The optimality of Geometric data Structures and Algarithms

John Iacano

Université Libre de Bruxelles

Warm up: 1D

Given N points P, for each report the point to its left / right.

AKA sort

OPTIMAL

OPTIMAL

- Try what we did for BST

OPTIMAL

- Try what we did for BST
- Let Time(A, X) : Time for A to sort $x_1, x_2 \cdots x_n$

OPTIMAL

- Try what we did for BST
- Let Time(A, X) : Time for A to sort $x_1, x_2 \cdots x_n$
- Define $OPT(X) = \min_{A} Time(A, X)$

OPTIMAL

- Try what we did for BST
- Let $\text{Time}(A, X)$: Time for A to sort $x_1, x_2 \cdots x_n$
- Define $\text{OPT}(X) = \min_A \text{Time}(A, X)$
- Is there some A_{OPT}, $c > 0$ s.t. ?

$$\forall x \quad \text{Time}(A_{OPT}, X) \leq c \cdot \text{OPT}(x) \text{ .}$$

OPTIMAL

- Try what we did for BST
- Let $\text{Time}(A, X)$: Time for A to sort $x_1, x_2 \cdots x_n$
- Define $OPT(X) = \min_A \text{Time}(A, X)$
- Is there some A_{OPT}, $c > 0$ s.t. $\forall x \quad \text{Time}(A_{OPT}, x) \leq c \cdot OPT(x)$?

$X =$	x_1	x_2	x_3	x_4
3	1	4	2

A: if $x_2 < x_4$ and $x_4 < x_1$ and $x_1 < x_3$
print (x_2, x_4, x_1, x_3)
else run mergesort

OPTIMAL

- Try what we did for BST
- Let $\text{Time}(A, X)$: Time for A to sort $x_1, x_2 \cdots x_n$
- Define $OPT(X) = \min_A \text{Time}(A, X)$
- Is there some A_{OPT}, $c > 0$ s.t.

$$\forall x \quad \text{Time}(A_{OPT}, x) \leq c \cdot OPT(x) \; ?$$

$X = \underset{x_1}{3}\ \underset{x_2}{1}\ \underset{x_3}{4}\ \underset{x_4}{2}$

A: if $x_2 < x_4$ and $x_4 < x_1$ and $x_1 < x_3$
print (x_2, x_4, x_1, x_3)
else run mergesort

Thus $OPT(X) = O(N)$

But every comparison-based Alg A needs $\log n! = \Theta(N \log N)$ time on some inputs of size N

Runtime?

- $O(n \log n)$
- Optimal? What does that mean?
- $\Omega(n \log n)$ if comparisons
- $\Omega(n \log n)$ if polynomial tests (Ben-Or)
- $\Theta\left(n \sqrt{\frac{\log N}{\log \log N}}\right)$ if RAM model with same assumptions

What does this mean?

Can we do better?

- If points are given in sorted order only takes $O(N)$ time to "verify"

Can we do better?

- If points are given in sorted order only takes $O(N)$ time to "verify"

- Actually for any input permutation we could make an algorithm that takes $O(N)$ time on this input

Can we do better?

- If points are given in sorted order only takes $O(N)$ time to "verify"

- Actually for any input permutation we could make an algorithm that takes $O(N)$ time on this input

- No natural way to pick a few sequences to be fast on

Worst-Case Runtime

Sorting takes $\Omega(N \log N)$ in the comparison model means:

There is a $c > 0$, $n_0 > 1$

For all comparison-based algs A

For all $n \geq n_0$

There is a set of points P

The time to sort P with $A \geq cn \log n$

Can we do better?

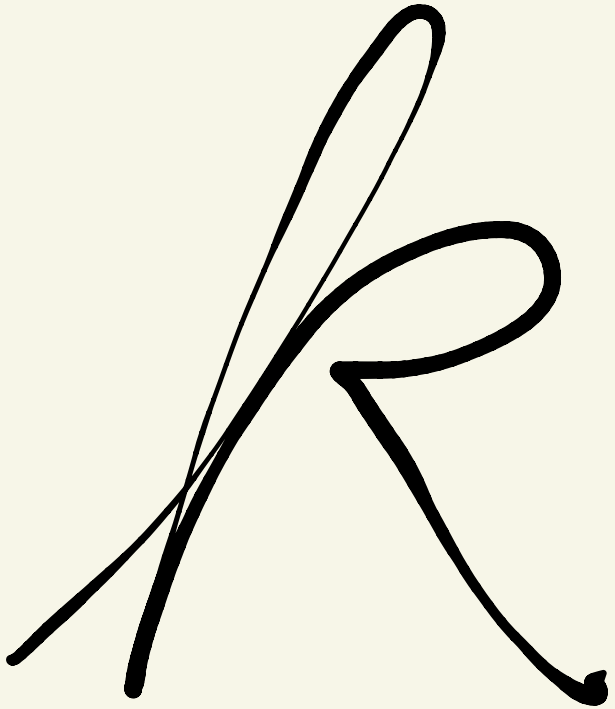
- Possibly, if instead of using "N" only we can express the runtime as a function of how easy / hard an instance is.

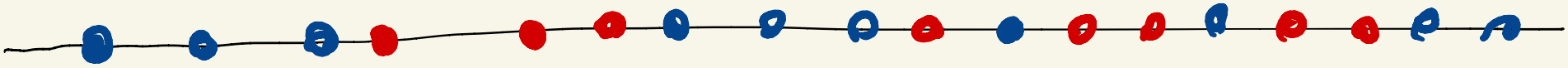
Can we do better?

- Possibly, if instead of using "N" only we can express the runtime as a function of how easy/hard an instance is.

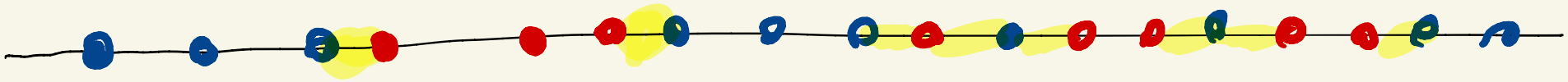
- Easy instances should not be punished by the existance of hard instances.

PART I:

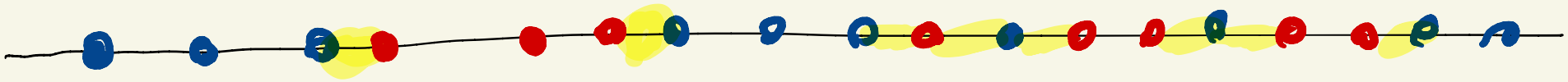




Report adjacent Red-Blue Pairs



Report adjacent Red-Blue Pairs



Has same $\Omega(N \log N)$ worst-case runtime.

What if output has size 1?

What if output has size 1?

if max(RED) < min(BLUE)

return [(max(RED), min(BLUE))]

else ...

What if output has size 1?

if max(RED) < min(BLUE)
    return [(max(RED), min(BLUE))]
else ....

O(N) if output size 1
O(N log N) in general

what about a general function of output size

What if output has size 1?

if max(RED) < min(BLUE)
    return [(max(RED), min(BLUE))]
else ....

O(N) if output size 1
O(N log N) in general

What if output has size 1?

Works no matter what order the input is given

if max(RED) < min(BLUE)

return [(max(RED), min(BLUE))]

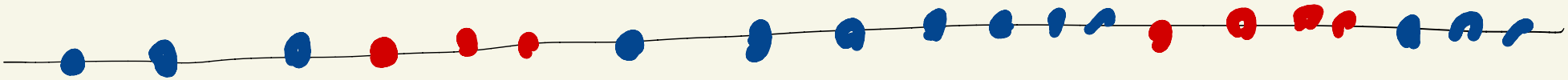
else ....

O(N) if output size 1

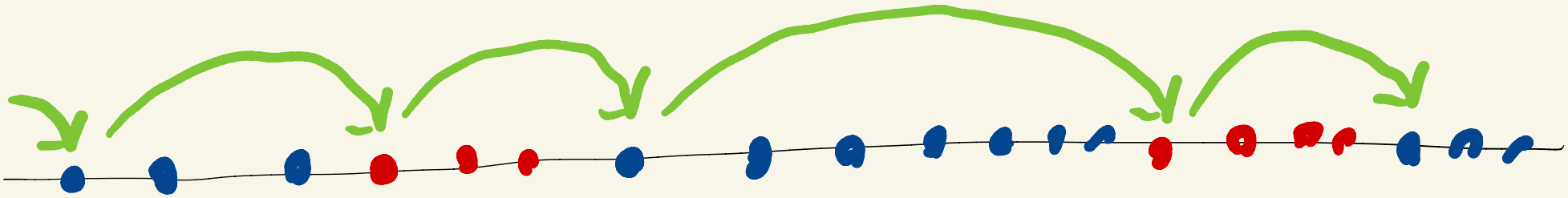
O(N log N) in general

what about a general function of output size

A proposed algorithm

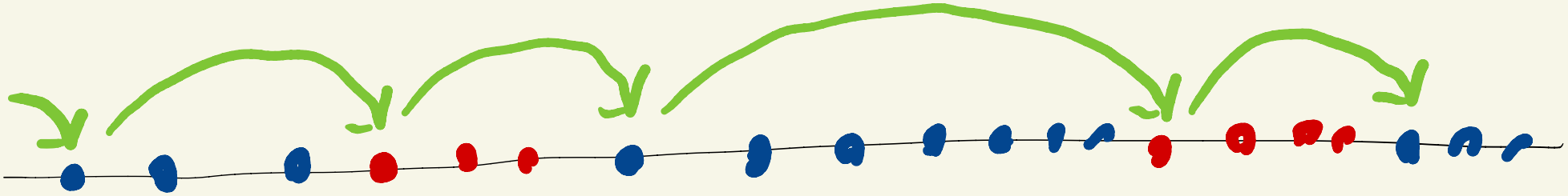


A proposed algorithm



"Find next in a given color"

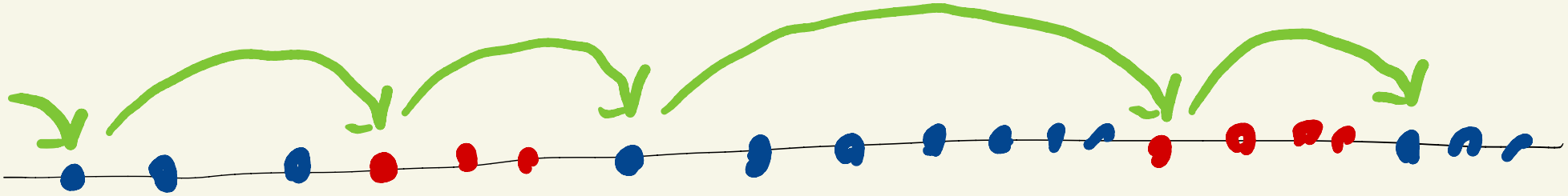
A proposed algorithm



"Find next in a given color"

Let K = # of hops / connected components

A proposed algorithm

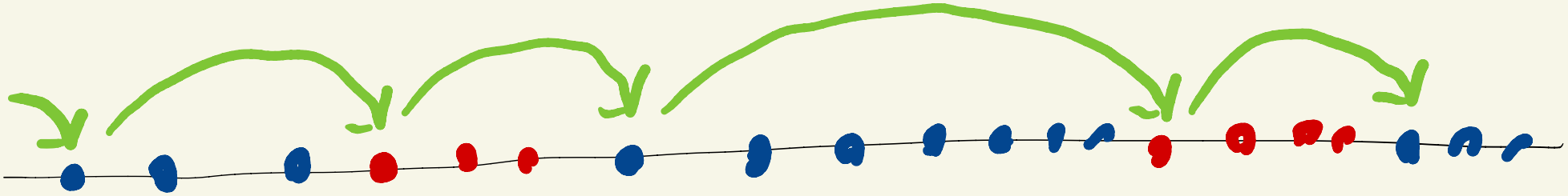


"Find next in a given color"

Let K = # of hops / connected components

- In each color split into $\frac{N}{K}$ groups of K

A proposed algorithm

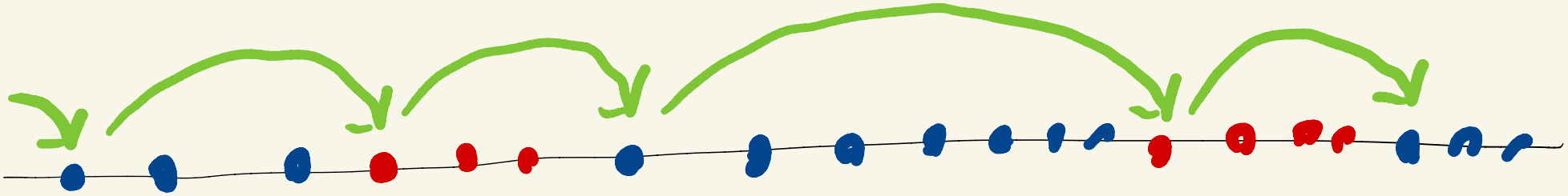


"Find next in a given color"

Let K = # of hops / connected components

- In each color split into $\frac{N}{K}$ groups of K
- Sort each group: $\frac{N}{K} \cdot K \log K = O(N \log K)$

A proposed algorithm



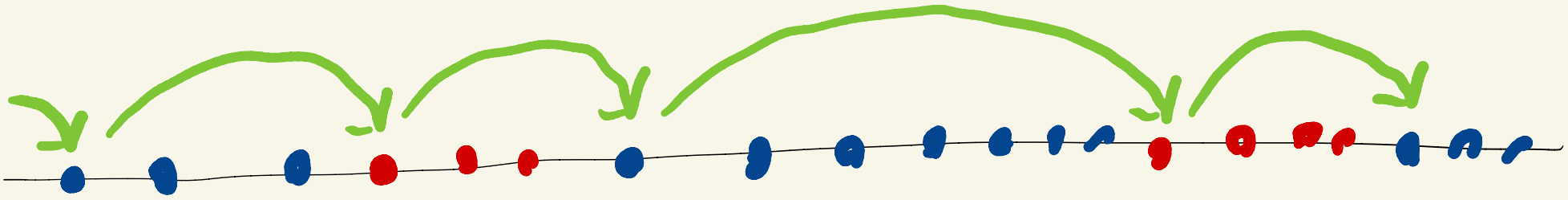
"Find next in a given color"

Let K = # of hops / connected components

- In each color split into $\frac{N}{K}$ groups of K
- Sort each group: $\frac{N}{K} \cdot K \log K = O(N \log K)$

- For each find next, binary search in each group

A proposed algorithm



"Find next in a given color"

Let K = # of hops / connected components

- In each color split into $\frac{N}{K}$ groups of K
- Sort each group: $\frac{N}{K} \cdot K \log K = O(N \log K)$

- For each find next, binary search in each group

groups: $\frac{N}{K}$ · sort: $K \log K$ + searches: K · groups: $\frac{N}{K}$ · cost per search: $\log K$ $= O(N \log K)$

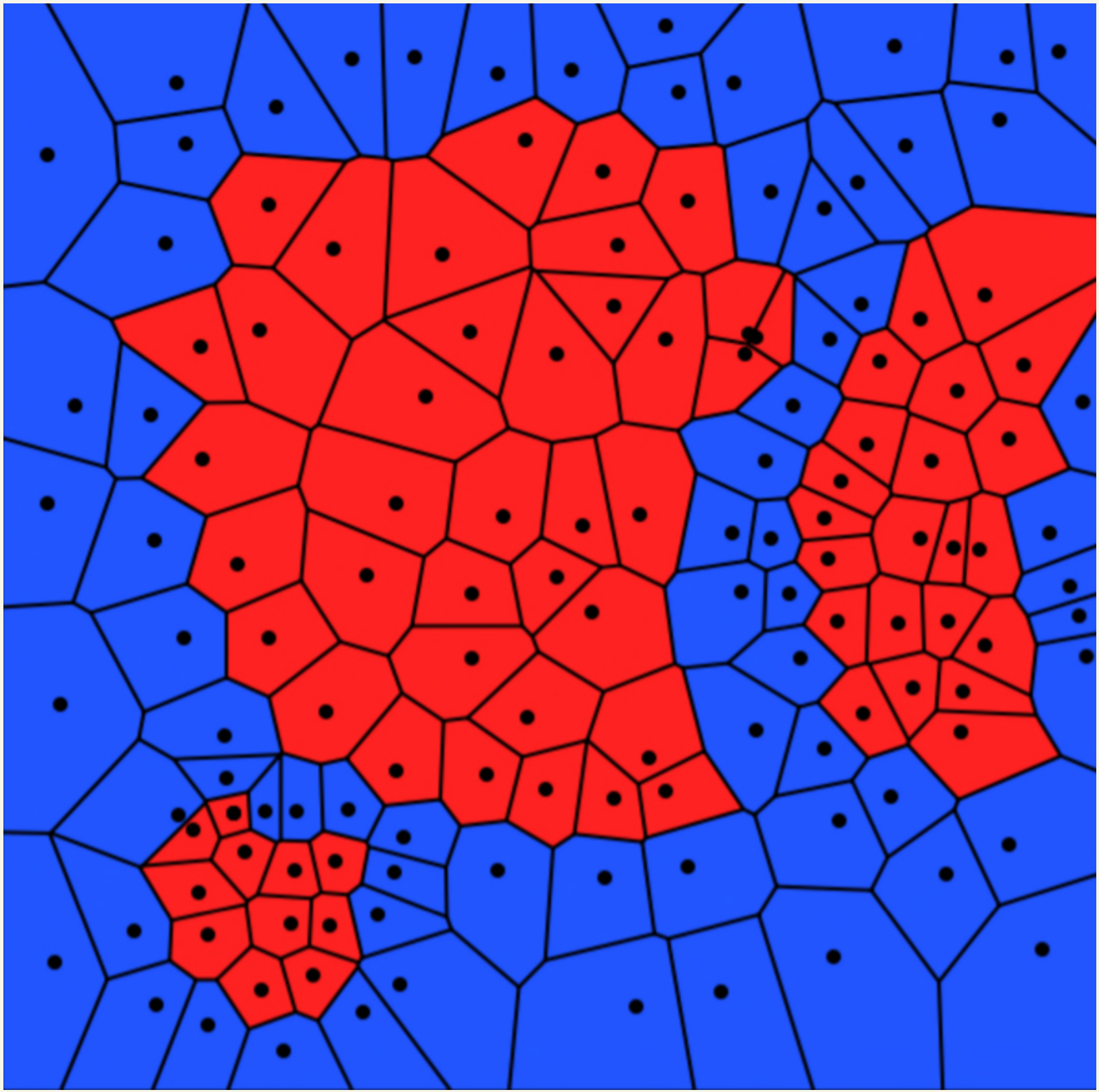
Runtime: $\frac{N}{K} \cdot K \log K + K \cdot \frac{N}{K} \cdot \log K = O(N \log K)$

What is K?

- Try $k_i = 2^{2^i}$
- Repeat until it works

$$\sum_{i=1}^{\lceil \log \log k \rceil} N \log 2^{2^i} = N \sum_{i=1}^{\lceil \log \log k \rceil} 2^i = O(N \log k)$$

