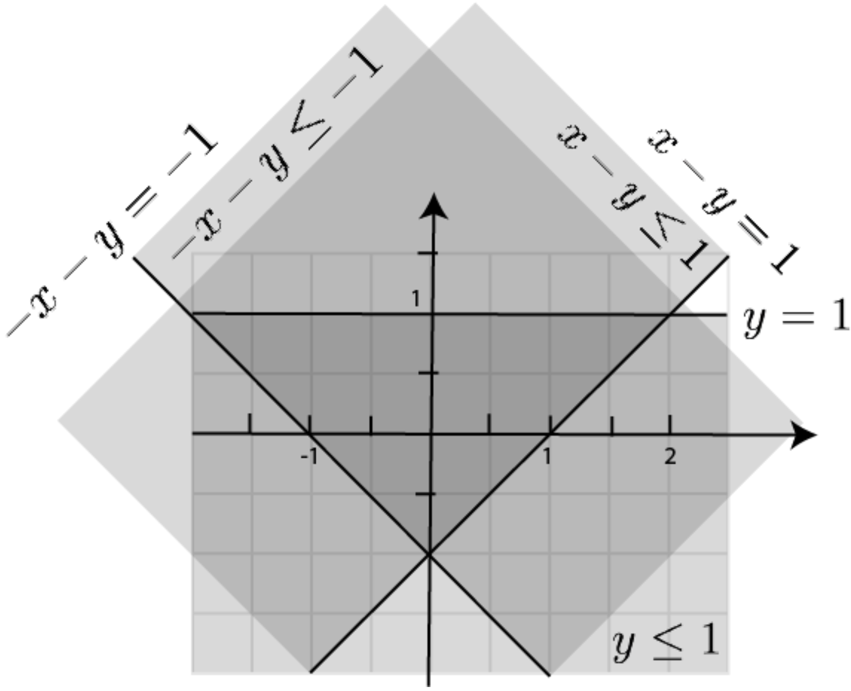
$$\begin{aligned} \text{Line } l_i &: a_{i,1}x_1 + a_{i,2}x_2 = b_i \\ \text{Halfplane } h_i &: a_{i,1}x_1 + a_{i,2}x_2 \leq b_i \\ \text{Inthedirection} &: \vec{c} = (c_1, c_2) \end{aligned}$$

Find the points in the intersection of planes that are the furthest in the direction $\vec{c}$.

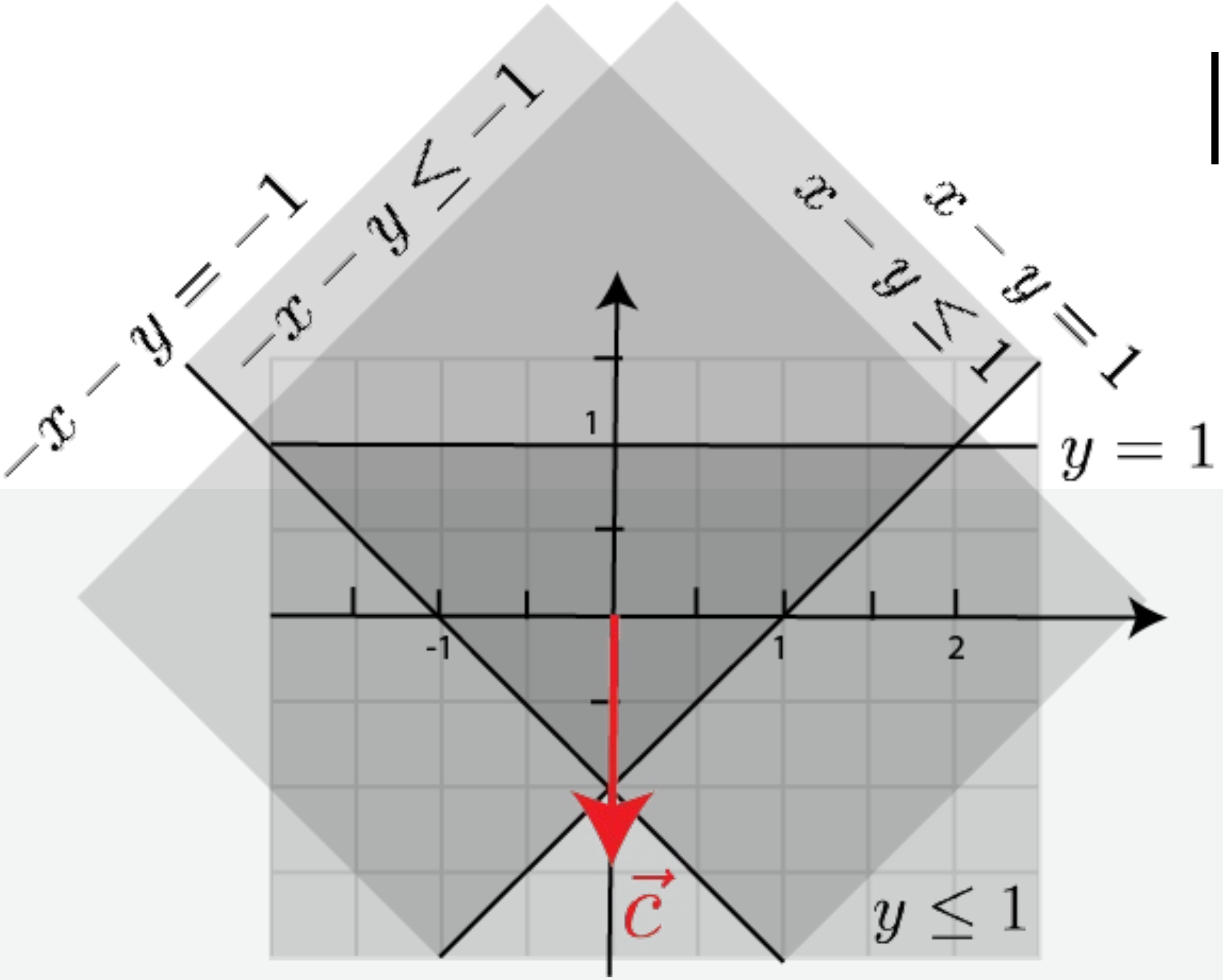
Step 1. Find the points in the intersection of planes.