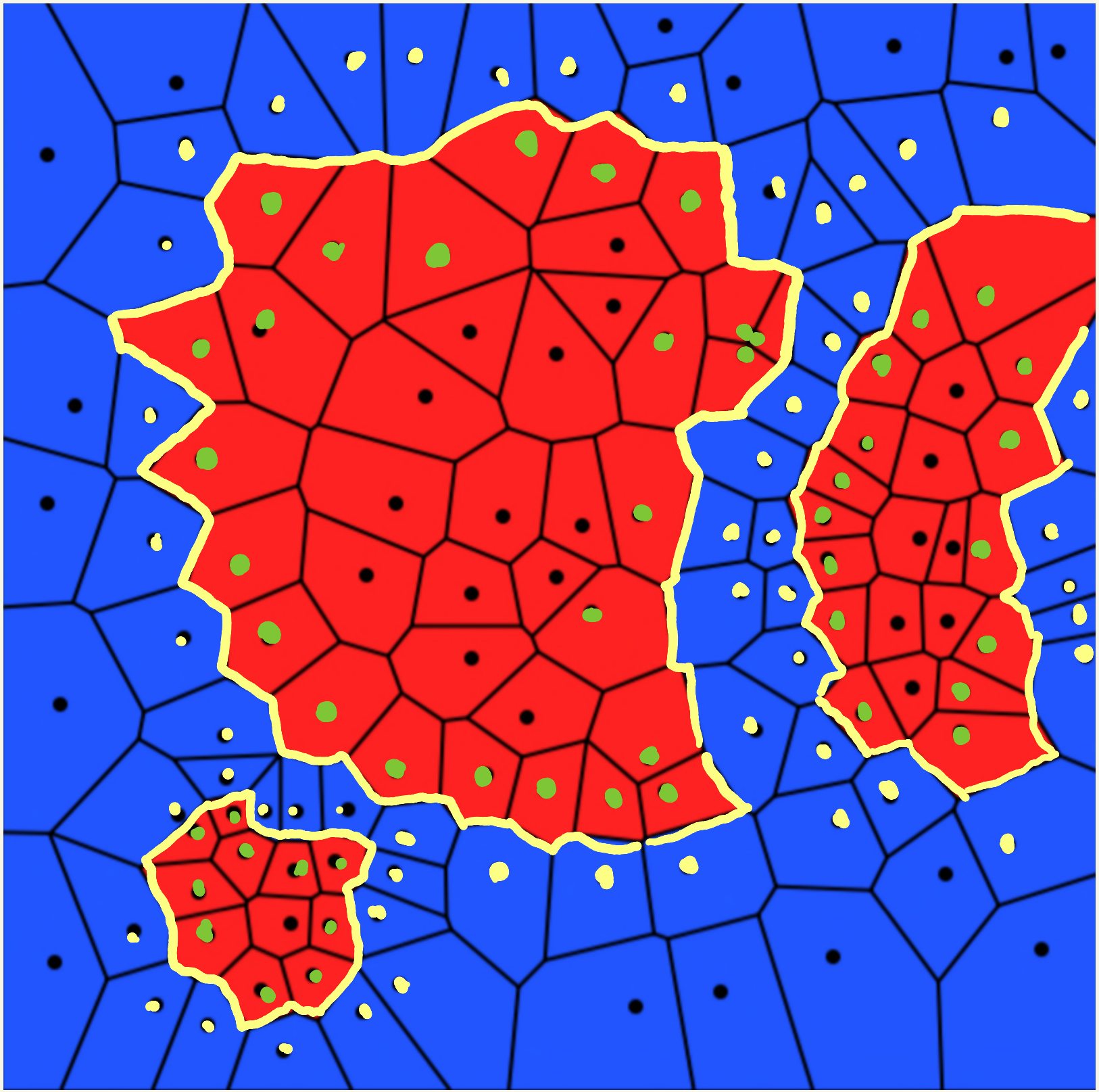
GEOMETRY



GEOMETRY

GEOMETRY

Find all points on the red-blue boundary

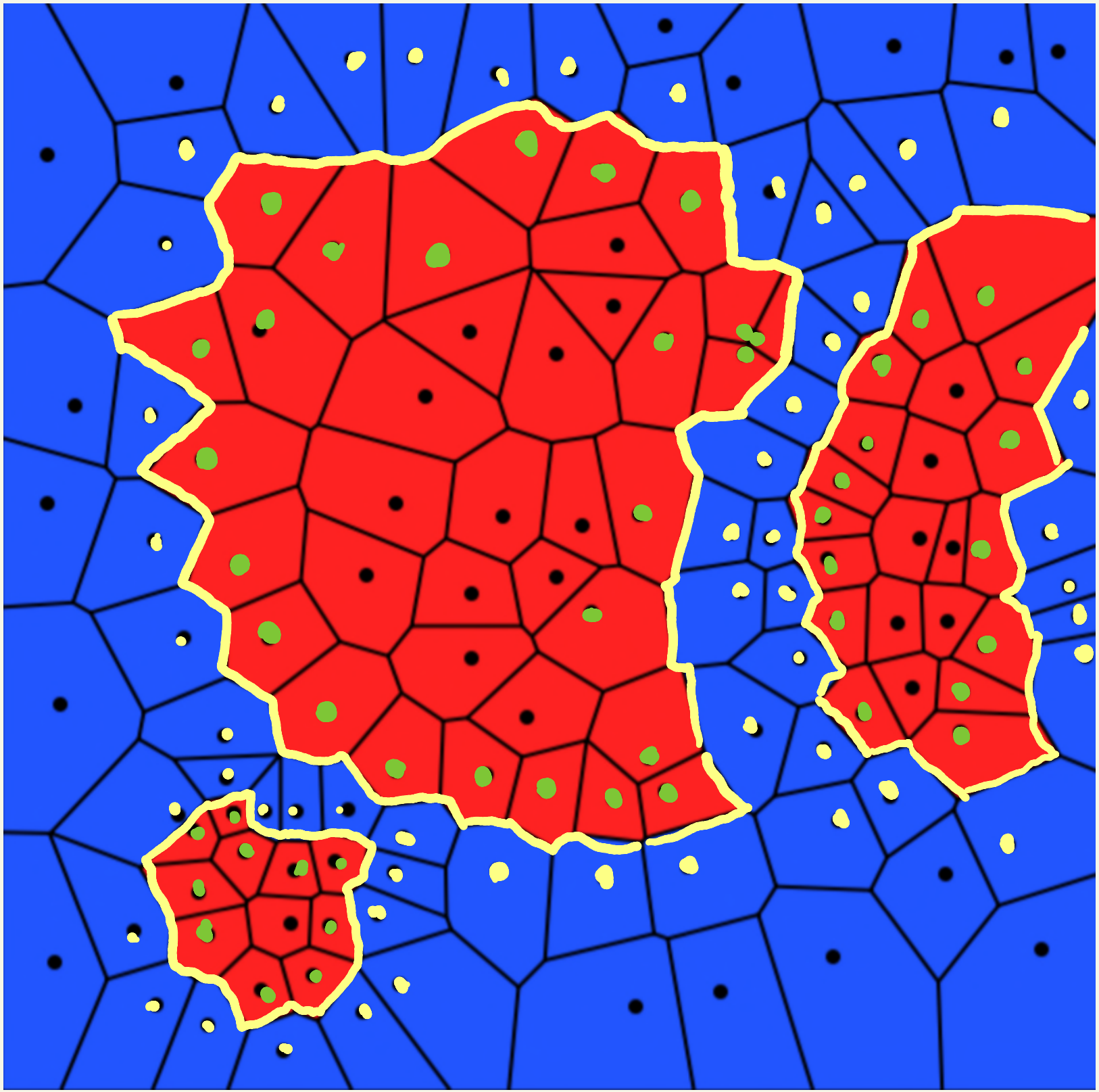


GEOMETRY

$O(N \log N)$ Trivial

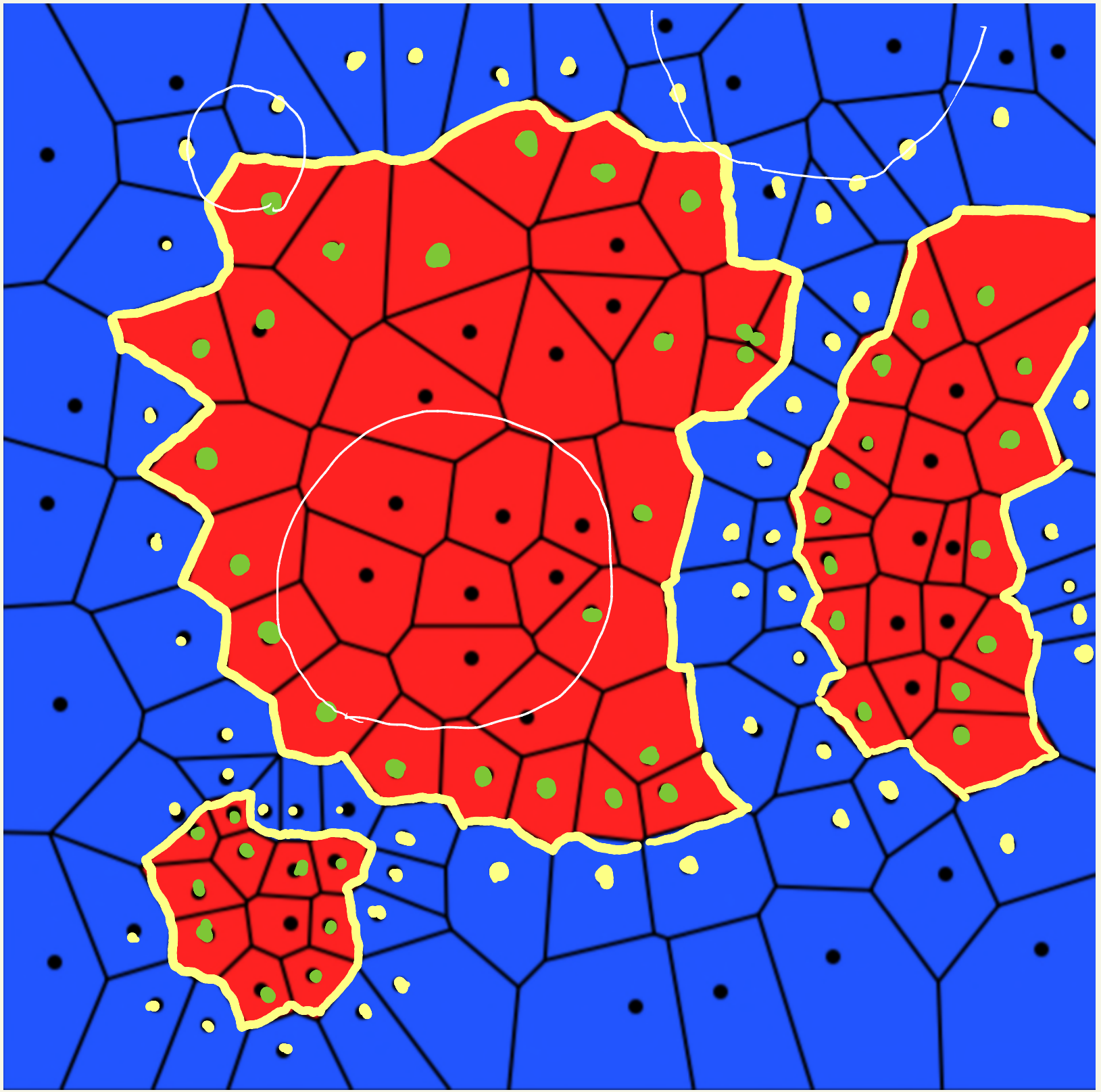
Better??

Find all points on the red-blue boundary



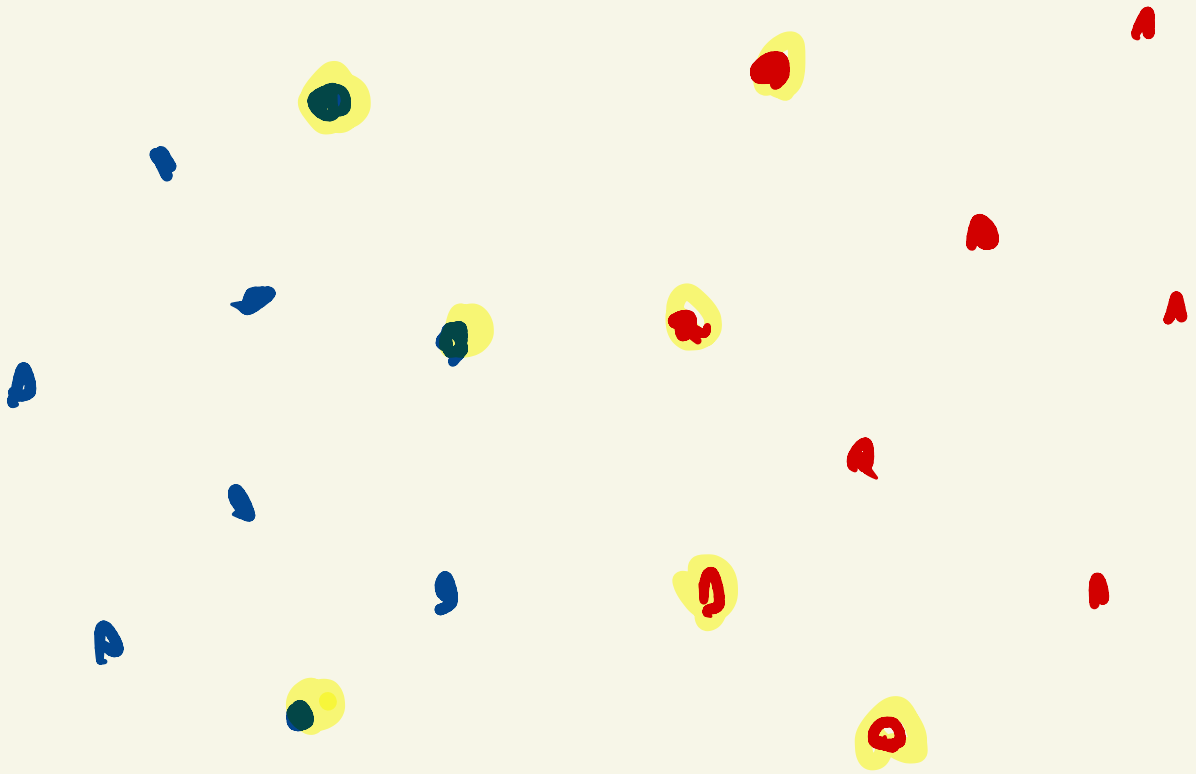
Certificate of answer

* of original point set



"Pivot" Operation

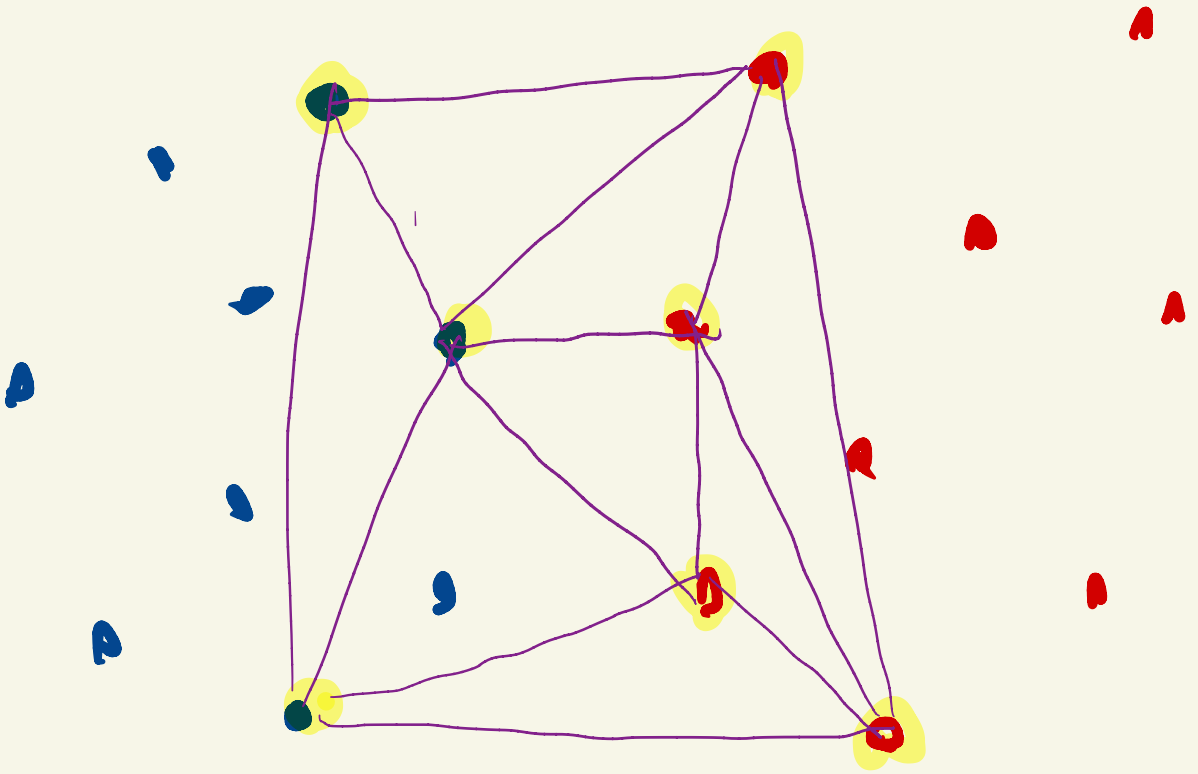
Given a possible solution verify it or find a point that needs to be added



"Pivot" Operation

Given a possible solution verify it or find a point that needs to be added

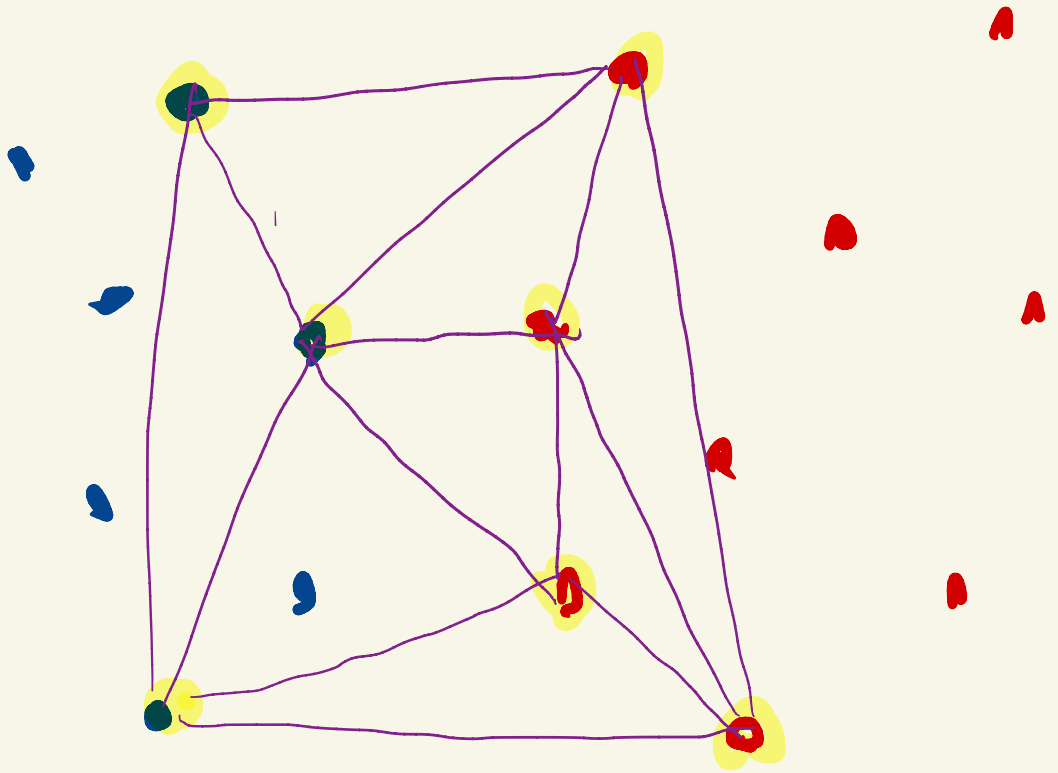
- Construct Delaunay



"Pivot" Operation

Given a possible solution verify it or find a point that needs to be added

- Construct Delaunay
- Check all for emptyness using pivot.

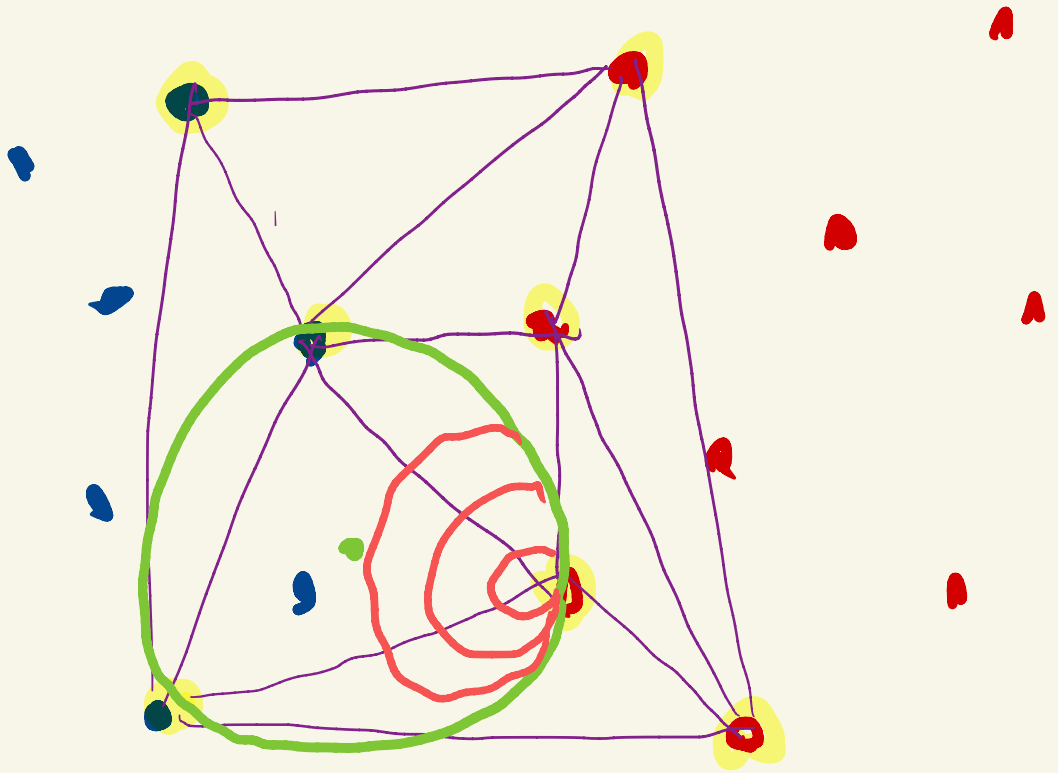


Every red, check for blue, every blue, check for red

"Pivot" Operation

Given a possible solution verify it or find a point that needs to be added

- Construct Delaunay
- Check all for emptyness using pivot.



Every red, check for blue, every blue, check for red

Algorithm to find points $D \subseteq P$ where D define the decision Boundary

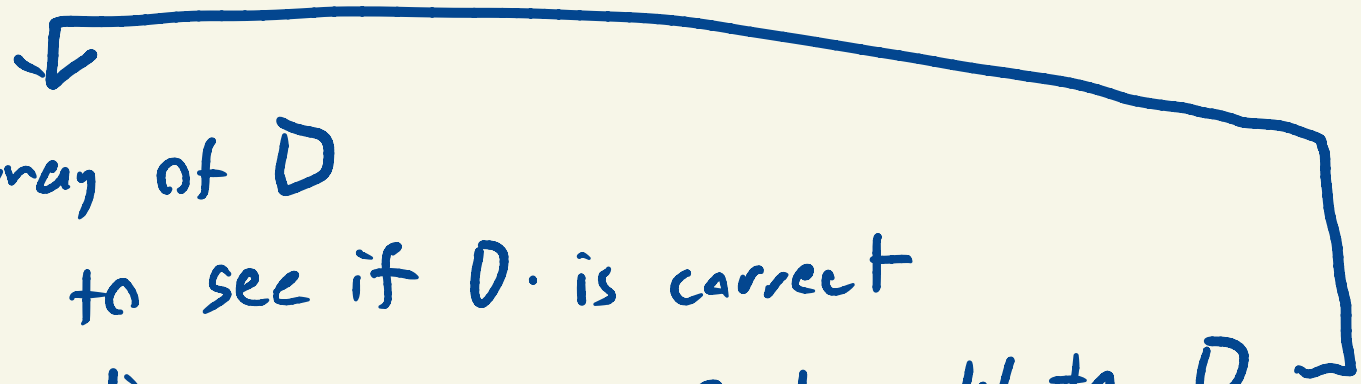
Algorithm to find points $D \subseteq P$ where D define the decision Boundary

$D = \emptyset$

Compute Delaunay of D

Do $|D|$ pivots to see if D is correct

if not we discover some p to add to D,



Algorithm to find points $D \subseteq P$ where D define the decision Boundary

$D = \emptyset$

Compute Delaunay of D

Do $|D|$ pivots to see if D is correct

if not we discover some p to add to D,

Analysis: Let $K = |D|$

$$\sum_{i=1}^{K} (\text{Delaunay of } i \text{ points} + i \cdot \text{cost of pivot})$$

Algorithm to find points $D \subseteq P$ where D define the decision Boundary

$D = \emptyset$

Compute Delaunay of D

Do $|D|$ pivots to see if D is correct

if not we discover some p to add to D,

Analysis: Let $K = |D|$

$$\sum_{i=1}^{K} (\text{Delaunay of } i \text{ points} + i \cdot \text{cost of pivot})$$

$$\leq K \cdot K \log K + K \cdot K \cdot \text{cost of pivot}$$

Cost of pivot

preprocess	Query
$O(N \log N)$	$O(\log N)$

[Dobkin Kirkpatrick 1983]

Cost of pivot	
preprocess	Query
$O(N \log N)$	$O(\log N)$

[Dobkin Kirkpatrick 1983]

$O(N \log d)$	$O(\frac{N}{d} \log d)$

[Chan 1996]

$k^2 \log k + k^2 \cdot$ pivot + create pivot struct

Cost of pivot	
preprocess	Query
$O(N \log N)$	$O(\log N)$

[Dobkin Kirkpatrick 1983]

$O(N \log \alpha)$	$O(\frac{N}{\alpha} \log \alpha)$

[Chan 1996]

))

$k^2 \log k + k^2 \cdot \text{pivot} + \text{create pivot struct}$

$k^2 \log k + k^2 \cdot \frac{N}{\alpha} \log \alpha + N \log \alpha$

Cost of pivot	
preprocess	Query
$O(N \log N)$	$O(\log N)$

[Dobkin Kirkpatrick 1983]

$O(N \log \alpha)$	$O(\frac{N}{\alpha} \log \alpha)$

[Chan 1996]

$K^2 \log K + K^2 \cdot \text{pivot} + \text{create pivot struct}$

$K^2 \log K + K^2 \cdot \frac{N}{\alpha} \log \alpha + N \log \alpha$

$\alpha = K^2$

$K^2 \log K + N \log K$

Cost of pivot	
preprocess	Query
$O(N \log N)$	$O(\log N)$
[Dobkin Kirkpatrick 1983]	
$O(N \log \alpha)$	$O(\frac{N}{\alpha} \log \alpha)$
[Chan 1996]	

$K^2 \log K + K^2 \cdot$ pivot + create pivot struct

Cost of pivot	
preprocess	Query
$O(N \log N)$	$O(\log N)$

[Dobkin Kirkpatrick 1983]

$K^2 \log K + K^2 \cdot \frac{N}{\alpha} \log \alpha + N \log \alpha$

$O(N \log \alpha)$	$O(\frac{N}{\alpha} \log \alpha)$

[Chan 1996]

$\alpha = K^2$

$K^2 \log K + N \log K$

$= O(N \log K)$

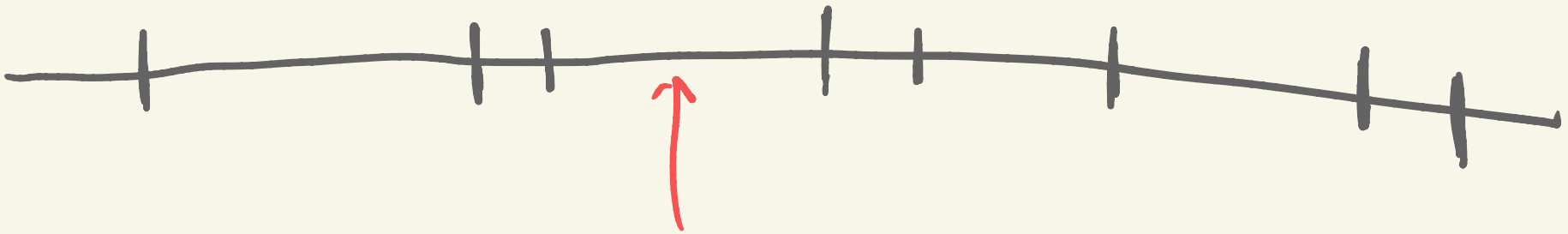
when $K^2 < N$

(else $N \log K = \Theta(N \log N)$)