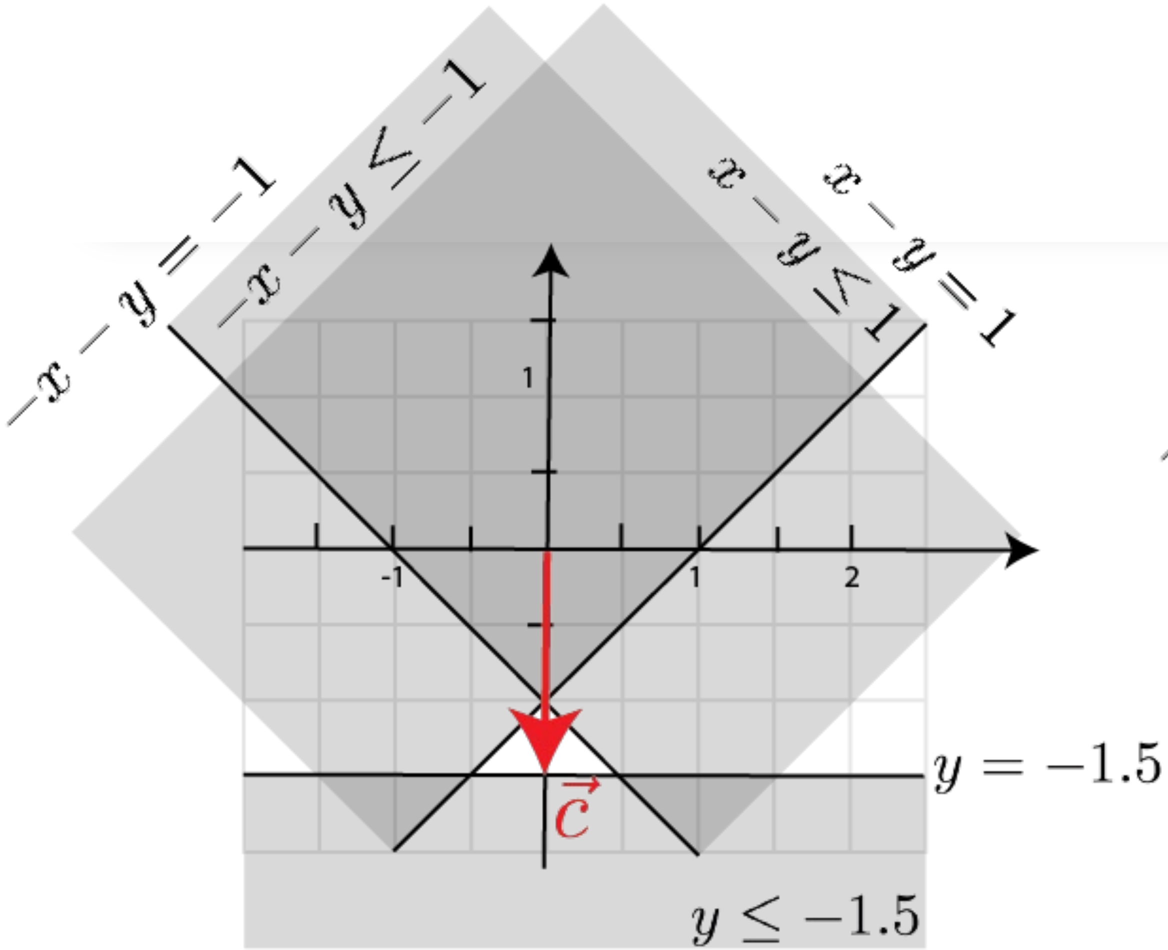




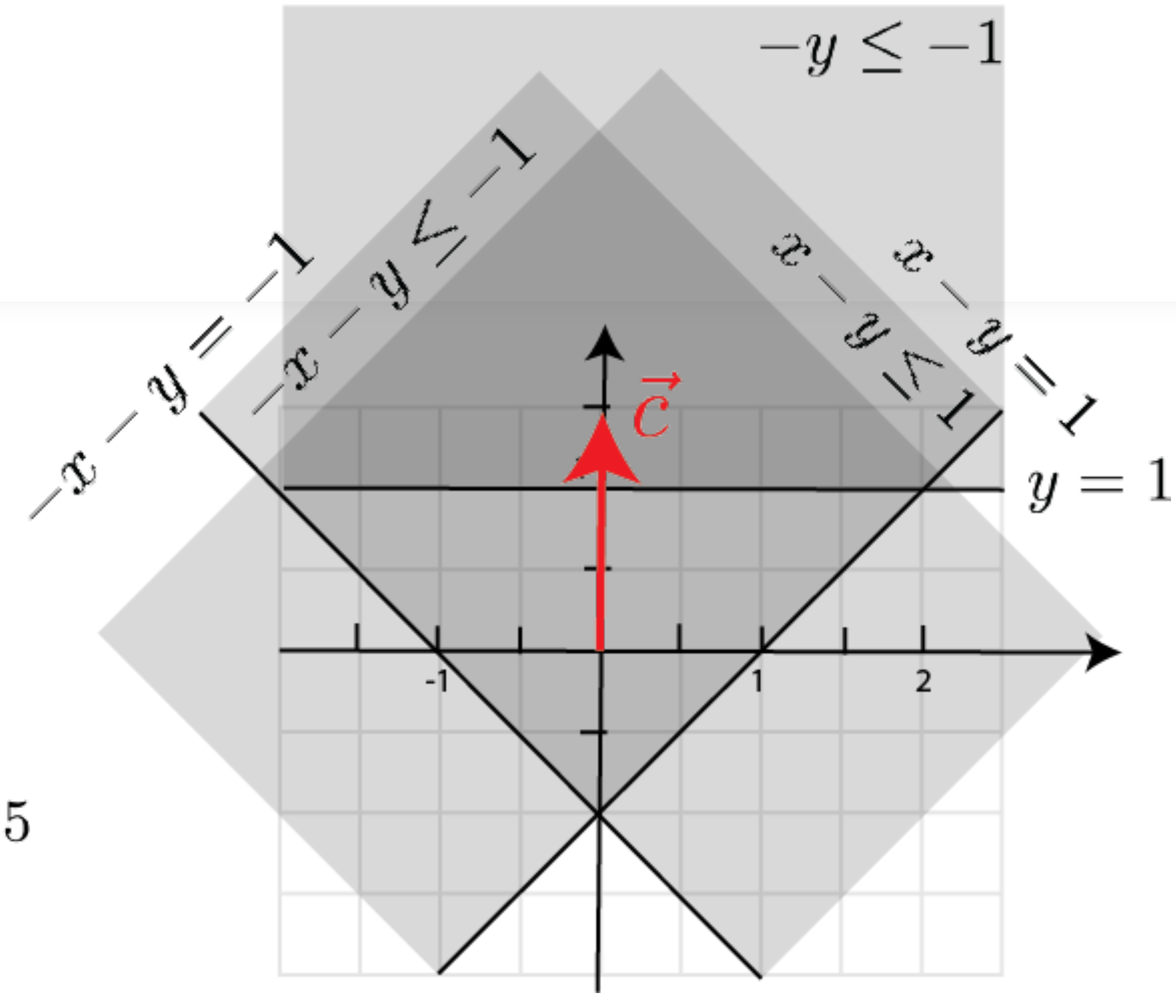
Step 2: Find the points that are the furthest in the direction.