PART II

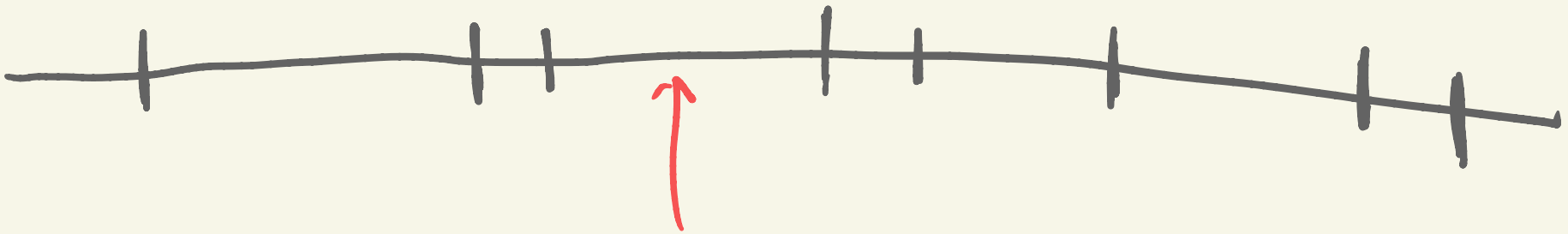
Entropy

Warm up : 1-D point location



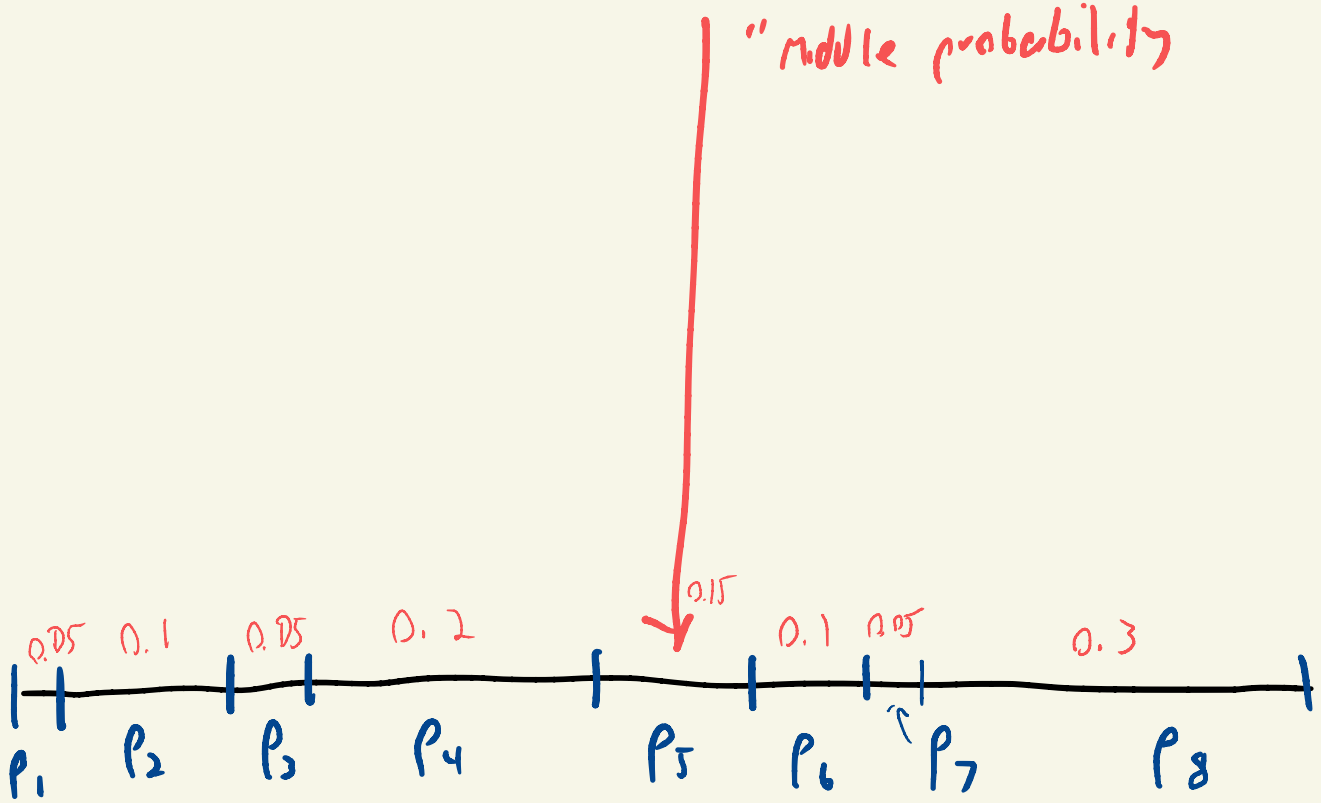
Binary search: $O(\log N)$

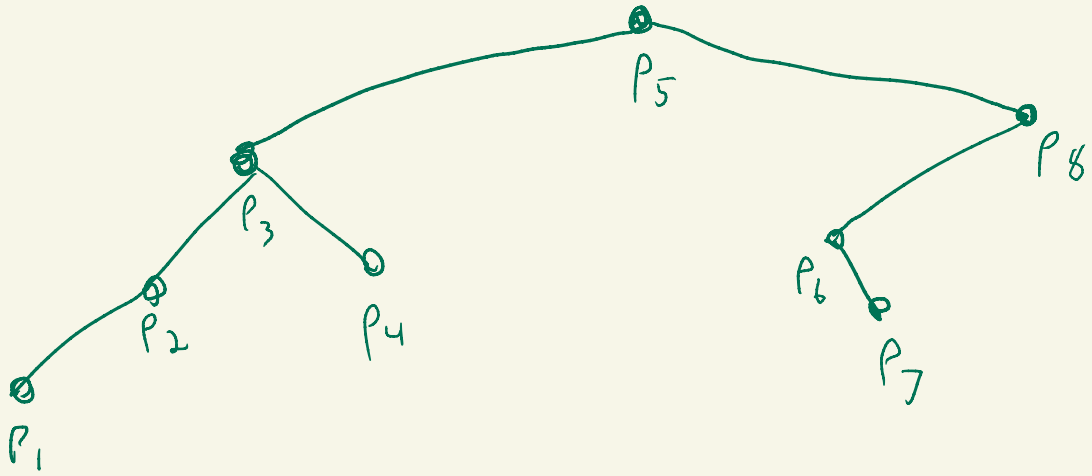
Warm up: 1-D point location

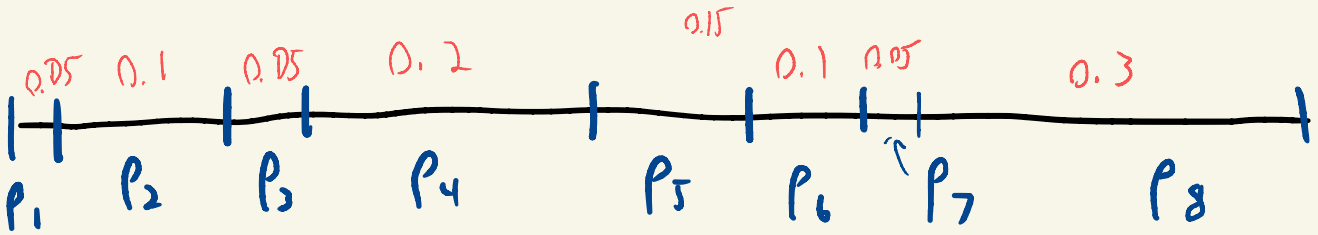


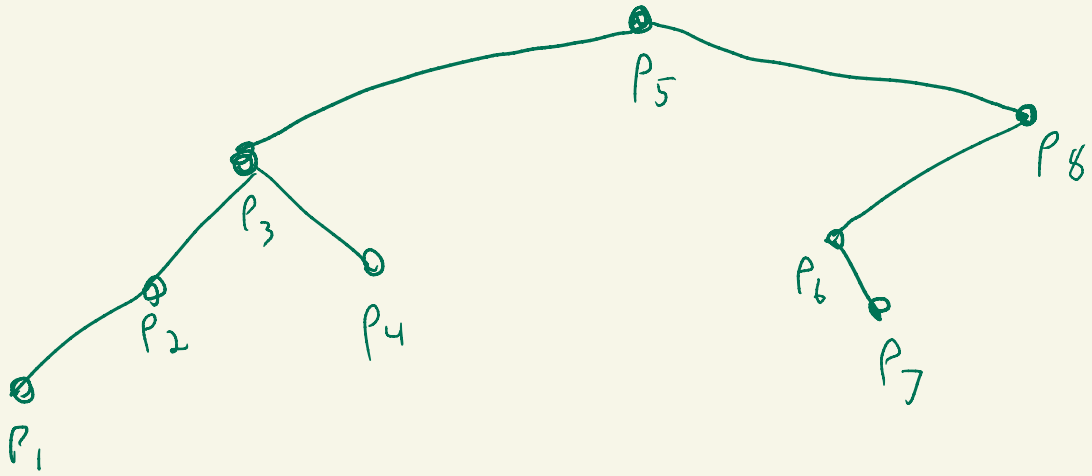
Binary search: $O(\log N)$

But what if queries are drawn independently from a known distribution?


"middle probability
0.15
0.05
0.1
0.05
0.2
0.1
0.05
0.3
p1
p2
p3
p4
p5
p6
p7
p8

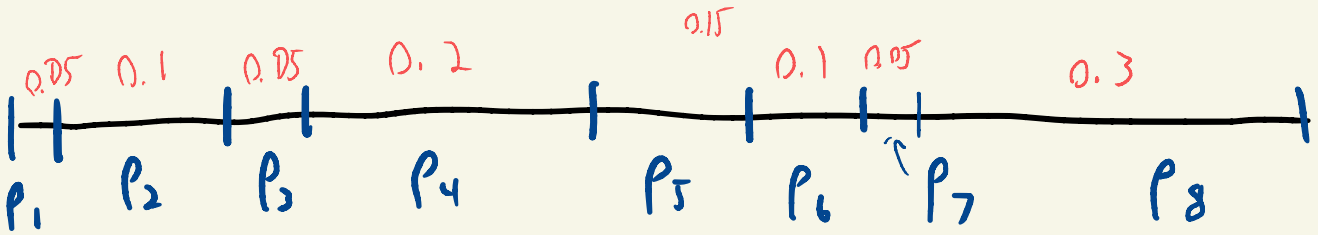

P5
P8
P3
P6
P2
P4
P7
P1

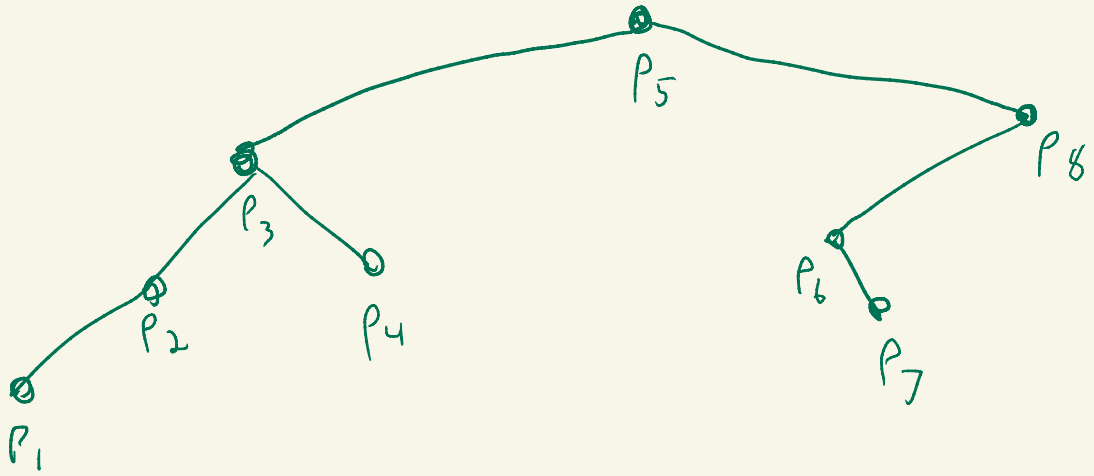

0.15
0.05
0.1
0.05
0.2
0.05
0.1
0.3
P1
P2
P3
P4
P5
P6
P7
P8



$p_i \geq \frac{1}{2^i}$

$\Rightarrow p_i$ at depth $\leq i$



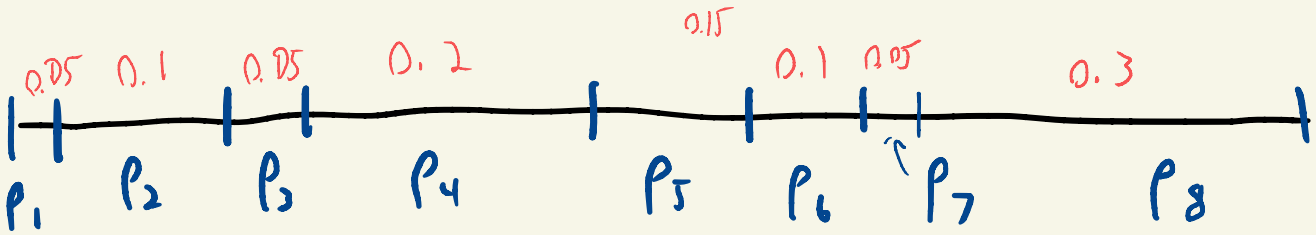


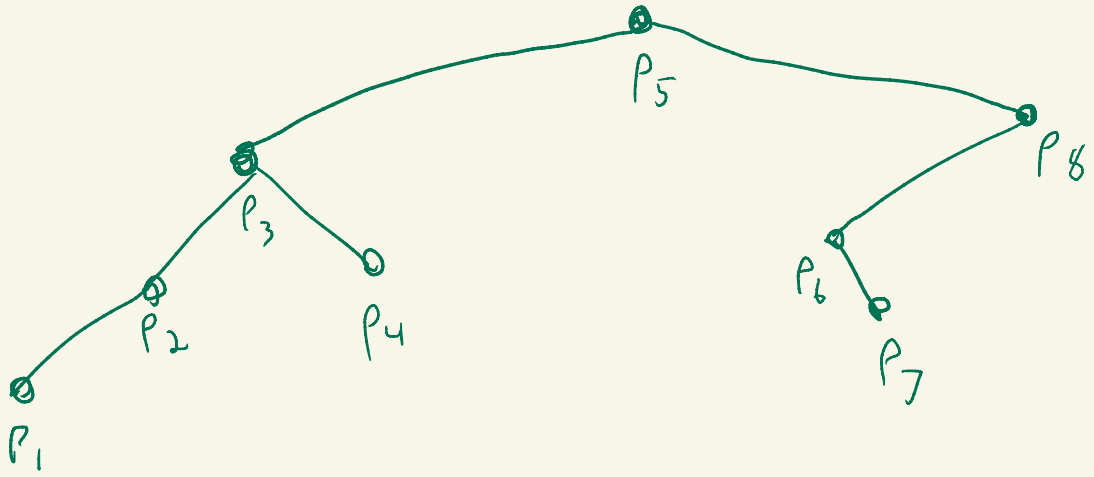
$p_i \geq \frac{1}{2^i}$

$\Rightarrow p_i$ at depth $\leq i$

Expected cost is

$$O\left(\sum_i p_i \log \frac{1}{p_i}\right)$$





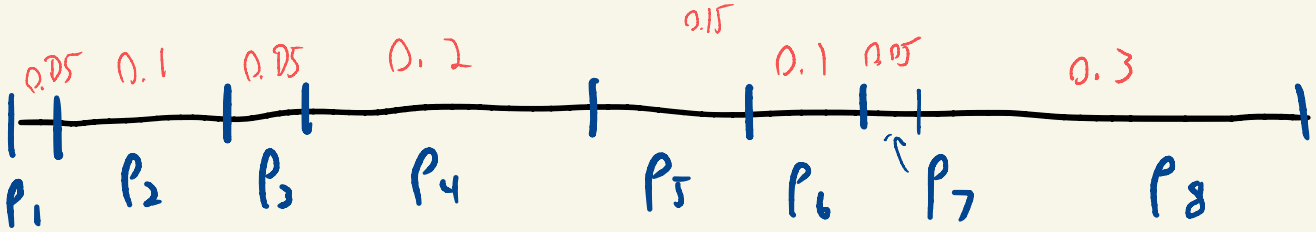
$p_i \geq \frac{1}{2^i}$

$\Rightarrow p_i$ at depth $\leq i$

Expected cost is

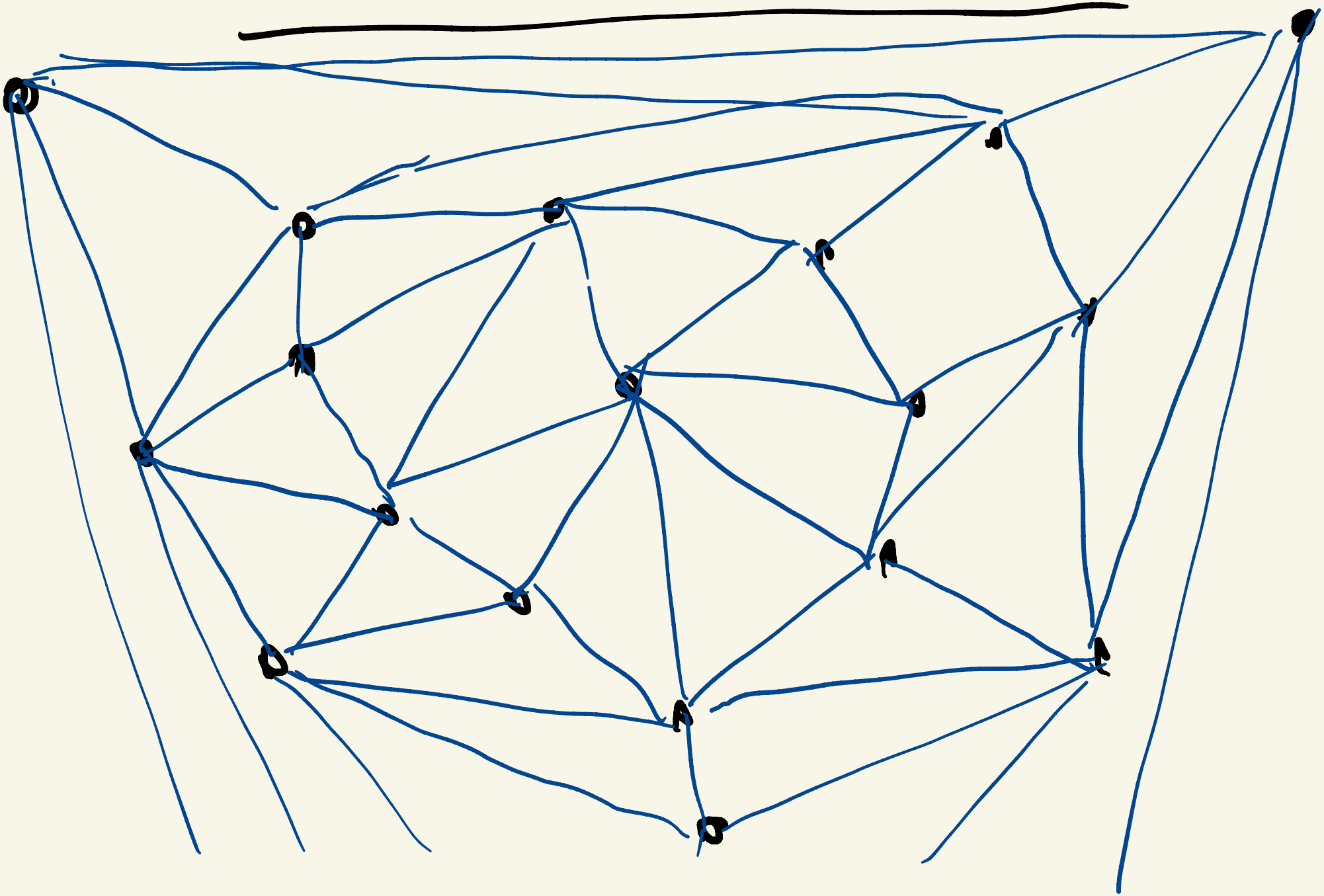
$$O\left(\sum_i p_i \log \frac{1}{p_i}\right)$$

Entropy H

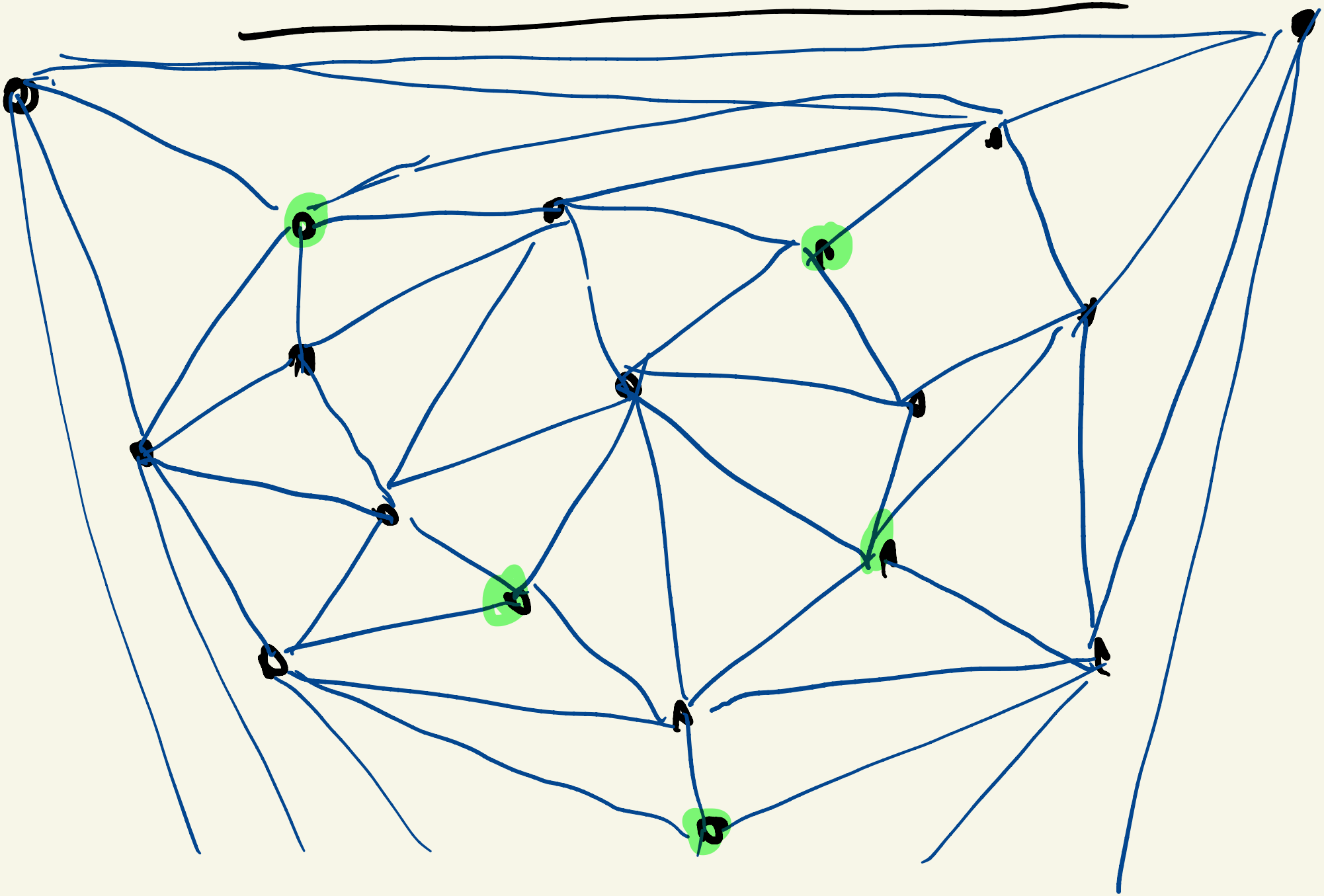


Optimal by Shannon

Planar Point Location



Planar Point Location



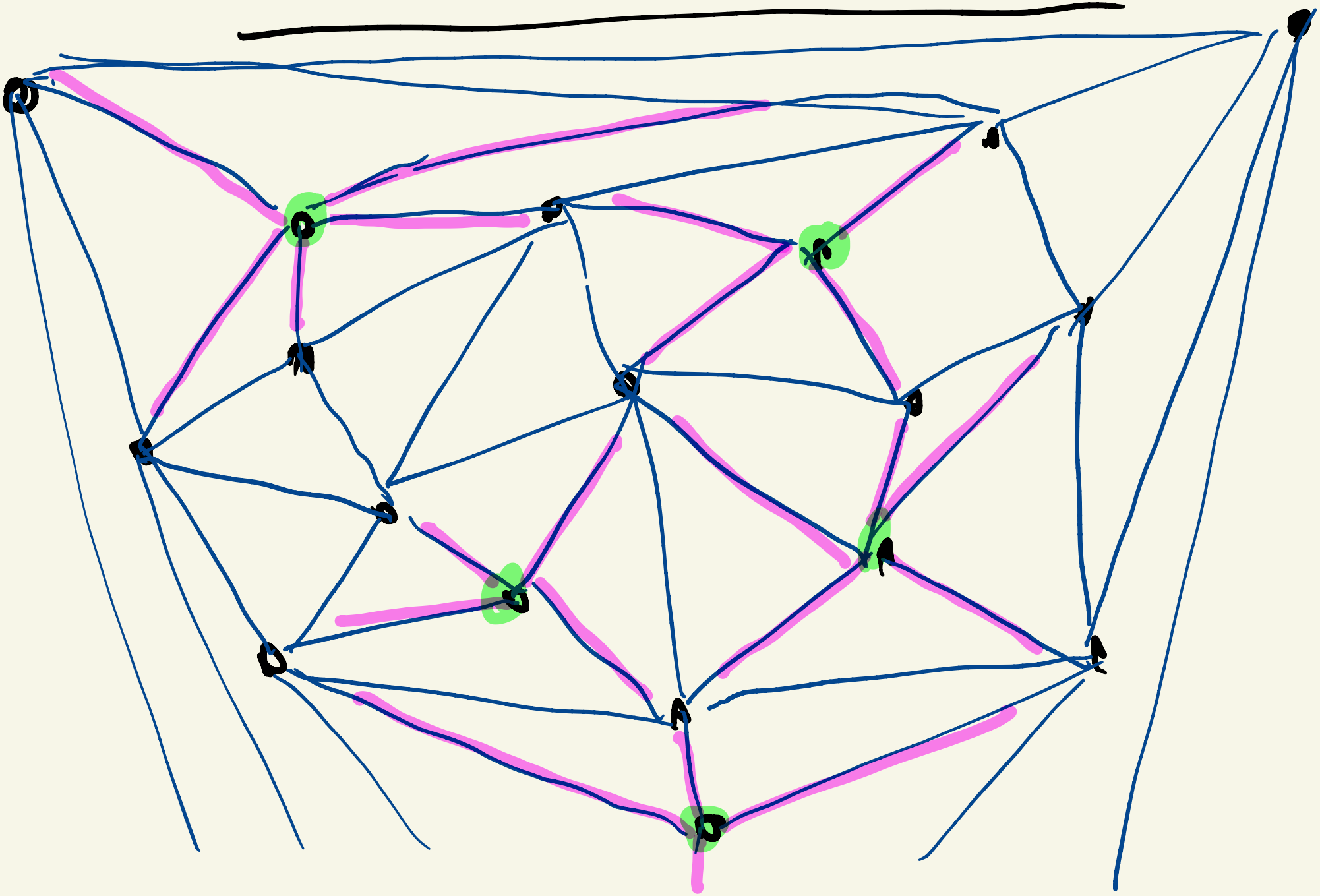
Pick a constant fraction of the verticies that are

- An independent set
- Constant degree

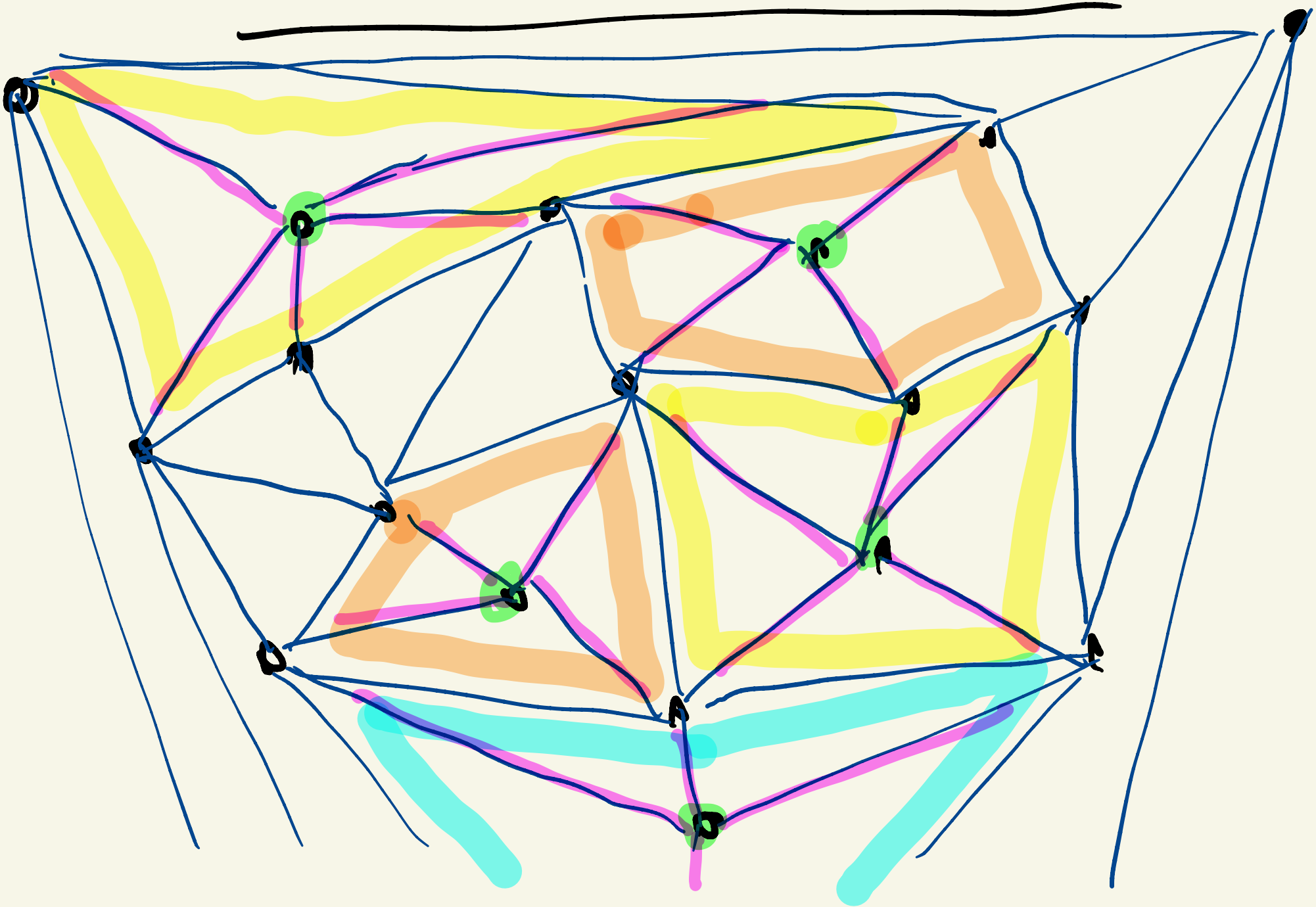
Average degree $< 6 \rightarrow$ 50% have degree < 12