Cases

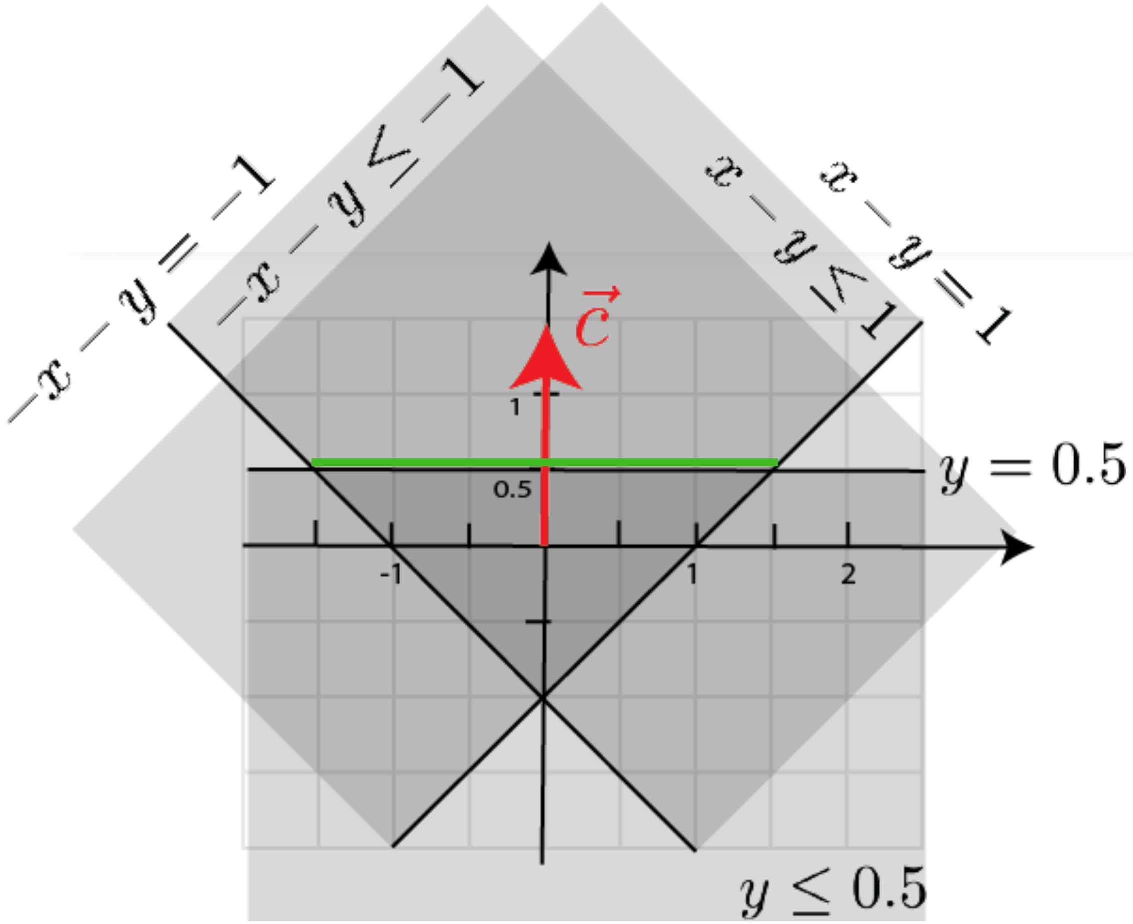


Case 1. No solution.