- Easy to find ind set of $V/24$ verticies of degee < 12 in $O(V)$ time

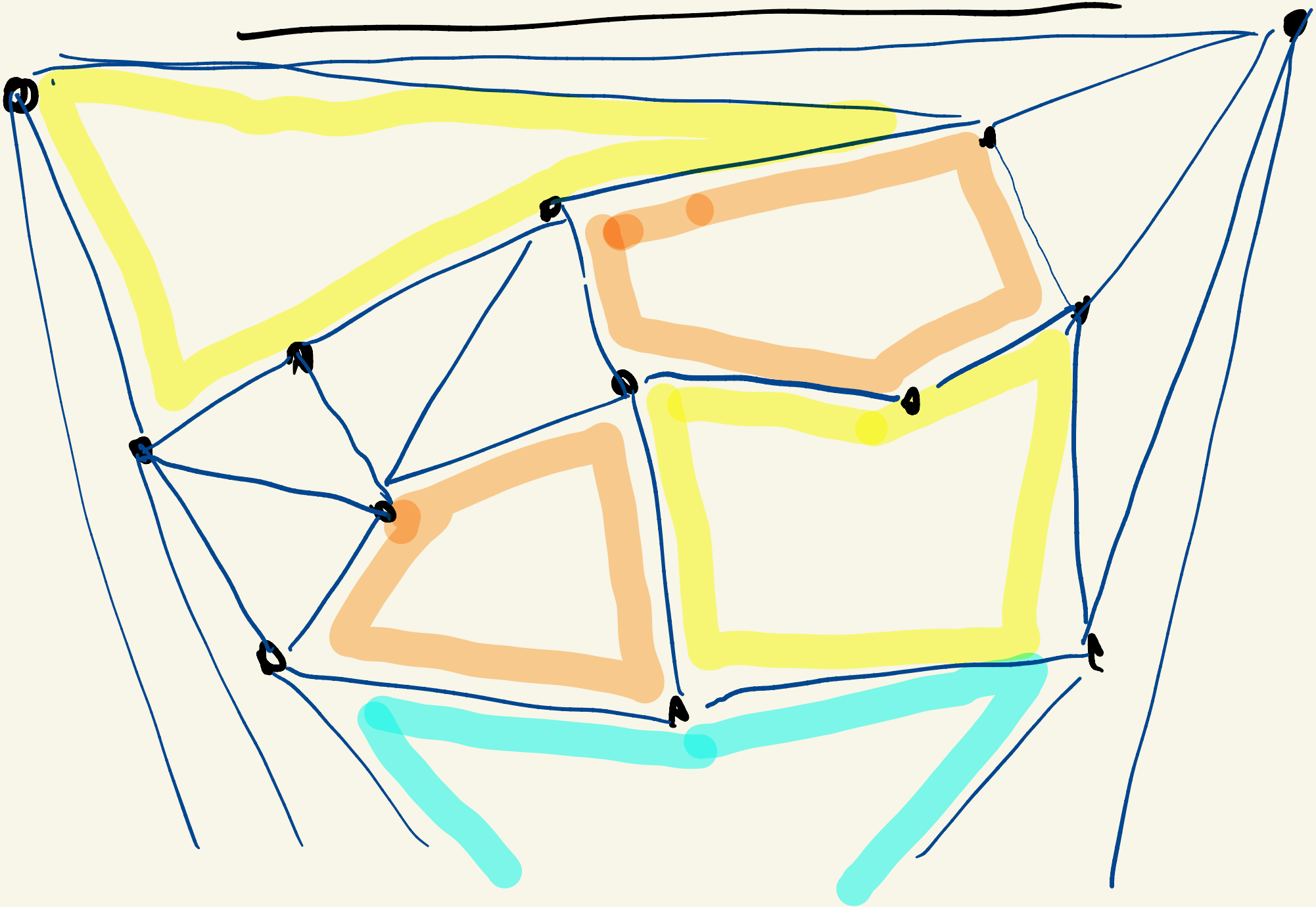
Planar Point Location



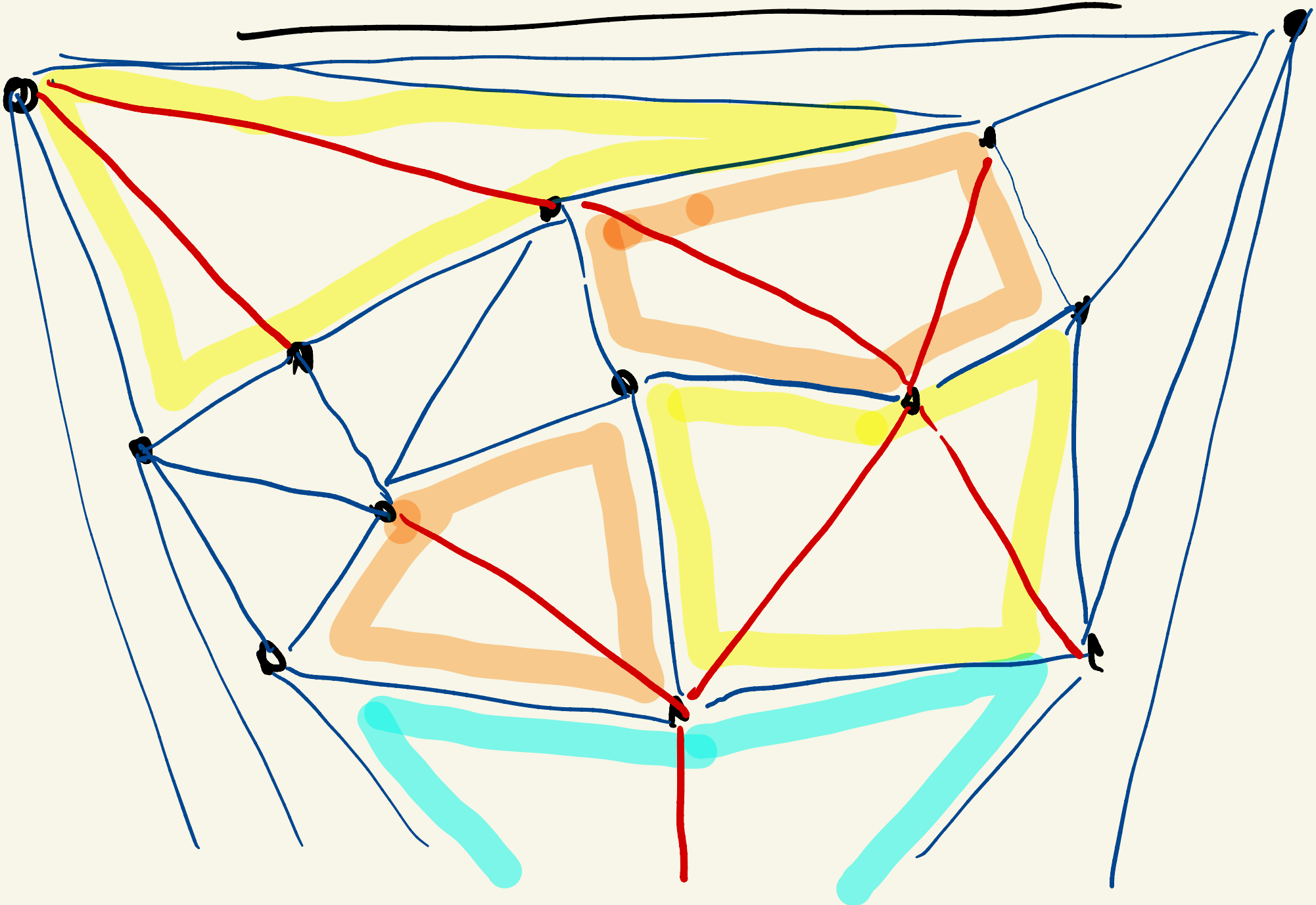
Planar Point Location



Planar Point Location



Planar Point Location

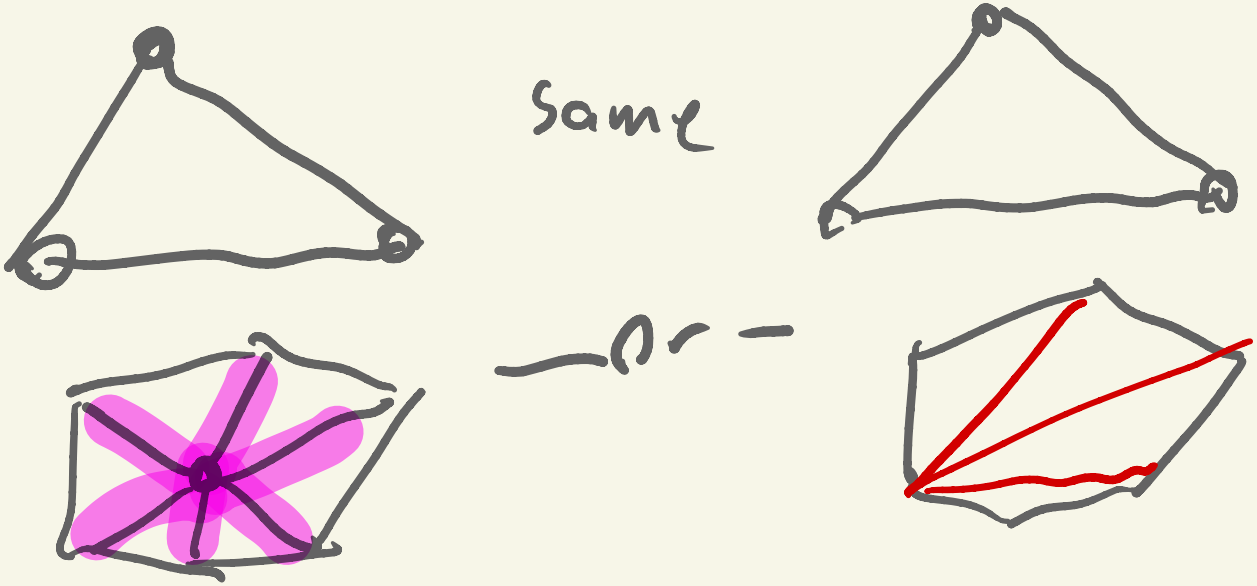


OLD

New

N vertexes

$\frac{23}{24} N$ vertexes



pt location takes time $T\left(\frac{23}{24}N\right) + O(1)$

pt loc takes time $T(N)$

$T(N) = O(\log N)$

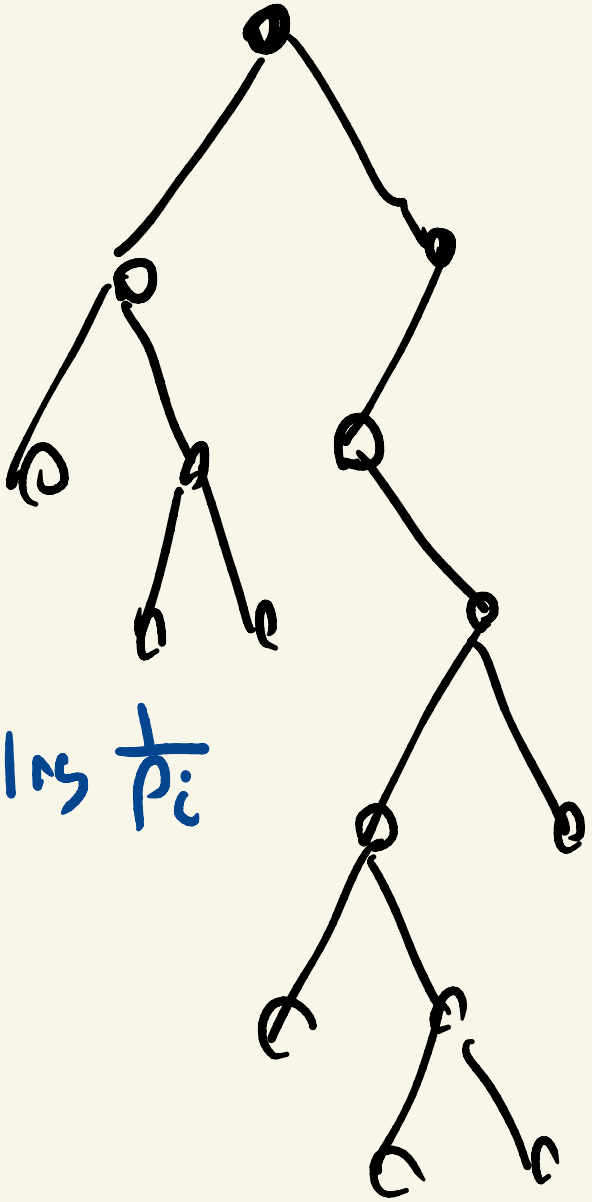
[Kirkpatrick Seidel 1983]

Better than $O(\log N)$?

What if the queries came independently from a known distribution?

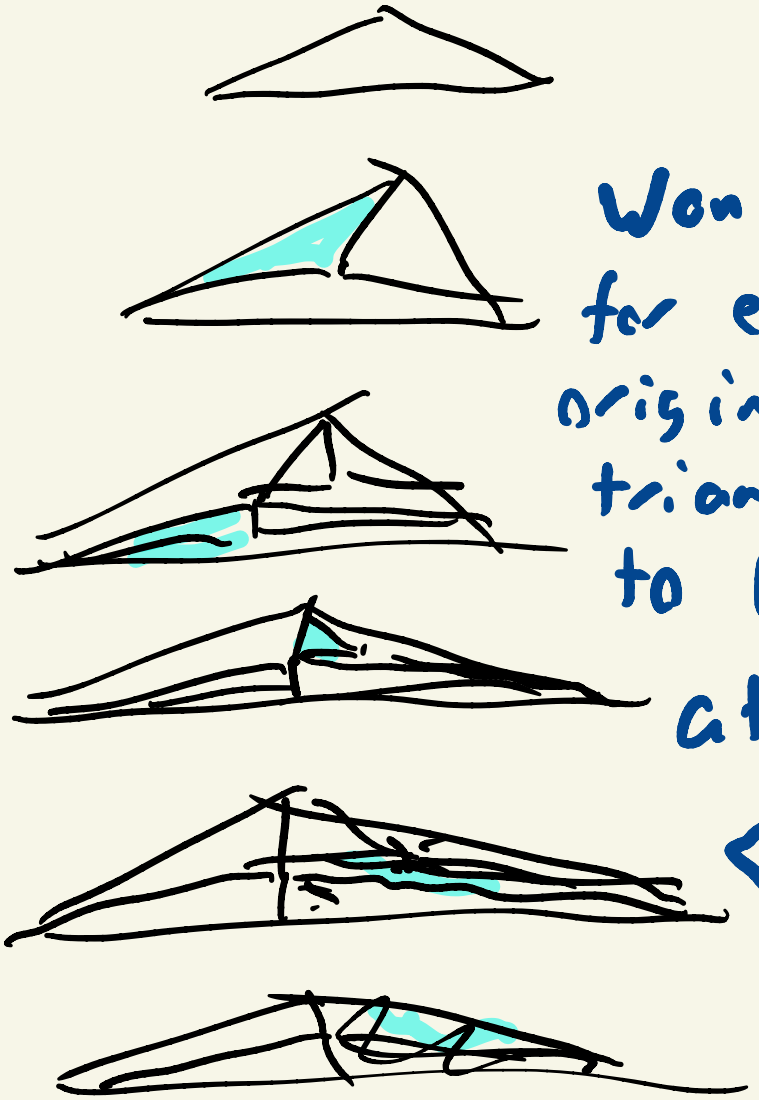
Can we achive $O(\text{entropy})$

1-D Tree



Have depth $\leq \log \frac{1}{p_i}$

Point Location



Want: for each original triangle to be at depth $< \log \frac{1}{p_i}$

A simple solution

- Independent Set
- Constant degree
* Incident to triangles whose probability is $\leq c \cdot$ Average probability.

A simple solution

- Independent Set
- Constant degree
* Incident to triangles whose probability is $\leq c \cdot$ Average probability.

Means that a triangle with prob p_i want get deleted until $O(\frac{1}{p_i}) = V$
and thus depth is $O(\log \frac{1}{p_i})$

Result [I 01 / Sada] [I 04]

Given a known distribution, PPL on ind queries takes $O(H) = O\left(\sum p_i \log \frac{1}{p_i}\right)$

Result [I 01 Sada] [I 04]

Given a known distribution, PPL on ind queries takes $O(H) = O\left(\sum p_i \log \frac{1}{p_i}\right)$

But wait, what if I don't know the distribution?

Result [I 01 Sada] [I 04]

Given a known distribution, PPL on ind queries takes $O(H) = O(\sum p_i \log \frac{1}{p_i})$

But wait, what if I don't know the distribution?

Is OK! Just periodically rebuild with the observed frequencies.

[I Mulzer 12] [I 11 SOCG]

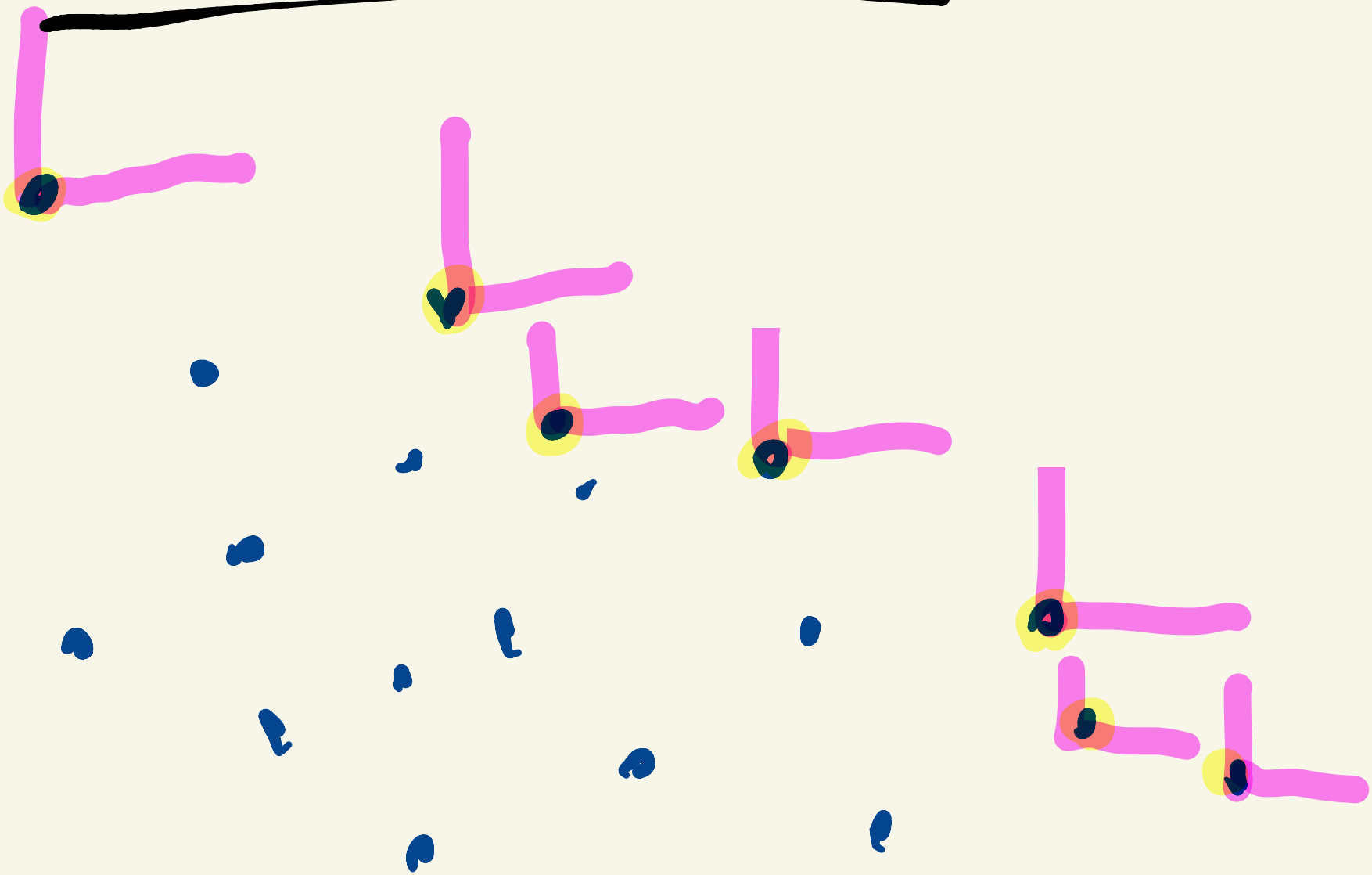
Part 3

Instance-Based Optimality

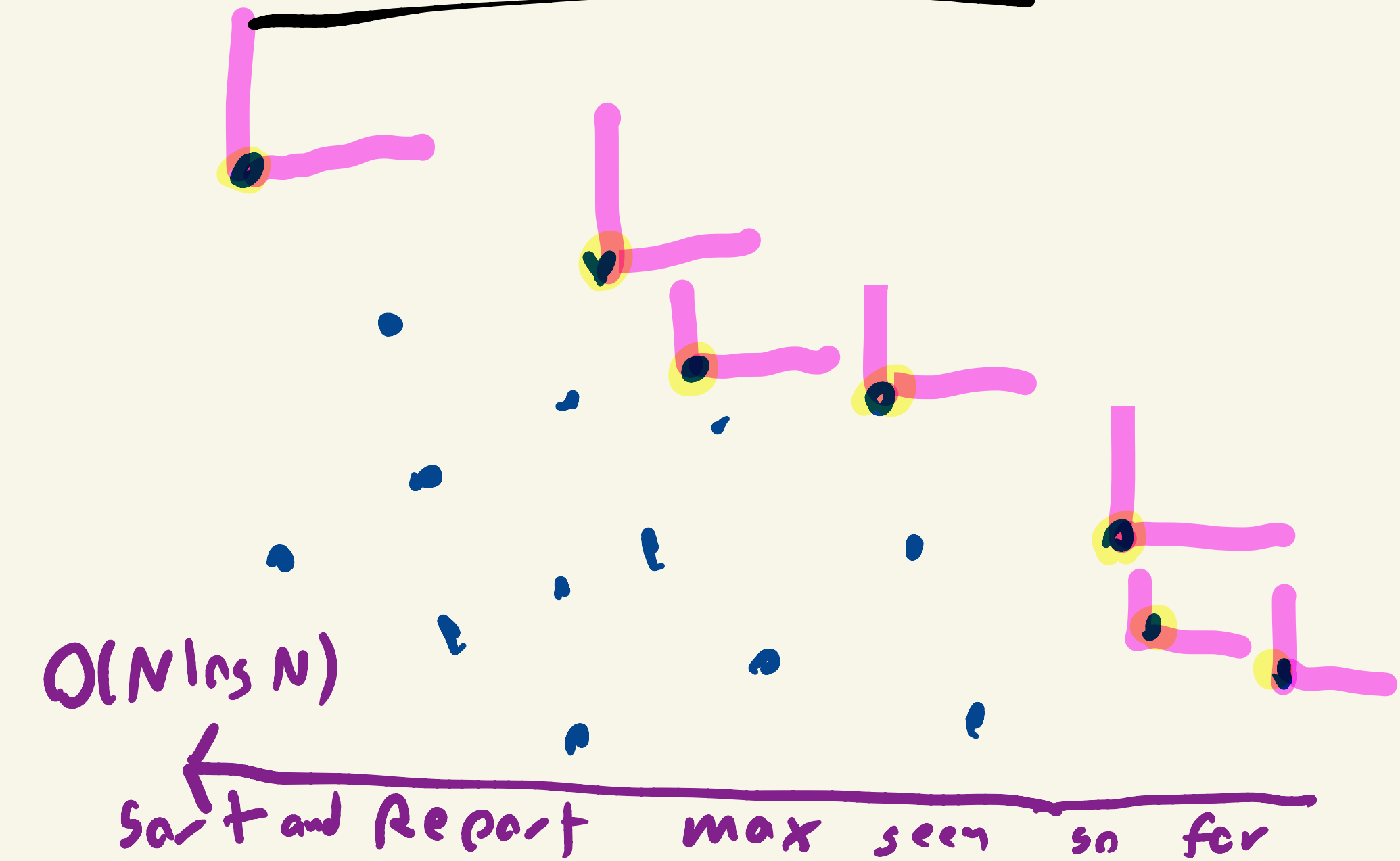
[Afshani Barbay Chan 2015]

Dominance

Dominance



Dominance



O(N log K)

- Need same ingredient as before

1 8 5 12 17 6 13 8 5

First to the left ≥ 12

O(N log K)

- Need same ingredient as before

1 8 5 12 17 6 13 8 5

First to the left ≥ 12

Construction	Query
O(n log n)	O(log N)

Augmented BST

$O(N \log K)$

- Need same ingredient as before

1 8 5 12 17 6 13 8 5

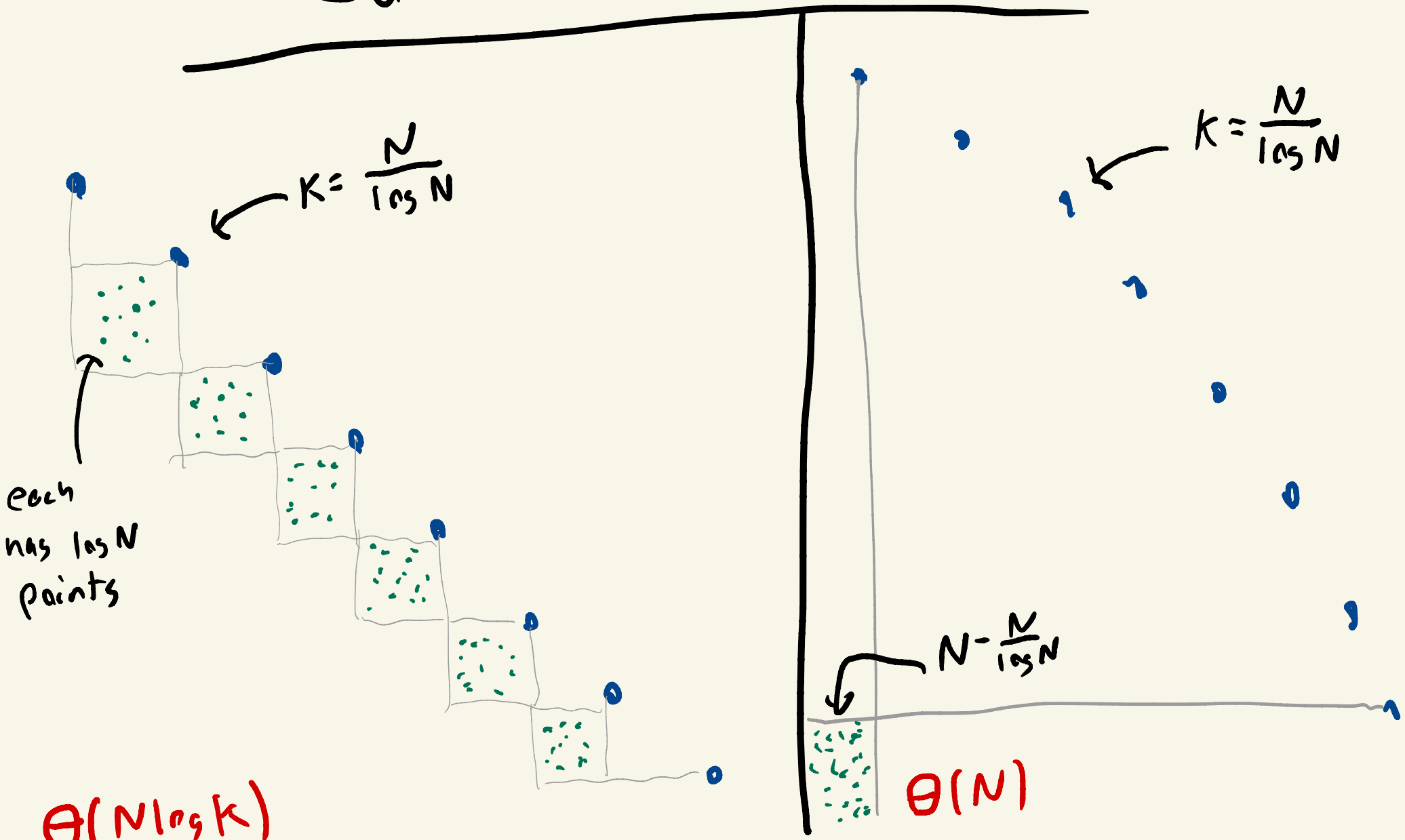
First to the left ≥ 12

Construction	Query
$O(n \log n)$	$O(\log N)$
$O(N \log K)$	$O(\frac{N}{K} \log K)$

Augmented BST

Groups

Can we do better?