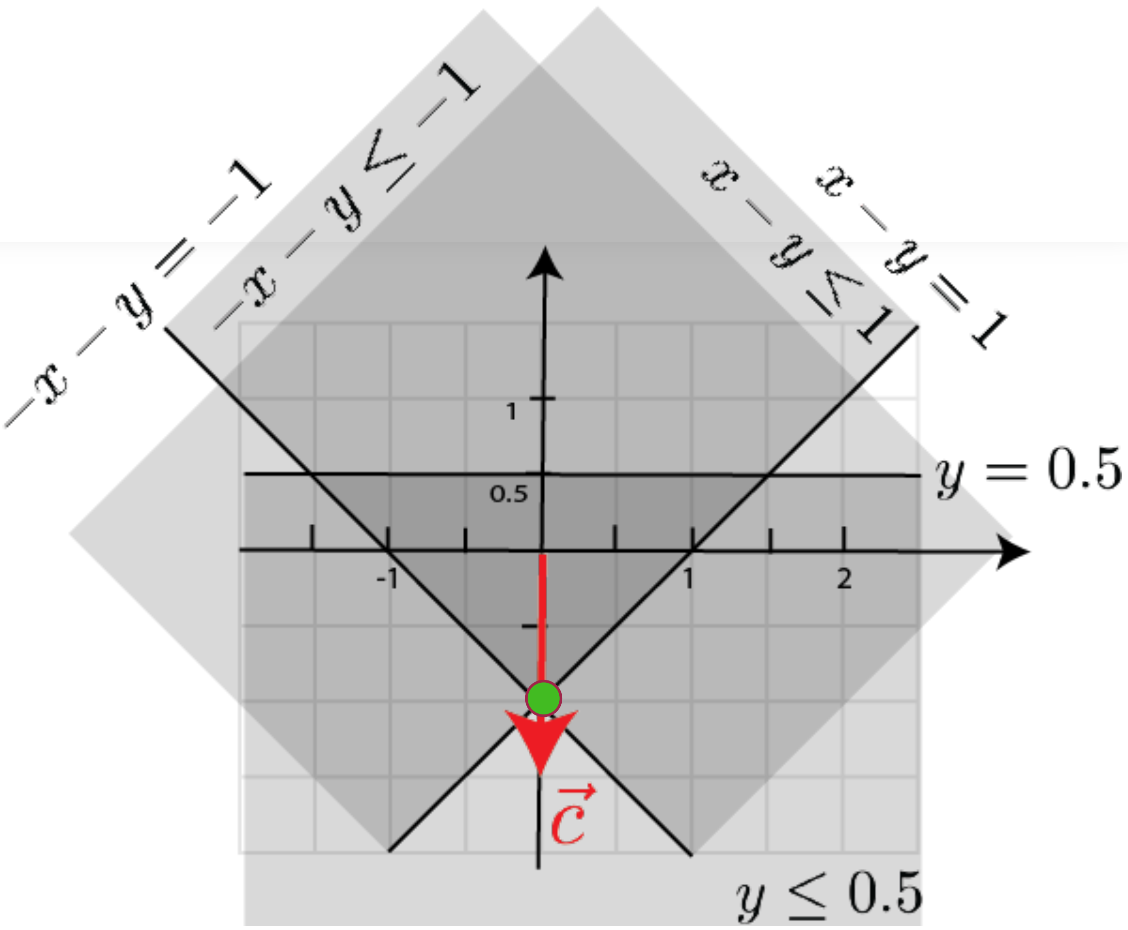


Case 2. Solution is ∞.