Beyond K: Instance-Based Optimality

- For any set of points $P = p_1, p_2 \ldots p_n$ there is an algorithm that runs in time $O(N)$ to compute the maxima

Beyond K: Instance-Based Optimality

- For any set of points $P = p_1, p_2 \ldots p_n$ there is an algorithm that runs in time $O(N)$ to compute the maxima
- This is useless, as the fast algorithm in essense must know the points <u>and</u> their order in advance

Beyond K: Instance-Based Optimality

- For any set of points $P = p_1, p_2 \ldots p_n$ there is an algorithm that runs in time $O(N)$ to compute the maxima

- This is useless, as the fast algorithm in essense must know the points <u>and</u> their order in advance

- Instance-Based Runtime with respect to P
  Worst-case runtime wrt all permutations of P.

  same point set, but unknown ordering

Beyond K: Instance-Based Optimality

- For any set of points $P = p_1, p_2 \ldots p_n$ there is an algorithm that runs in time $O(N)$ to compute the maxima

- This is useless, as the fast algorithm in essense must know the points **and** their order in advance

- Instance-Based Runtime with respect to P
  Worst-case runtime wrt all permutations of P.

  same point set, but unknown ordering

- Classifies "easy" vs "hard" instance

Instance-Based Optimality

- There is an instance-based optimal algorithm for Dominance

:

Instance-Based Optimality

- There is an instance-based optimal algorithm for Dominance

$$\exists A_{OPT,c} \forall p \quad Time(A_{OPT}, p) \leq c \cdot \min_{A} \max_{p' \text{ is a perm of } p} Time(A, p')$$

Instance-Based Optimality

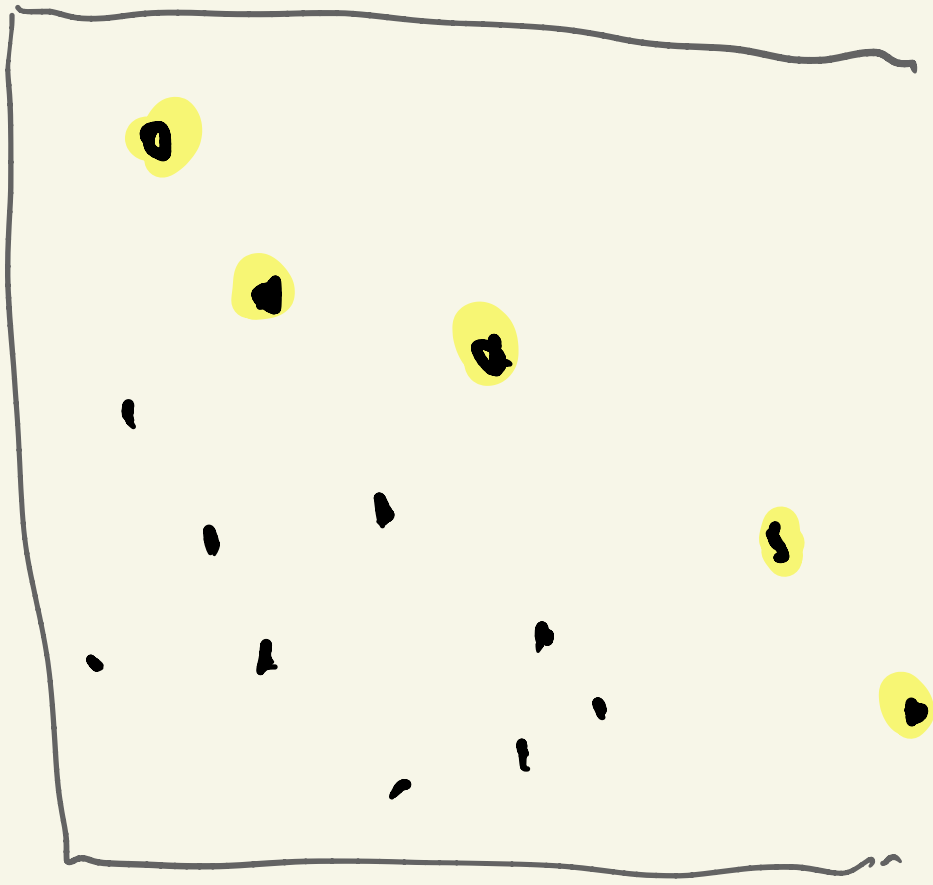
- There is an instance-based optimal algorithm for Dominance

$$\exists A_{OPT,C} \forall P \quad Time(A_{OPT}, P) \leq C \cdot \min_{A} \max_{P' \text{ is a perm of } P} Time(A, P')$$

A is the best algorithm that knows the points of P but not their order

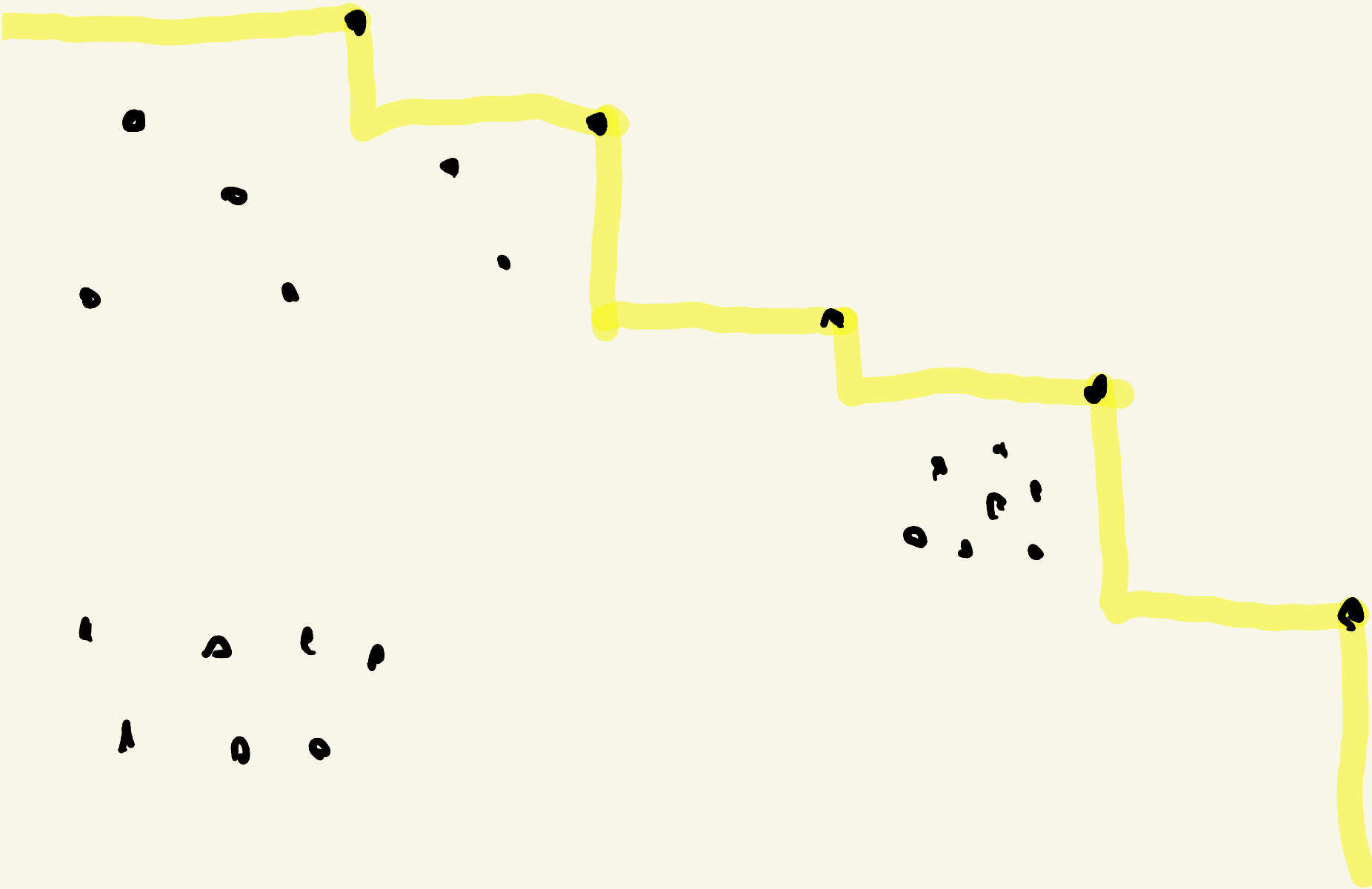
Back to Dominance

- Runtime
- Algorithm
- Prove the algorithm has the runtime
- Prove that for that set of points, NO alg can do better for all permutations of the points



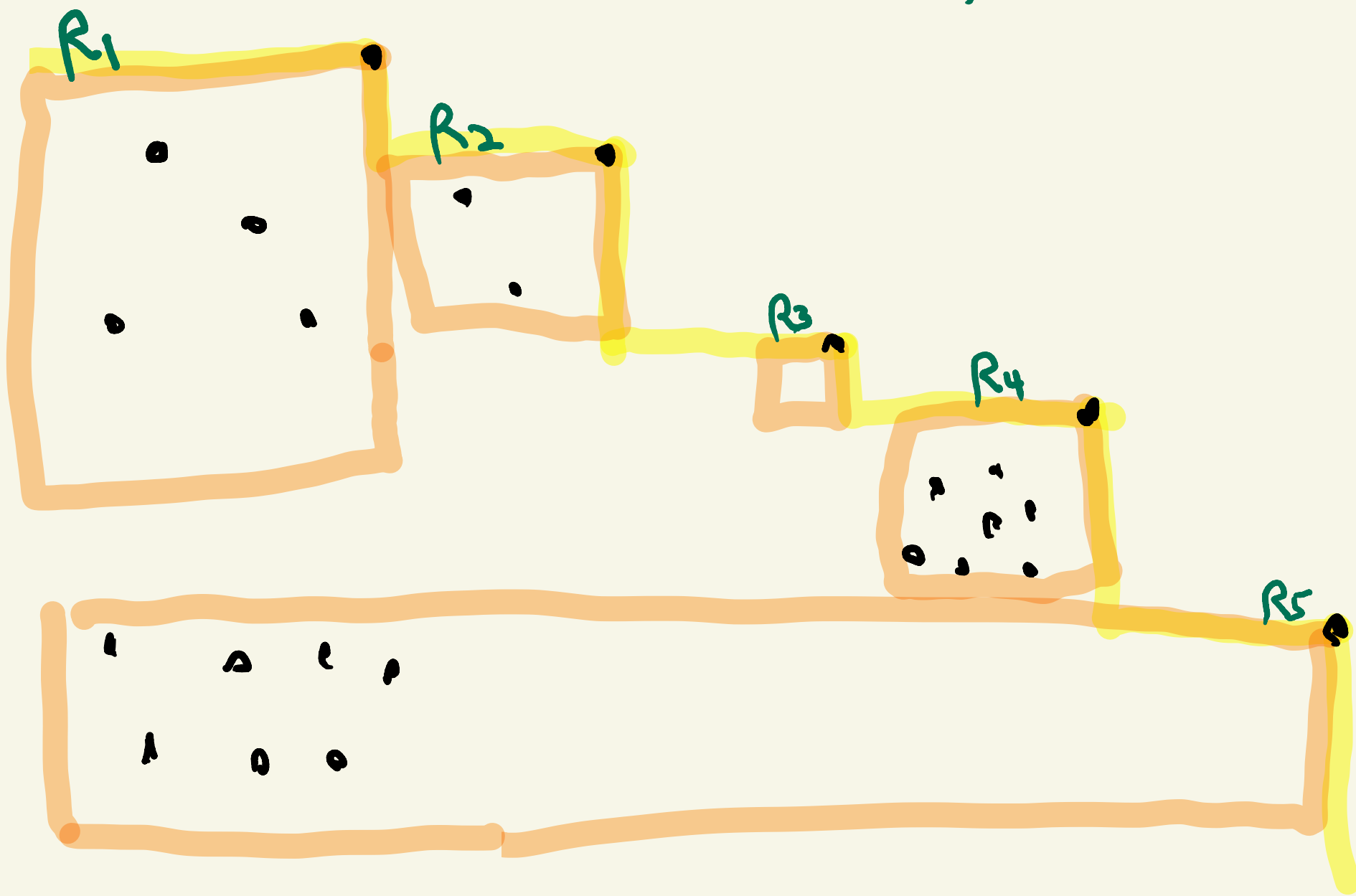
The Runtime Bound

The Runtime Bound



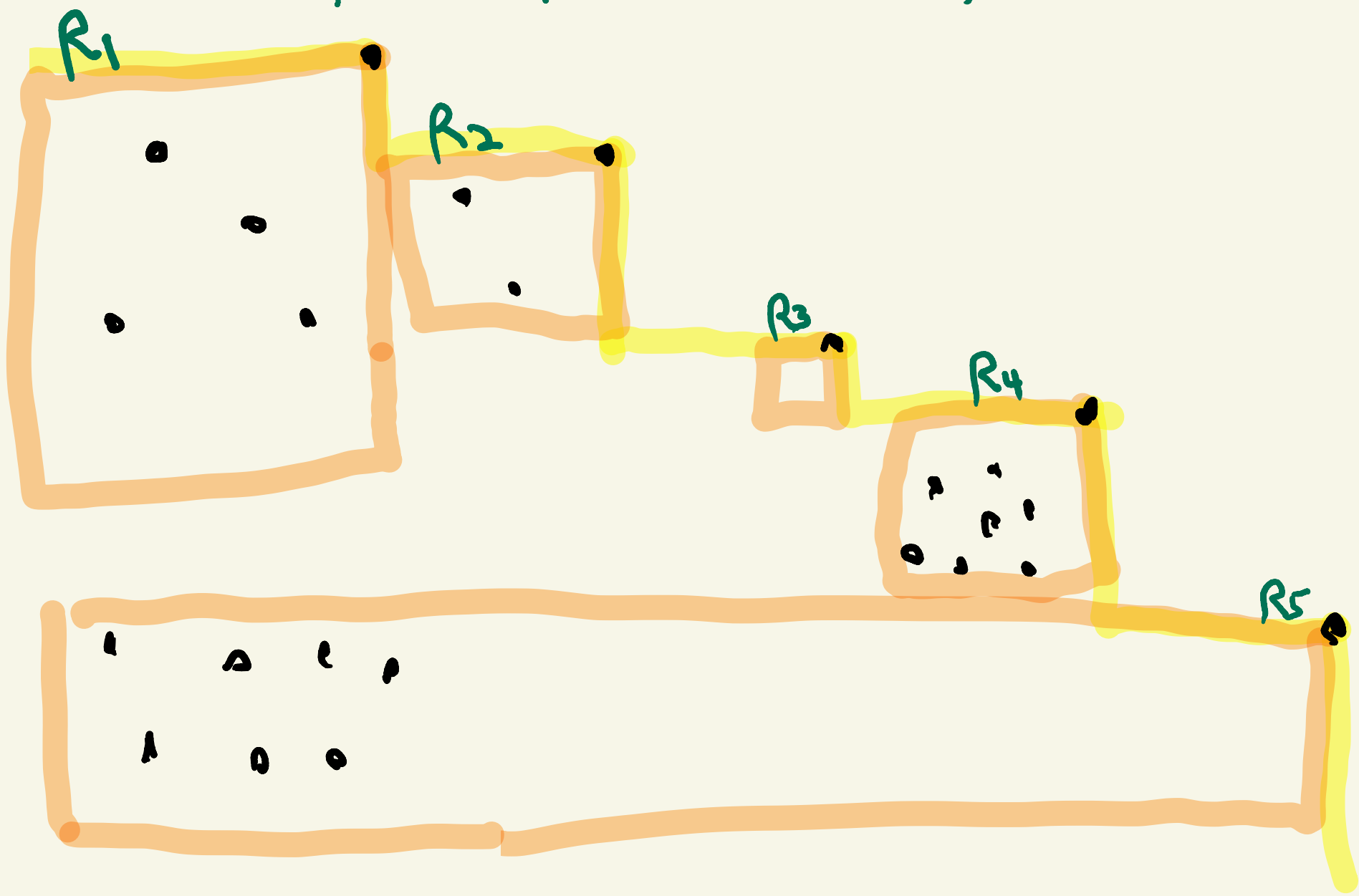
The Runtime Bound

~ Rectangles cove



The Runtime Bound

$\pi = \{R_1, R_2, \ldots\}$ ~ Rectangles cover points

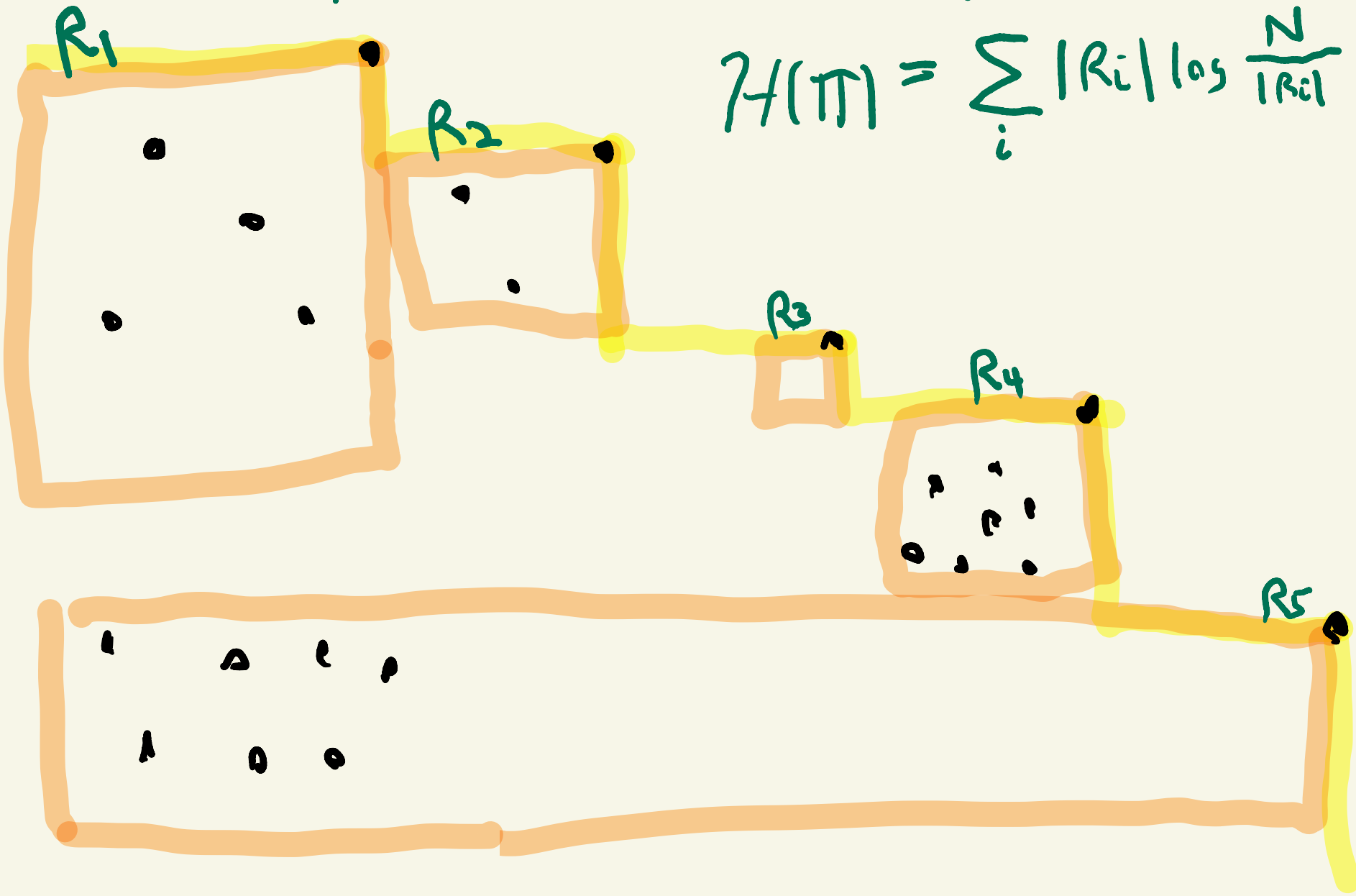


The Runtime Bound

$\pi = \{R_1, R_2, \ldots\}$

- Rectangles cover points

$$\mathcal{H}(\pi) = \sum_i |R_i| \log \frac{N}{|R_i|}$$

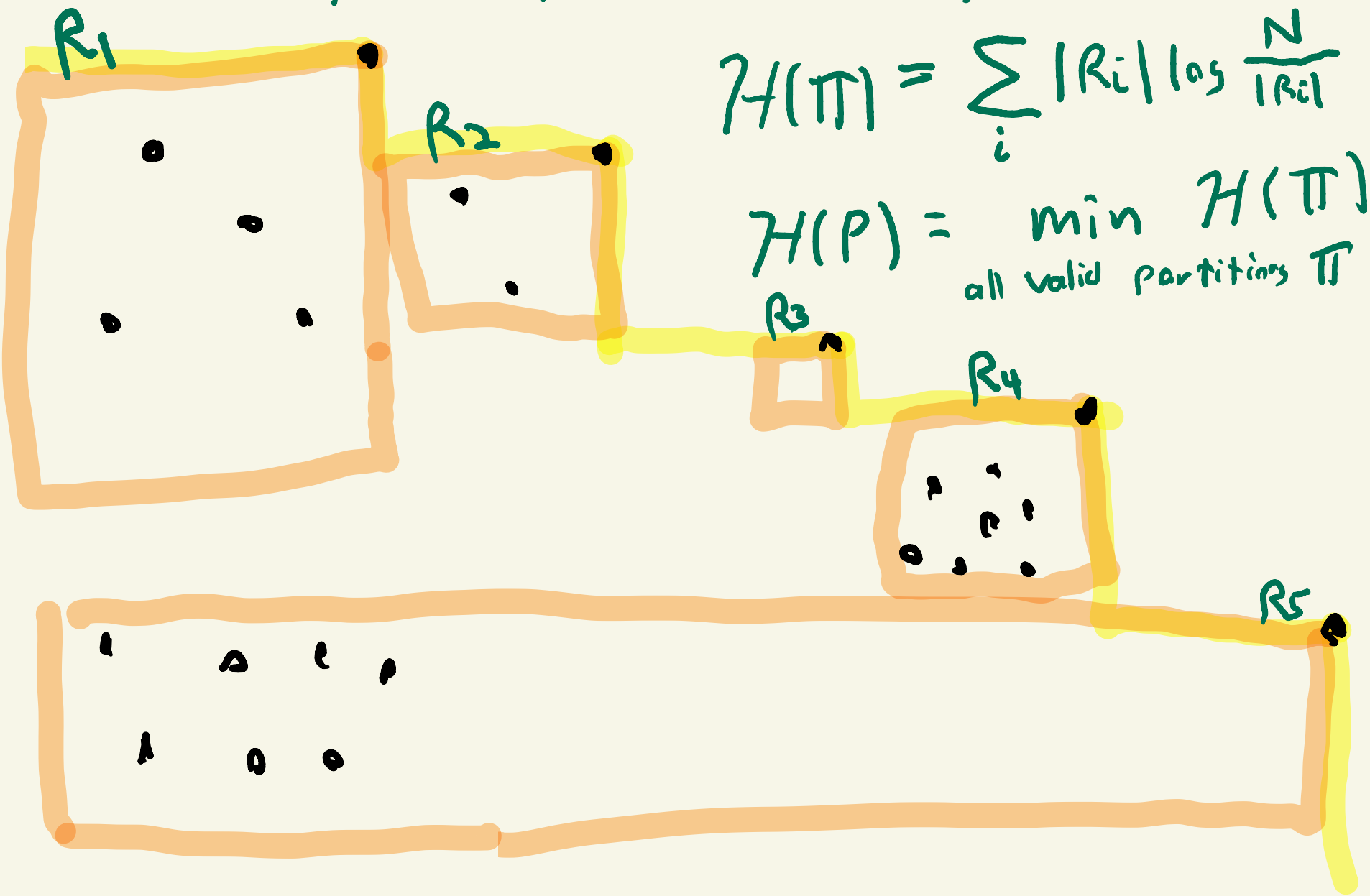


The Runtime Bound

$\Pi = \{R_1, R_2, \ldots\}$ - Rectangles cover points

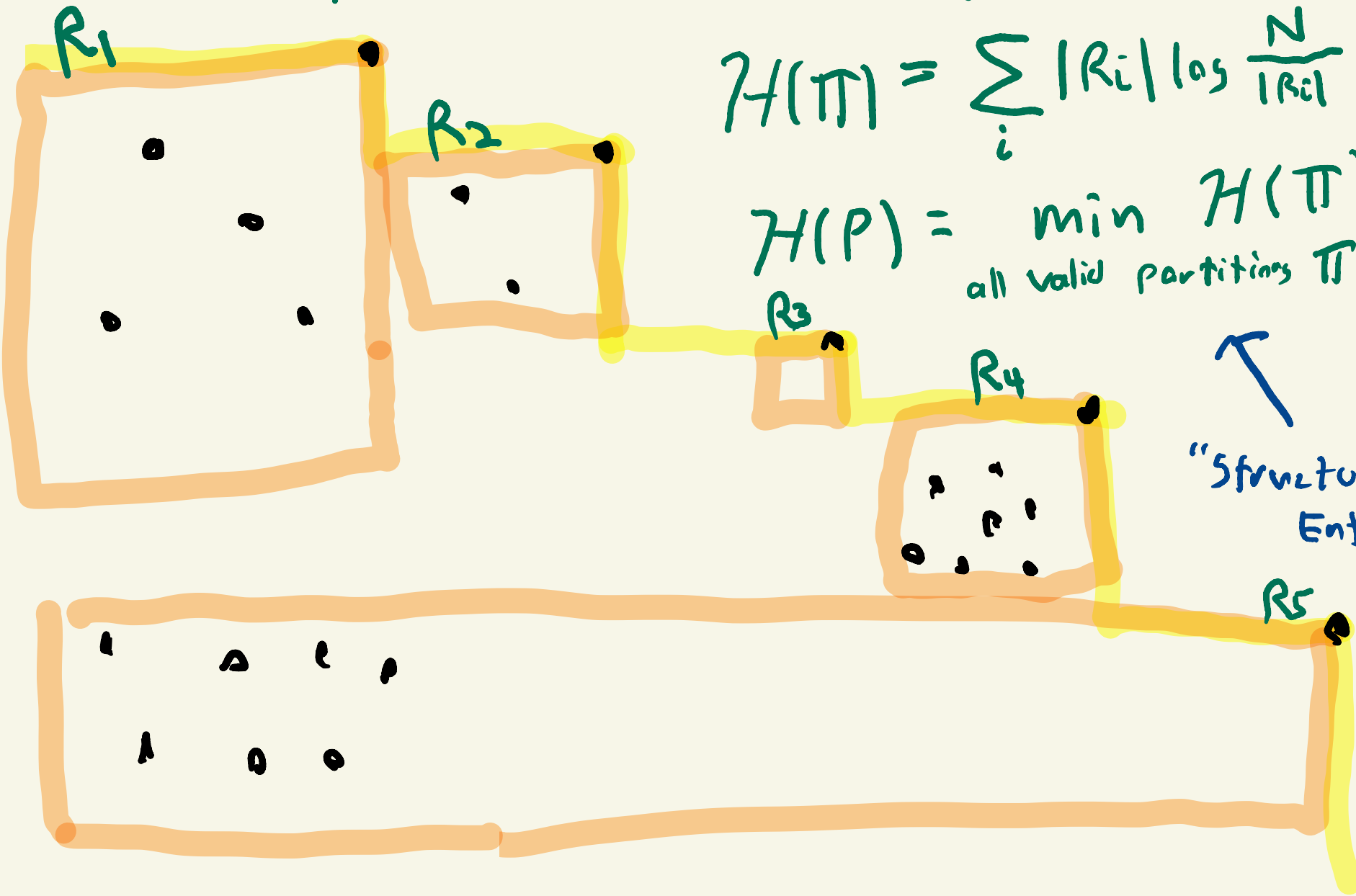
$$\mathcal{H}(\Pi) = \sum_i |R_i| \log \frac{N}{|R_i|}$$

$$\mathcal{H}(P) = \min_{\text{all valid partitions } \Pi \text{ of } P} \mathcal{H}(\Pi)$$



The Runtime Bound

$\pi = \{R_1, R_2, \ldots\}$ — Rectangles cover points

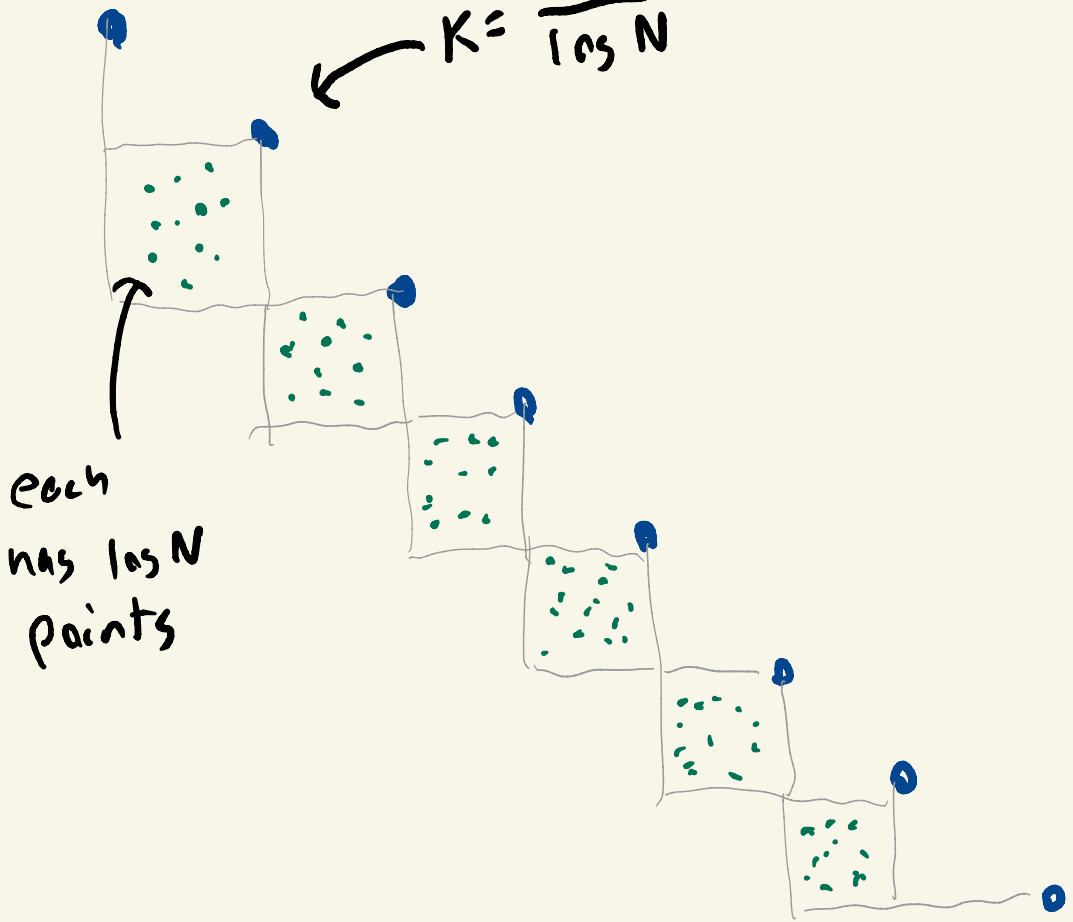


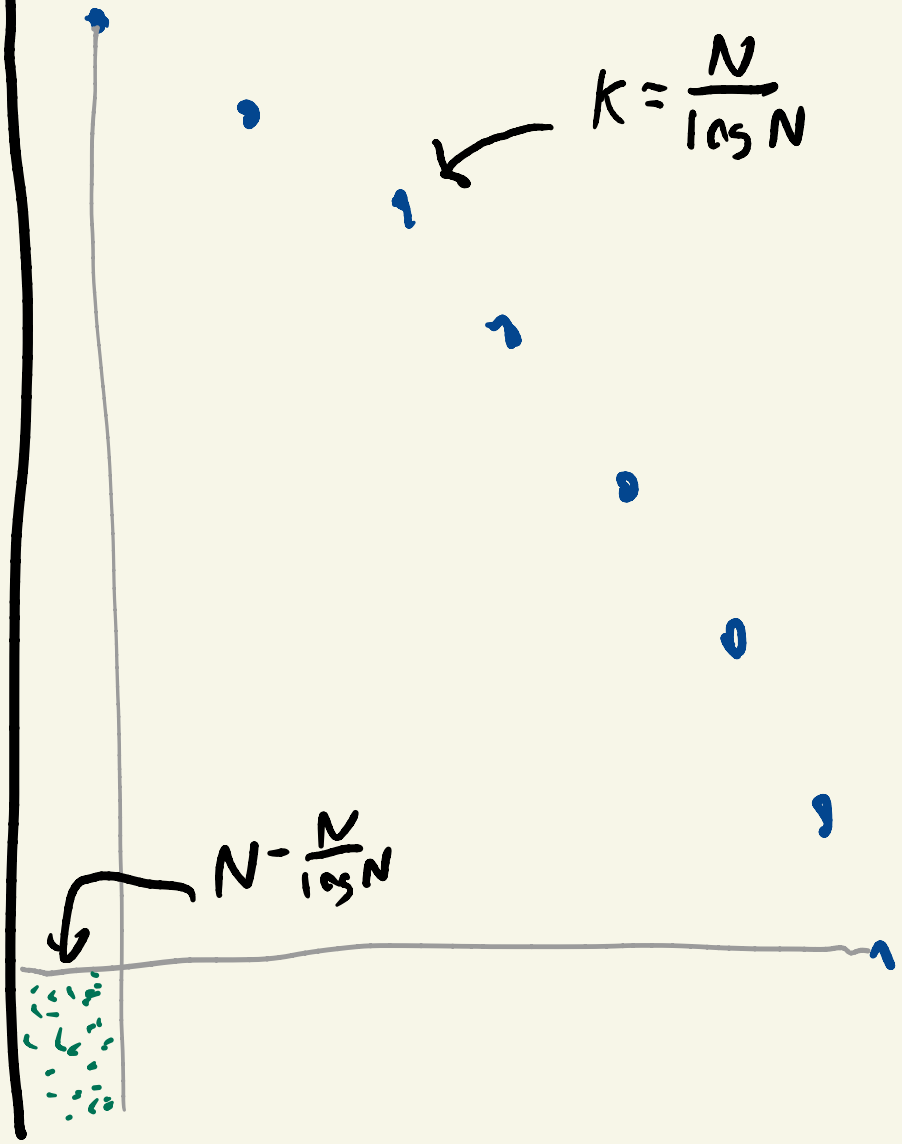
$$\mathcal{H}(\pi) = \sum_i |R_i| \log \frac{N}{|R_i|}$$

$$\mathcal{H}(P) = \min_{\text{all valid partitions } \pi \text{ of } P} \mathcal{H}(\pi)$$

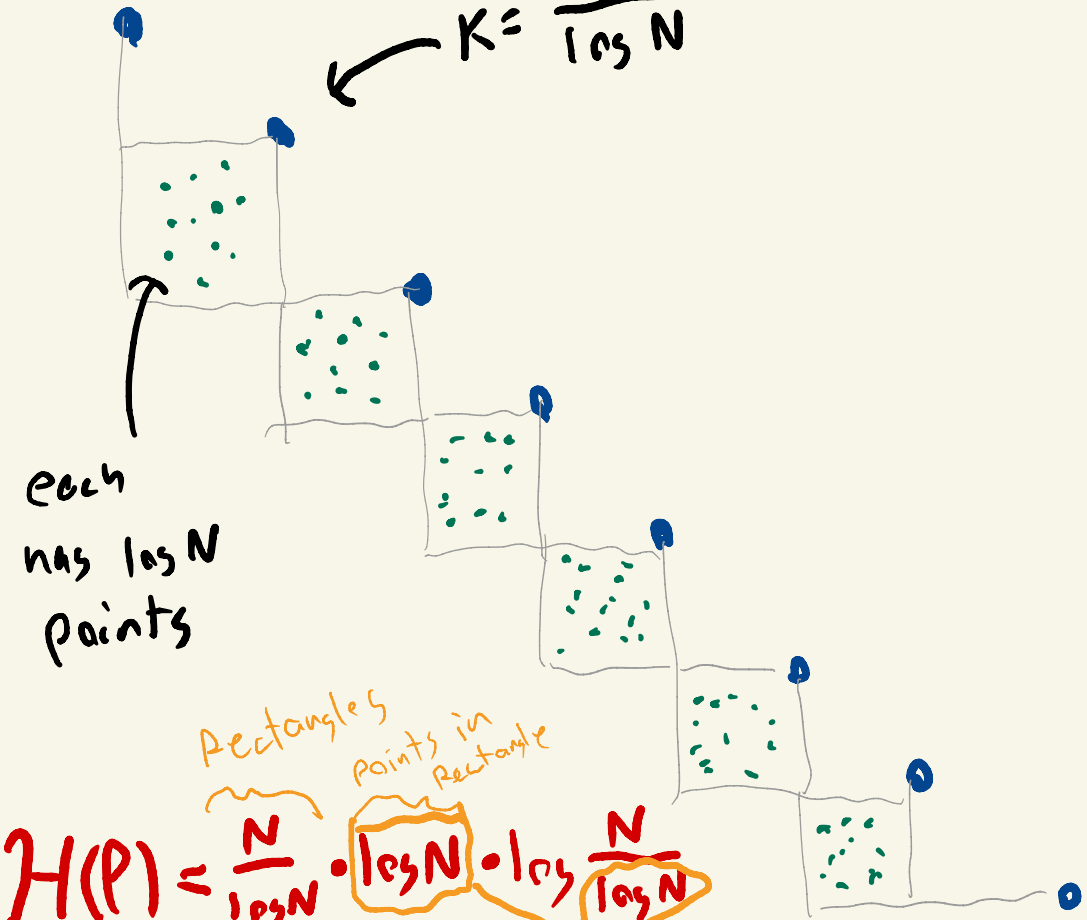
"Structural Entropy"

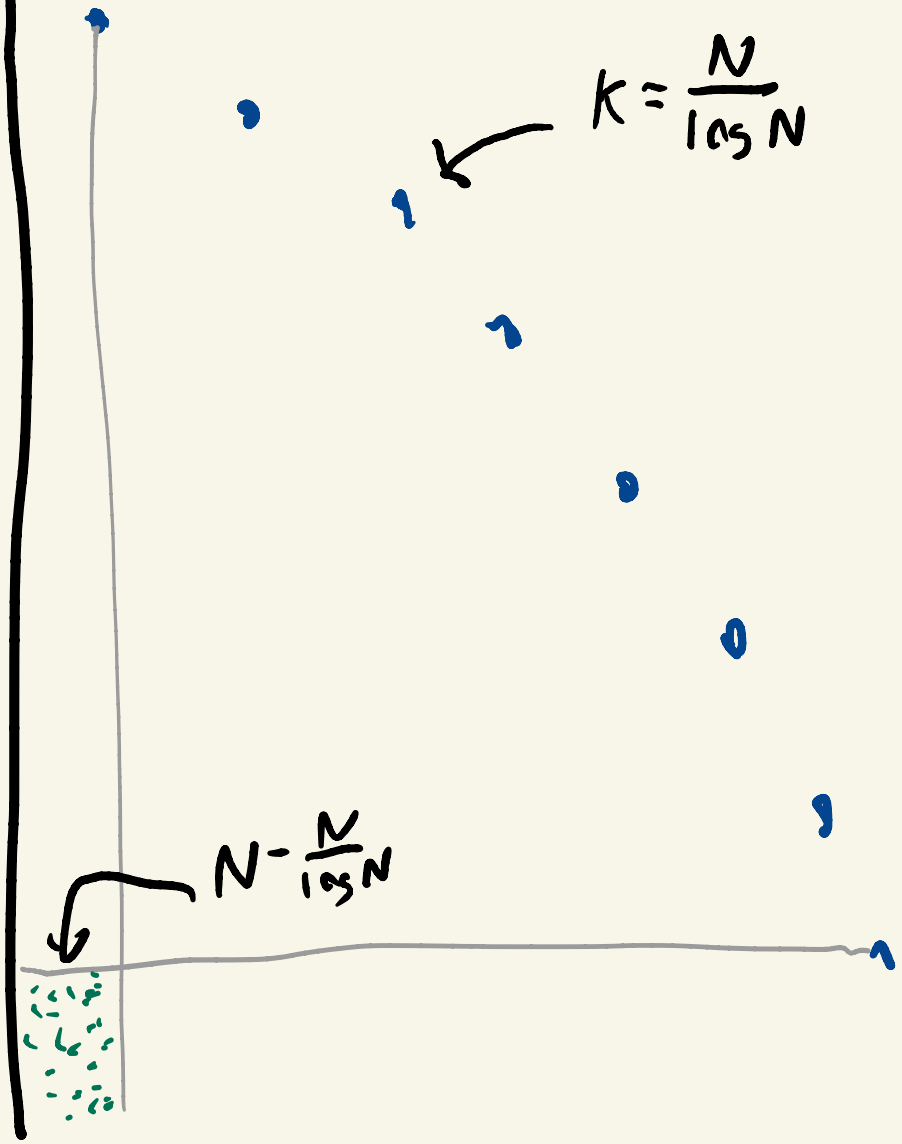
Structural Entropy $\sum_i |R_i| \log \frac{N}{|R_i|}$