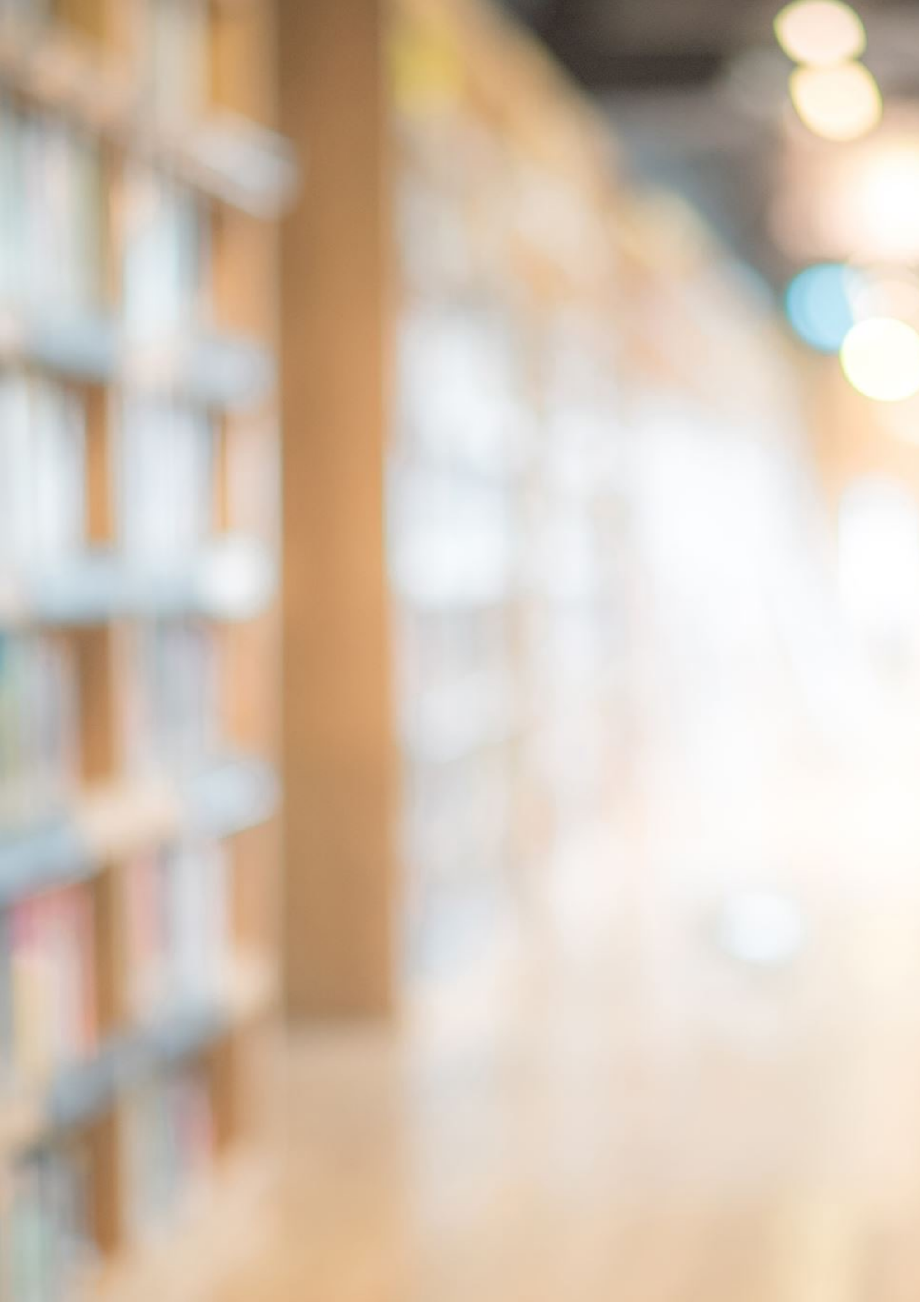
Cases



Case 3. Infinitely many solution.

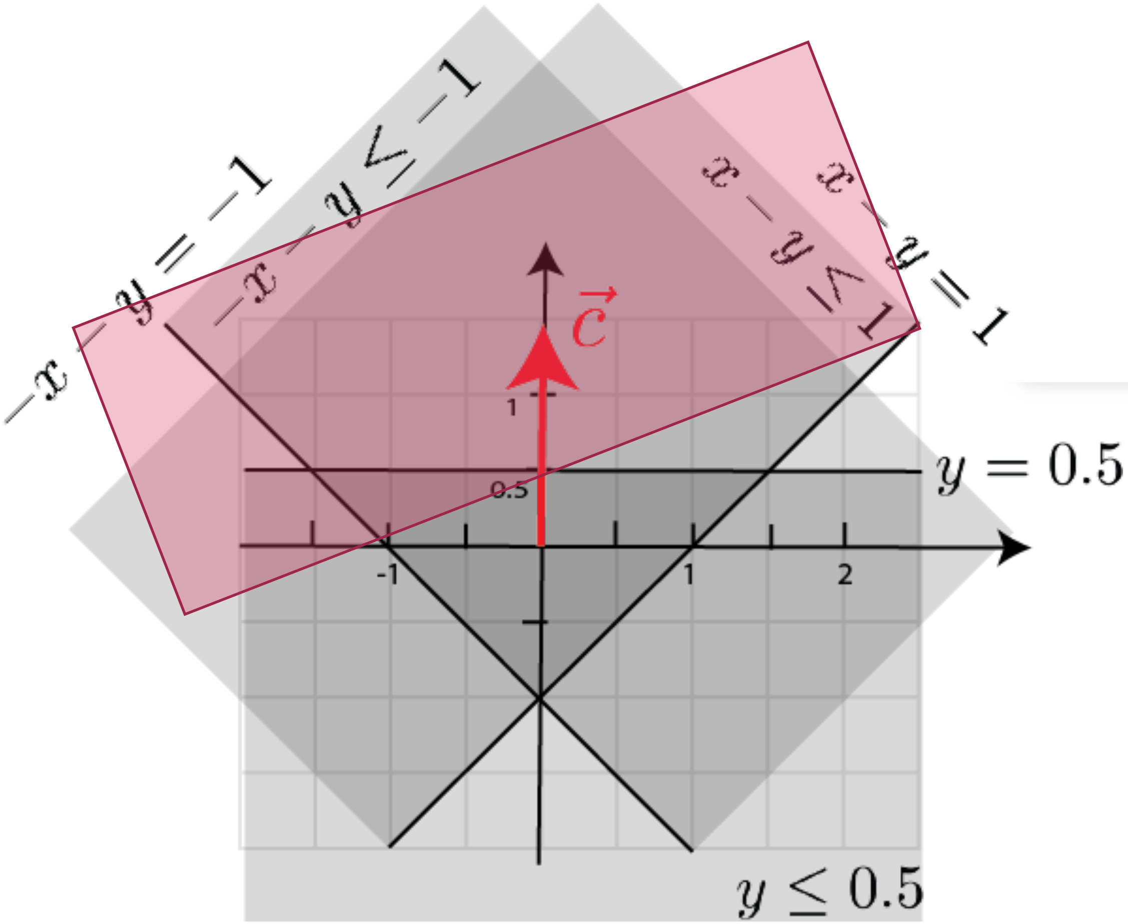


Case 4. Unique solution.



Discussions

- To avoid infinite solution, we bound at [-M, M] x [-M, M].
- Can use Divide-and-Conquer and intersect convex hulls to achieve O(n log n) complexity.
- Best possible solution for finding the kernel of polygon (set of points that can see all boundary).

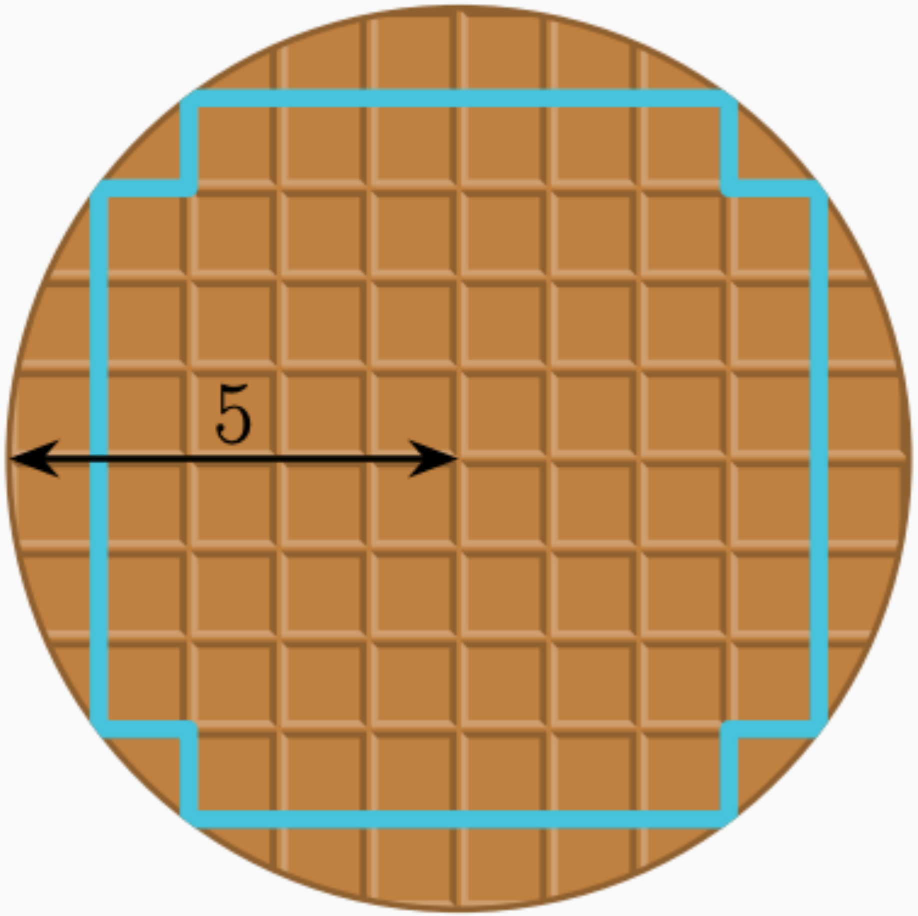


Incremental Algorithm

We are looking for the farthest point in the direction $\vec{c}$.
The other boundaries seems does not matter.