Structural Entropy $\sum_i |R_i| \log \frac{N}{|R_i|}$





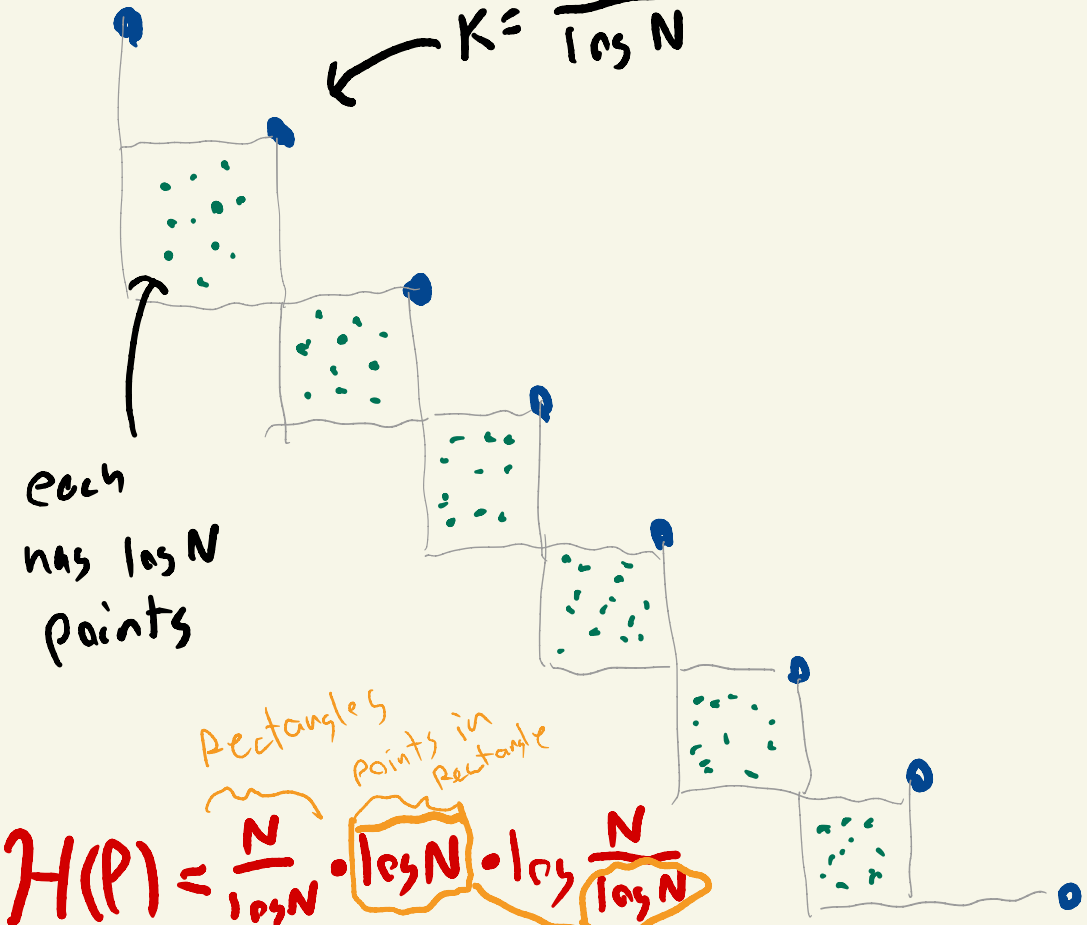
Rectangles · points in Rectangle

$$H(P) = \frac{N}{\log N} \cdot \log N \cdot \log \frac{N}{\log N}$$

$$= \Theta(N \log N)$$

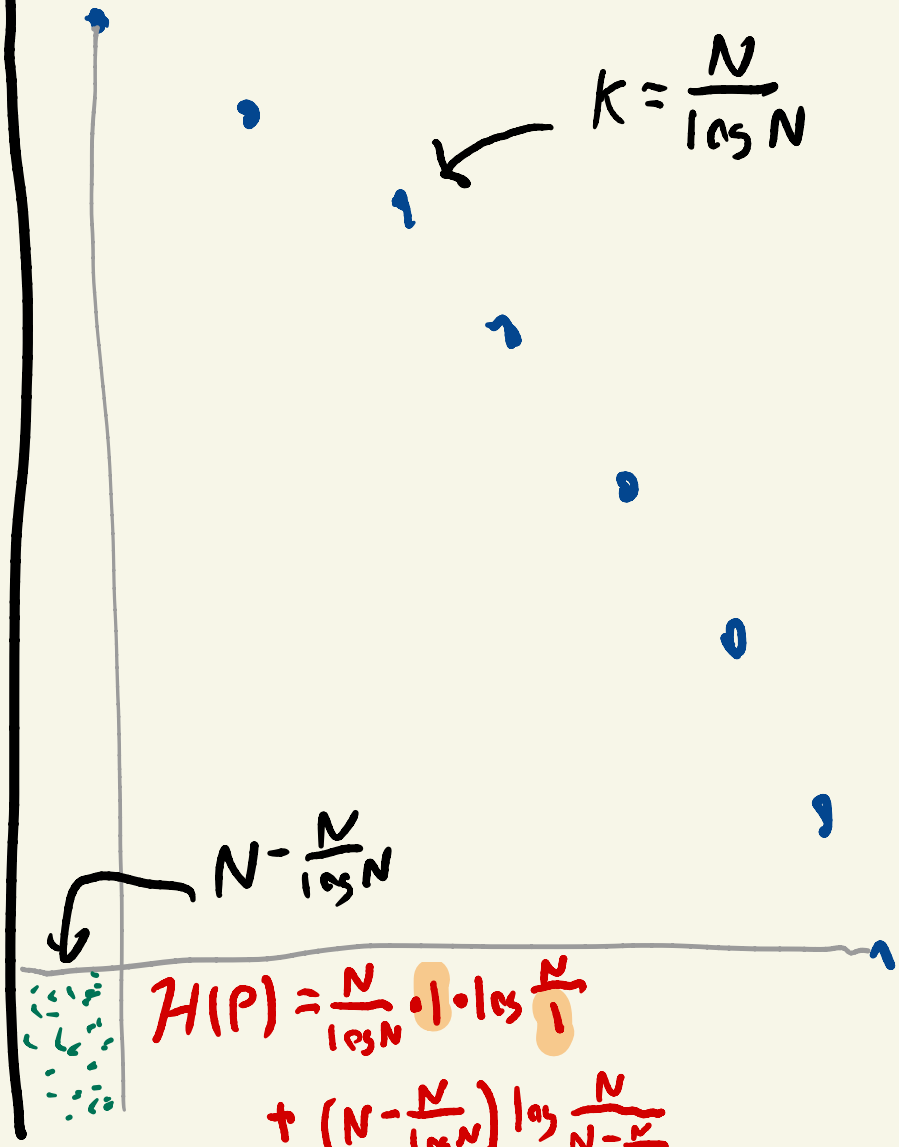
Structural Entropy

$$\sum_i |R_i| \log \frac{N}{|R_i|}$$



$$H(P) = \frac{N}{\log N} \cdot \log N \cdot \log \frac{N}{\log N}$$

$$= \Theta(N \log N)$$

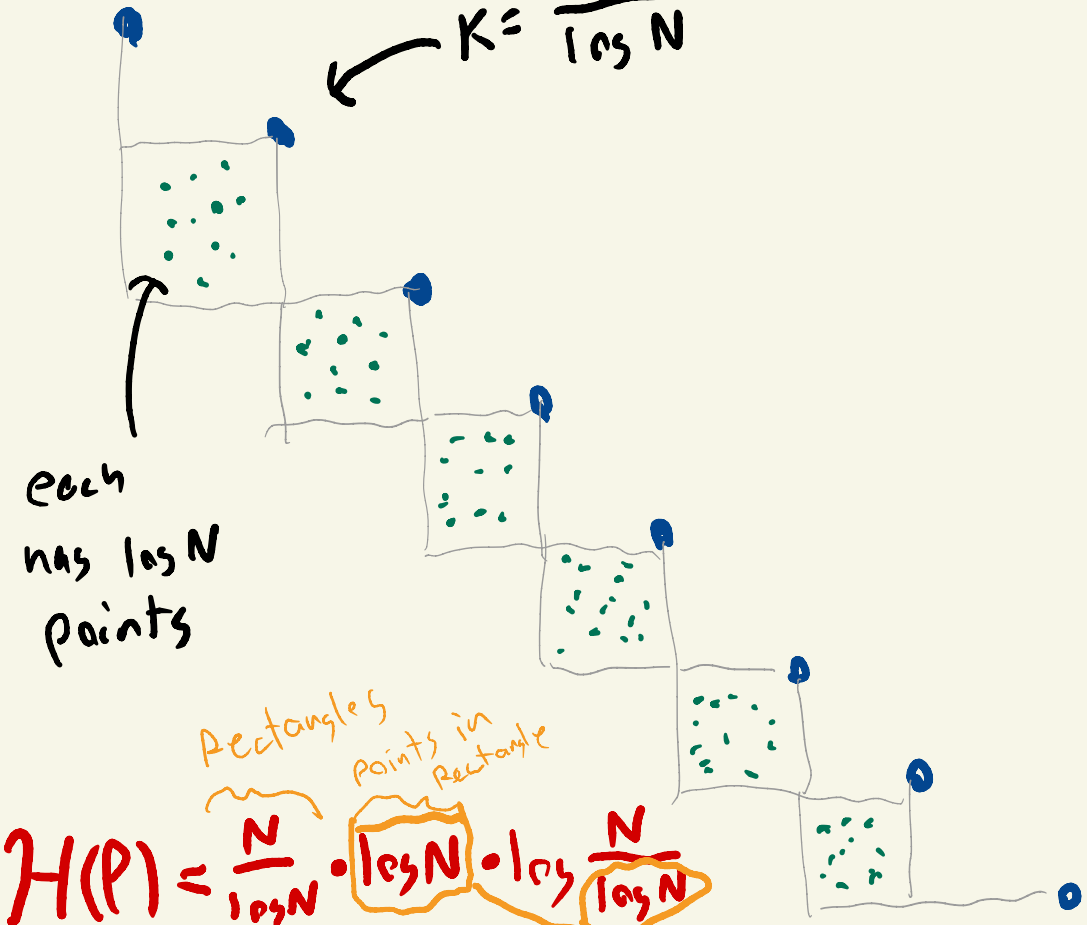


$$H(P) = \frac{N}{\log N} \cdot 1 \cdot \log \frac{N}{1}$$

$$+ \left(N - \frac{N}{\log N}\right) \log \frac{N}{N - \frac{N}{\log N}}$$

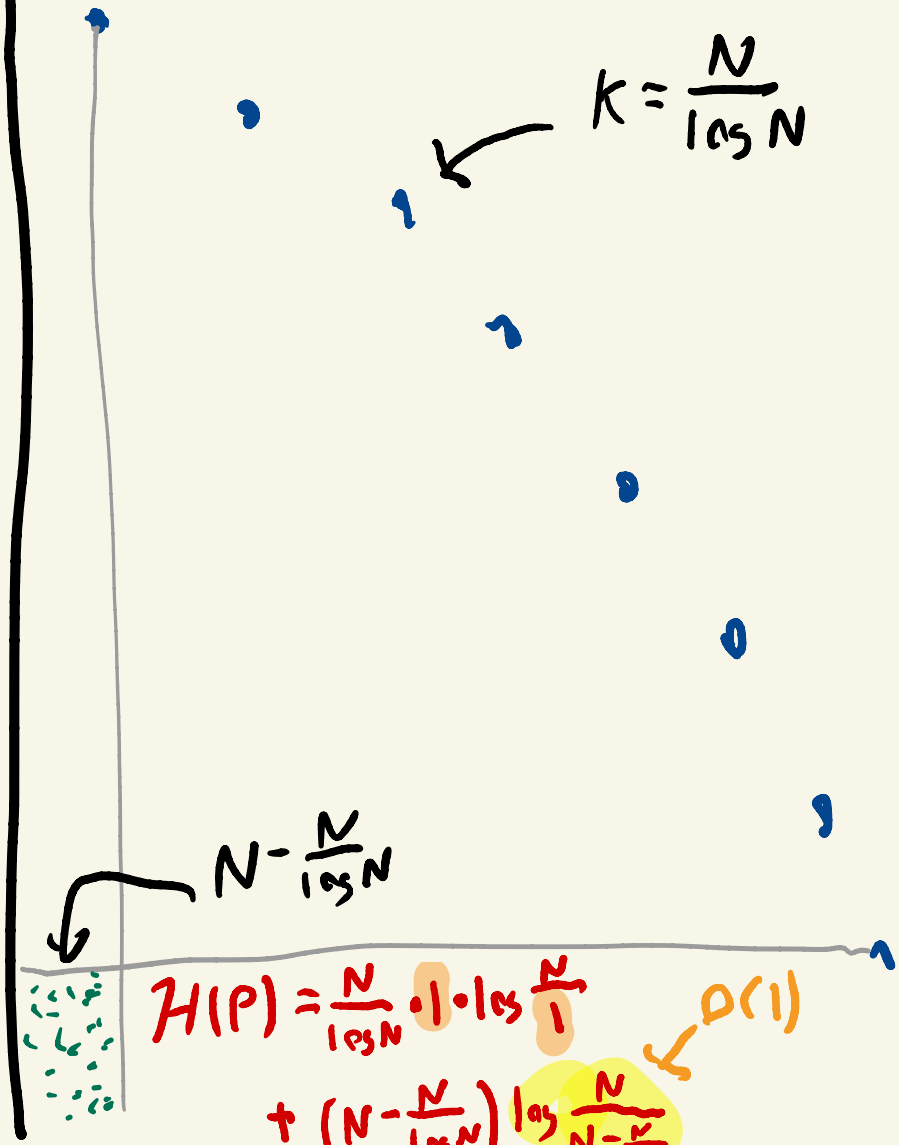
Structural Entropy

$$\sum_i |R_i| \log \frac{N}{|R_i|}$$



$$H(P) = \frac{N}{\log N} \cdot \log N \cdot \log \frac{N}{\log N}$$

$$= \Theta(N \log N)$$



$$H(P) = \frac{N}{\log N} \cdot 1 \cdot \log \frac{N}{1}$$

$O(1)$

$$+ \left(N - \frac{N}{\log N}\right) \log \frac{N}{N - \frac{N}{\log N}}$$

$$= \Theta(N)$$

Upper/Lower Bounds Overview

UB proof

$$\text{Runtime of Alg} \leq \text{Every } \Pi \sum_{R \in \Pi} |R| \cdot \log \frac{N}{|R|}$$

Upper/Lower Bounds Overview

UB proof

$$\text{Runtime of Alg} \leq \text{Every } \Pi \sum_{R \in \Pi} |R| \cdot \log \frac{N}{|R|}$$

$$\text{Runtime of alg} \leq \min_{\Pi} \sum_{R \in \Pi} |R| \cdot \log \frac{N}{|R|}$$

Upper/Lower Bounds Overview

UB proof

$$\text{Runtime of Alg} \leq \text{Every } \Pi \sum_{R \in \Pi} |R| \cdot \log \frac{N}{|R|}$$

$$\text{Runtime of alg} \leq \min_{\Pi} \sum_{R \in \Pi} |R| \cdot \log \frac{N}{|R|}$$ ← structural entropy

Upper/Lower Bounds Overview

UB proof

$$\text{Runtime of Alg} \leq \text{Every } \Pi \sum_{R \in \Pi} |R| \cdot \log \frac{N}{|R|}$$

$$\text{Runtime of alg} \leq \min_{\Pi} \sum_{R \in \Pi} |R| \cdot \log \frac{N}{|R|} \leftarrow \text{structural entropy}$$

$$\leq \text{some } \Pi \sum_{R \in \Pi} |R| \cdot \log \frac{N}{|R|}$$

Upper/Lower Bounds Overview

UB proof

Runtime of Alg

$\leq$ Every Π $\sum_{R \in \Pi} |R| \cdot \log \frac{N}{|R|}$

Runtime of alg $\leq \min_{\Pi} \sum_{R \in \Pi} |R| \cdot \log \frac{N}{|R|}$ ← structural entropy

LB proof

$\leq$ some Π $\sum_{R \in \Pi} |R| \cdot \log \frac{N}{|R|}$

$\leq$ Runtime of any algorithm

Upper/Lower Bounds Overview

UB proof

Runtime of Alg

$\leq$ Every Π $\sum_{R \in \Pi} |R| \cdot \log \frac{N}{|R|}$

Runtime of alg $\leq \min_{\Pi} \sum_{R \in \Pi} |R| \cdot \log \frac{N}{|R|}$ ← structural entropy

LB proof

$\leq$ Some Π $\sum_{R \in \Pi} |R| \cdot \log \frac{N}{|R|}$

$\leq$ Runtime of any algorithm

$\leq$ Runtime of alg

Upper/Lower Bounds Overview

UB proof

Runtime of Alg

$\leq$ Every Π $\sum_{R \in \Pi} |R| \cdot \log \frac{N}{|R|}$

Runtime of alg $\leq \min_{\Pi} \sum_{R \in \Pi} |R| \cdot \log \frac{N}{|R|}$ ← structural entropy

LB proof

$\leq$ Some Π $\sum_{R \in \Pi} |R| \cdot \log \frac{N}{|R|}$

$\leq$ Runtime of any algorithm

$\leq$ Runtime of alg

$\leq$ really is O

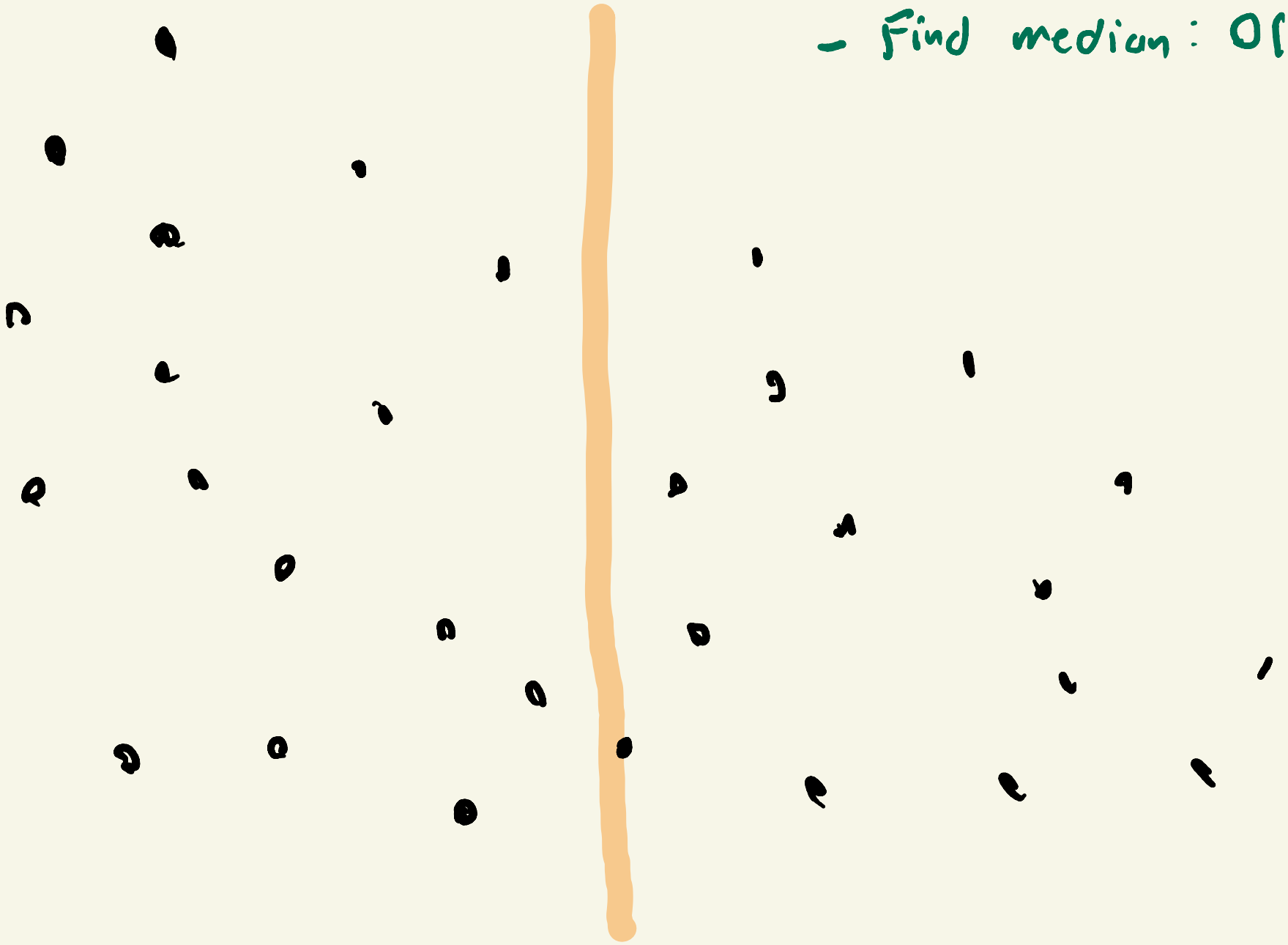
What is this magical algorithm that achieves the min entropy?

What is this magical algorithm that achieves the min entropy?

[Kirkpatrick Seidel 1985]

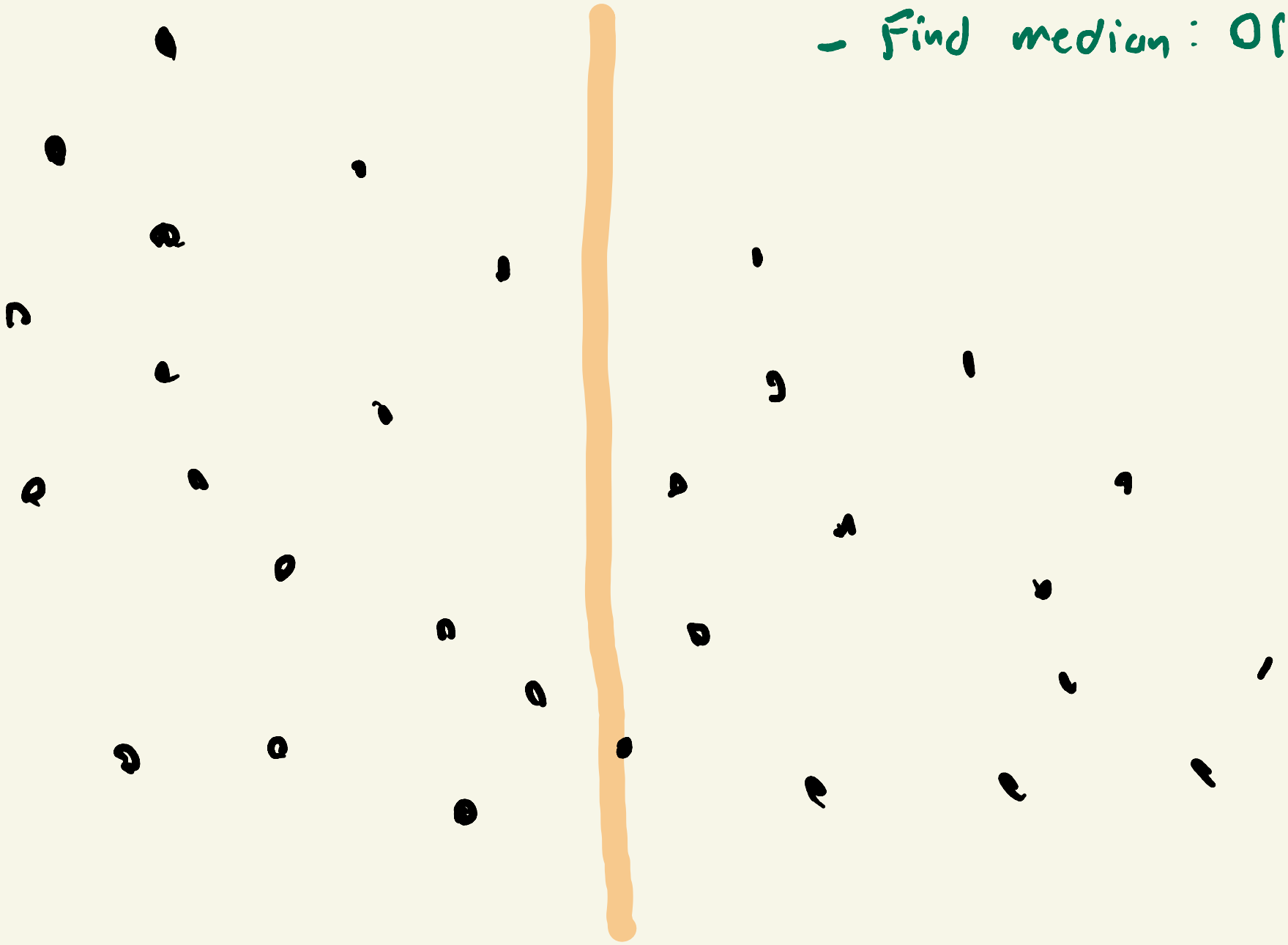
What is this magical algorithm that achieves the min entropy?

- Find median: O(N)

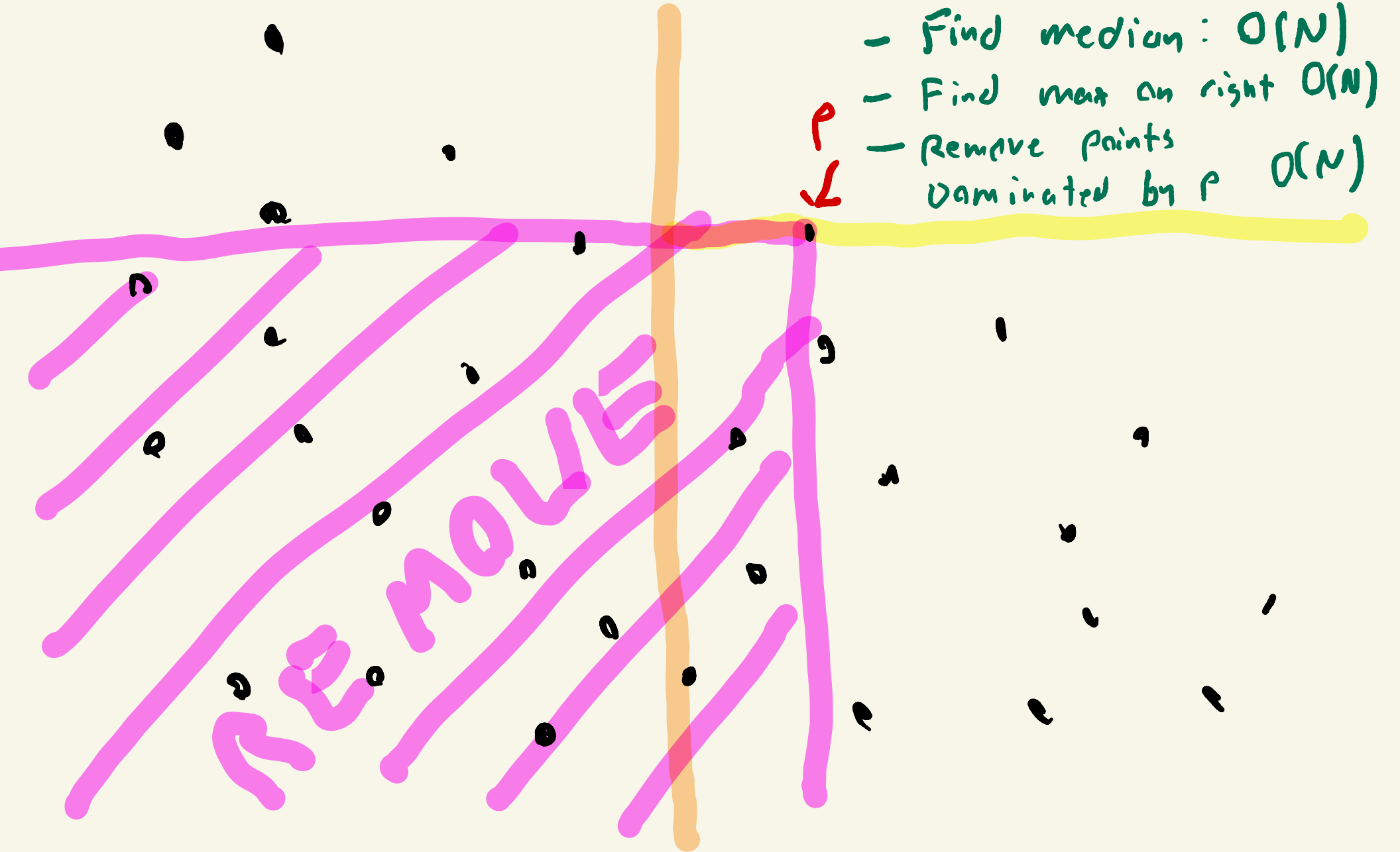


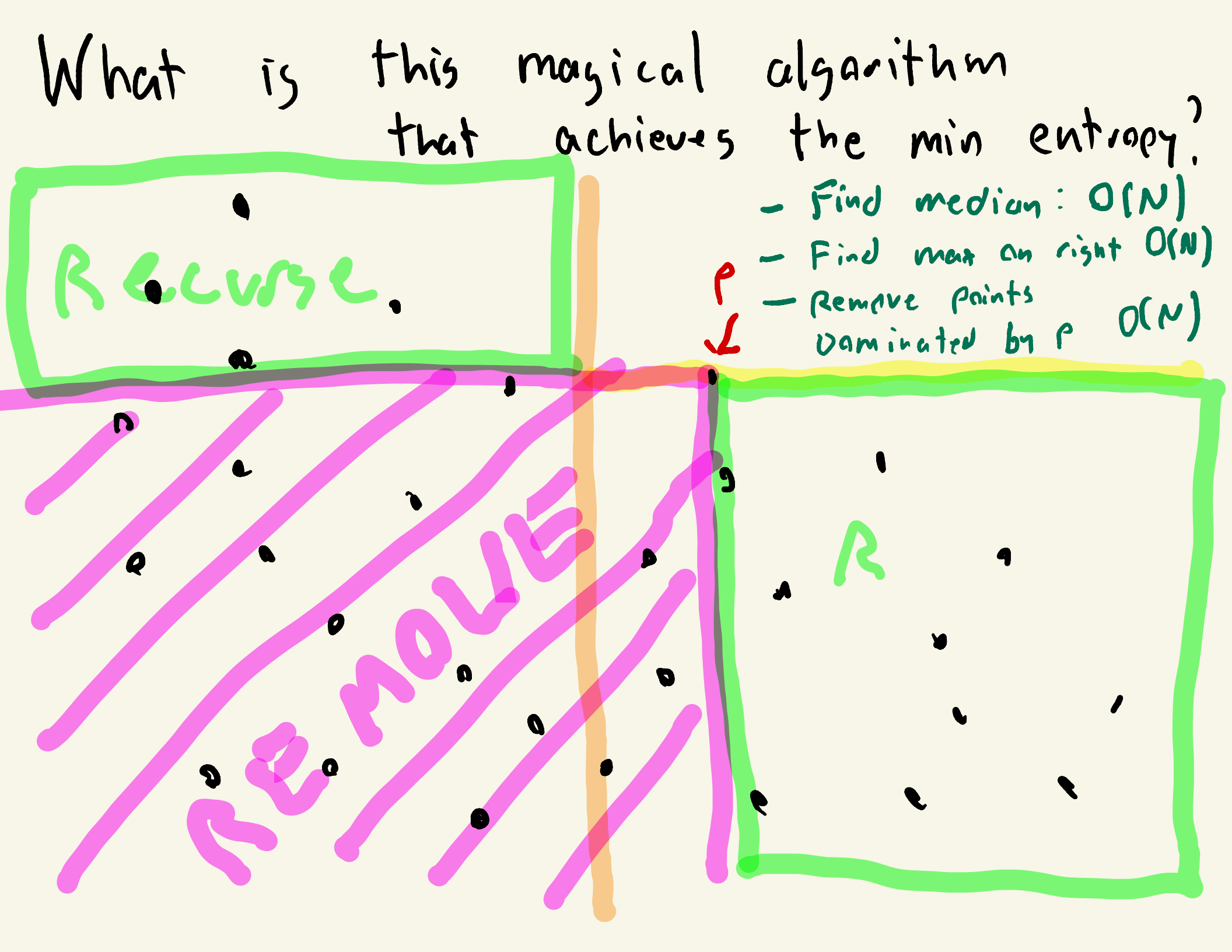
What is this magical algorithm that achieves the min entropy?

- Find median: O(N)



What is this magical algorithm that achieves the min entropy?