- Set V to be the (set of) solution(s)
- Adding a new half-plane.
- If V is contained in the new half-plane, pass.
- Otherwise update V to be contained in the new half-plane. Observation: V is on the new line.
- Randomize the lines to make it most efficient.

Circular Caramel Cookie



- For a fixed radius r, we can determine the number of whole unit squares that fit in the circle.
- Determine how many squares fit in each column using the Pythagorean Theorem, $O(\sqrt{s})$
- Use binary search to find the solution. Total time $O(\log s \cdot \sqrt{s})$

The Northwestern Europe Regional Contest (NWERC), 2022

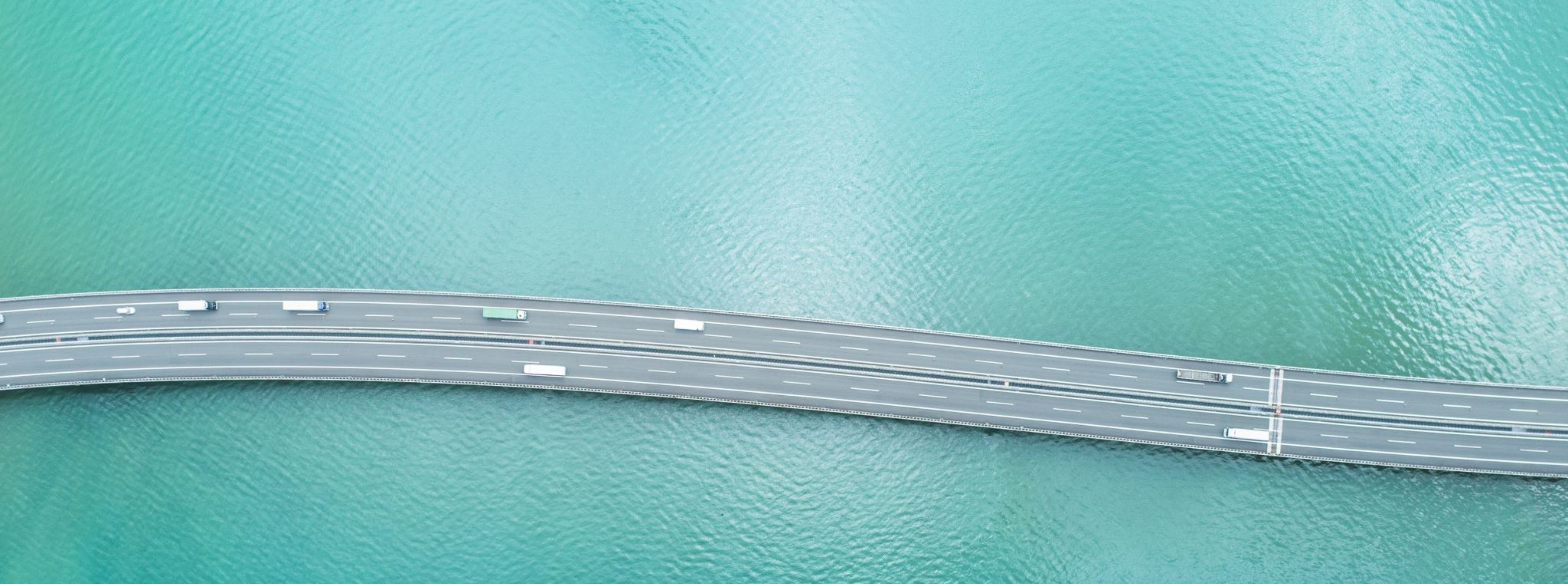
Trainer’s Choice

2019-2020 ICPC North America Championship, Bomas: https://open.kattis.com/problems/bomas

Stars in a Can: https://open.kattis.com/problems/starsinacan

NWERC 2002, The Picnic: https://archive.algo.is/icpc/nwerc/2002/Problem.pdf

Enclosure: https://open.kattis.com/problems/enclosure



Thank you

Yingyingwu.io