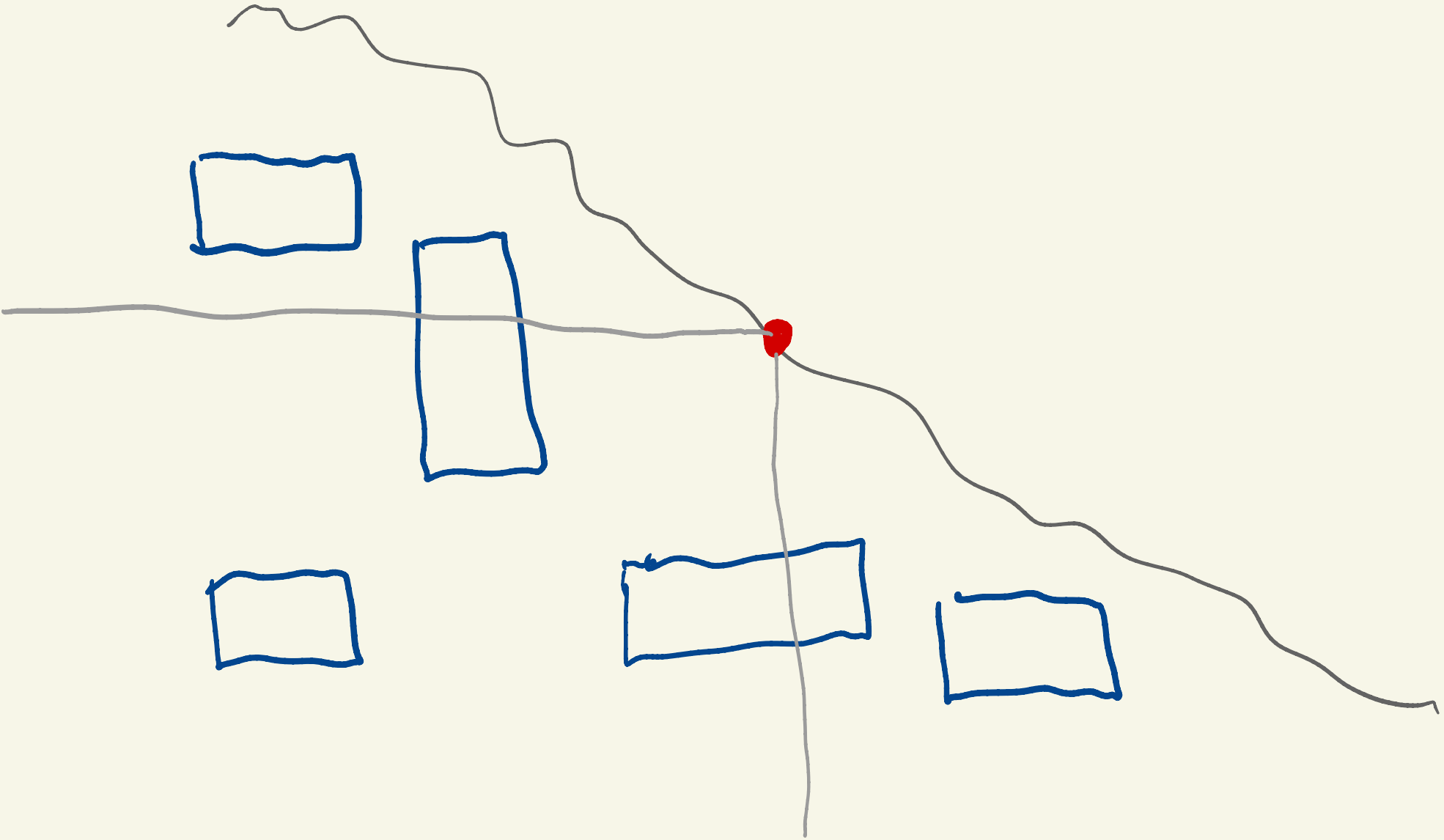

What is this magical algorithm that achieves the min entropy?
- Find median: O(N)
- Find max on right O(N)
- Remove points dominated by p O(N)
p
Recurse.
REMOVE
R

Runtime want $\sum_i |R_i| \log \frac{N}{|R_i|}$

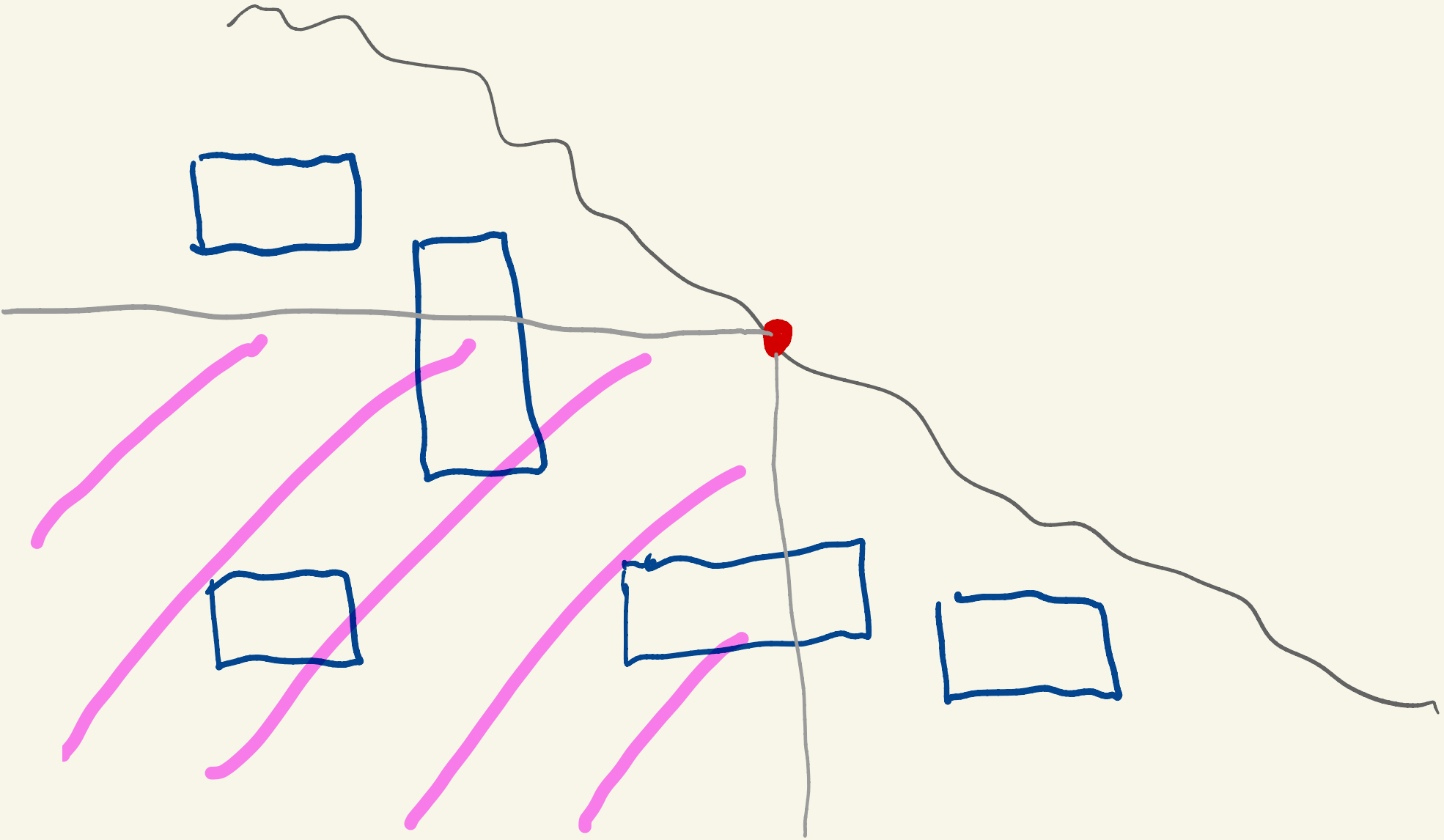
Runtime want $\sum_i |R_i| \log \frac{N}{|R_i|}$

this is $|R| \log \frac{N}{|R|}$ for any R

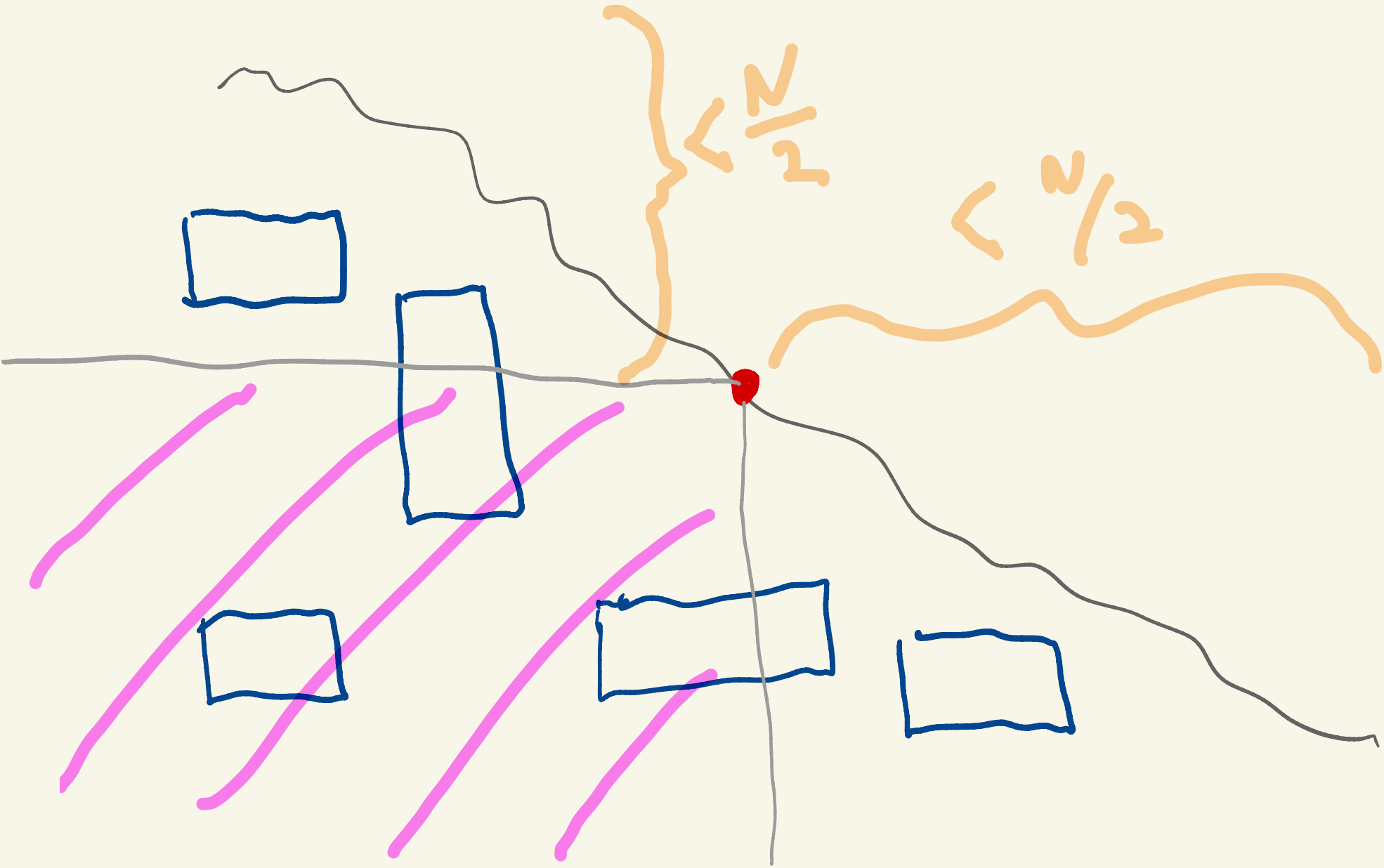
What happens to a Rectangle



What happens to a Rectangle



What happens to a Rectangle



How Long Can a rectangle with $|R|$ Points Survive?

N

$N/2$

$N/4$

$\vdots$

$N/2^i$

$\vdots$

$|R|$

How Long Can a rectangle with $|R|$ Points Survive?

N

$N/2$

$N/4$

$\vdots$

$N/2^i$

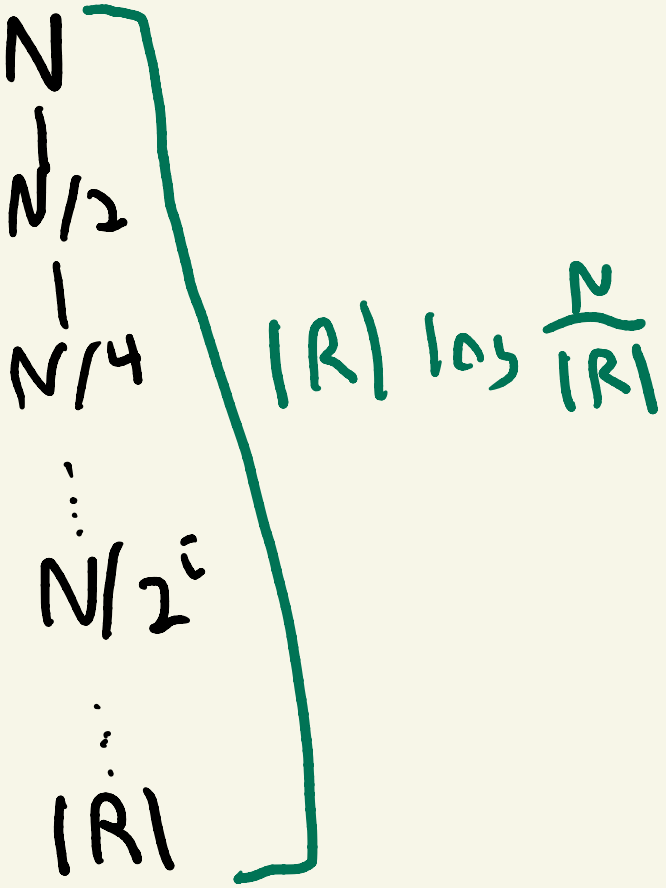
$\vdots$

$|R|$

$$\frac{N}{2^i} = |R|$$

$$i = \log \frac{N}{|R|}$$

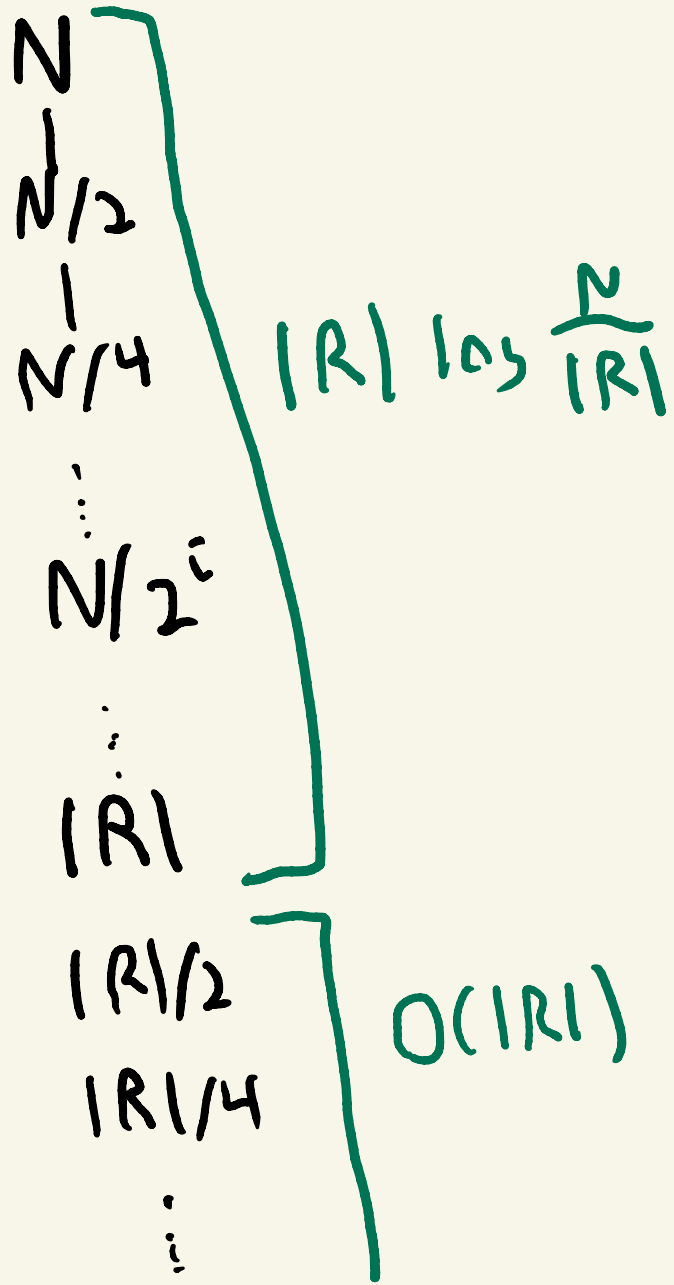
How Long Can a rectangle with $|R|$ Points Survive?



$$\frac{N}{2^i} = |R|$$

$$i = \log \frac{N}{|R|}$$

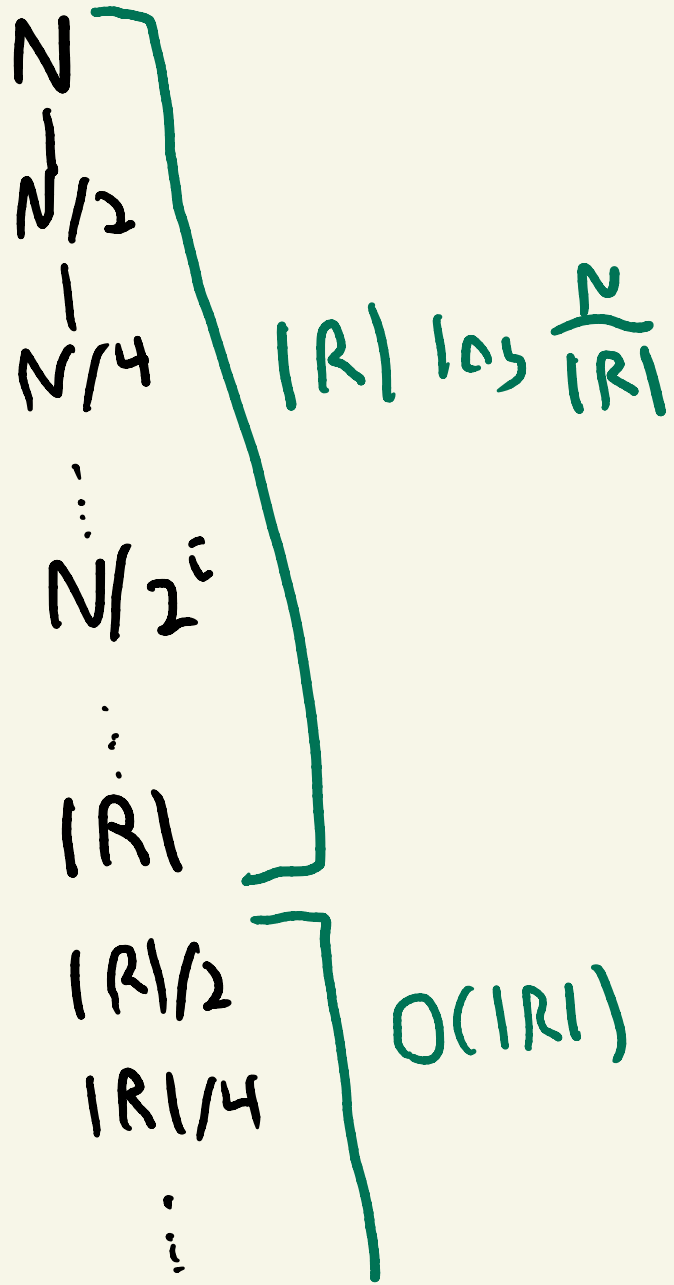
How Long Can a rectangle with $|R|$ Points Survive?



$$\frac{N}{2^i} = |R|$$

$$i = \log \frac{N}{|R|}$$

How Long Can a rectangle with $|R|$ Points Survive?



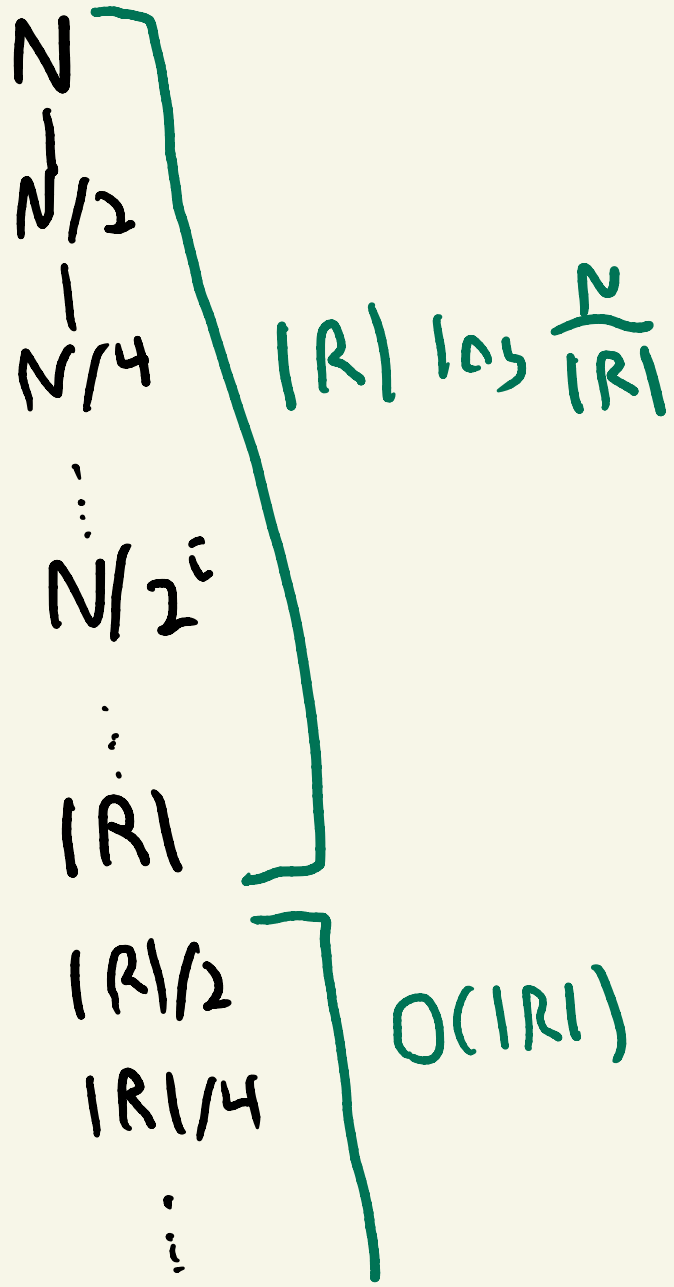
$$\frac{N}{2^i} = |R|$$

$$i = \log \frac{N}{|R|}$$

Total time one Rect:

$$O\left(|R| \log \frac{N}{|R|}\right)$$

How Long Can a rectangle with $|R|$ Points Survive?



$$\frac{N}{2^i} = |R|$$

$$i = \log \frac{N}{|R|}$$

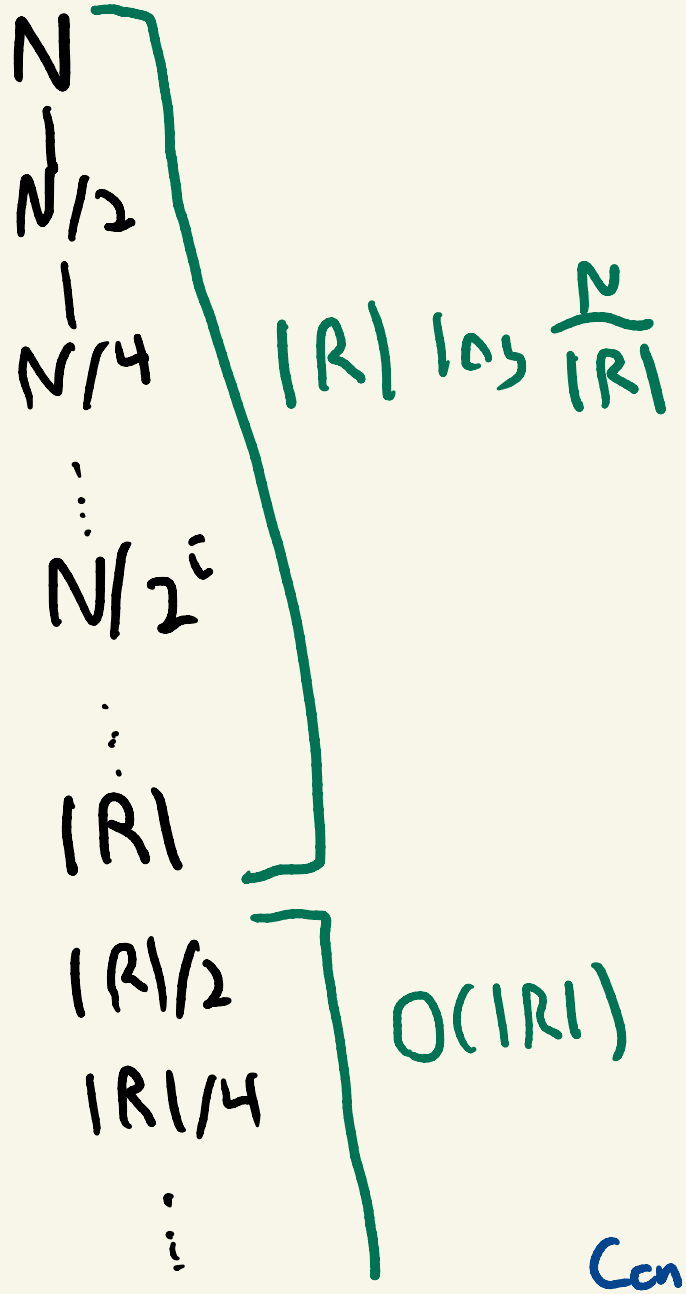
Total time one Rect:

$$O\left(|R| \log \frac{N}{|R|}\right)$$

Total time any partition:

$$O(\mathcal{H}(\Pi)) = O\left(\sum_{R \in \Pi} |R| \log \frac{N}{|R|}\right)$$

How Long Can a rectangle with $|R|$ Points Survive?



$$\frac{N}{2^i} = |R|$$

$$i = \log \frac{N}{|R|}$$

Total time one Rect:

$$O\left(|R| \log \frac{N}{|R|}\right)$$

Total time any partition:

$$O(\mathcal{H}(\Pi)) = O\left(\sum_{R \in \Pi} |R| \log \frac{N}{|R|}\right)$$

Can assume min partition $O(\mathcal{H}(P))$

Now, the lower bound

Now, the lower bound

For any set P

Now, the lower bound

For any set P

For any algorithm A

Now, the lower bound

For any set P

For any algorithm A

There is an ordering of P, $\mathcal{P}$

Now, the lower bound

For any set P

For any algorithm A

There is an ordering of P, $\mathcal{P}$

Such that

Now, the lower bound

For any set P

For any algorithm A

There is an ordering of P, $\mathcal{P}$

Such that

Running A on $\mathcal{P}$ takes time

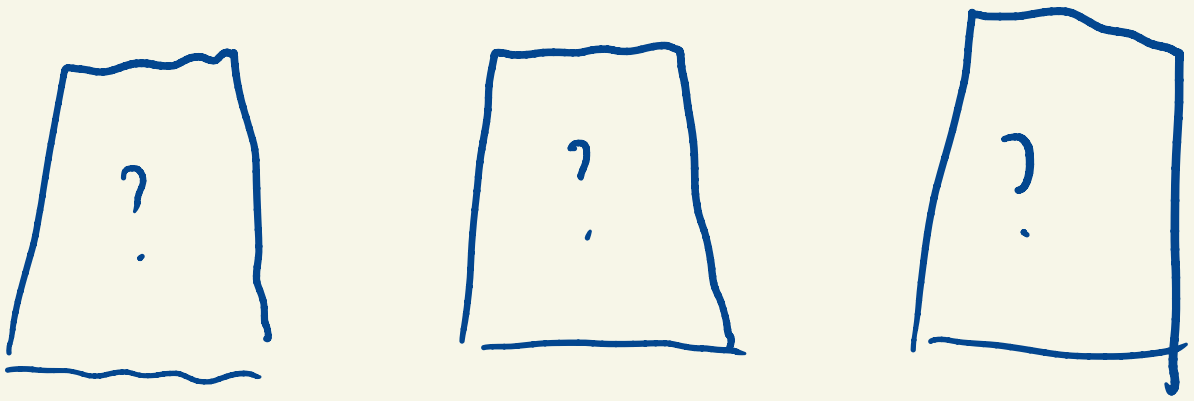
$$\Omega(\mathcal{H}(P))$$

Adversary Argument: A simple example

2 Blue Balls

1 Red

Find the red!

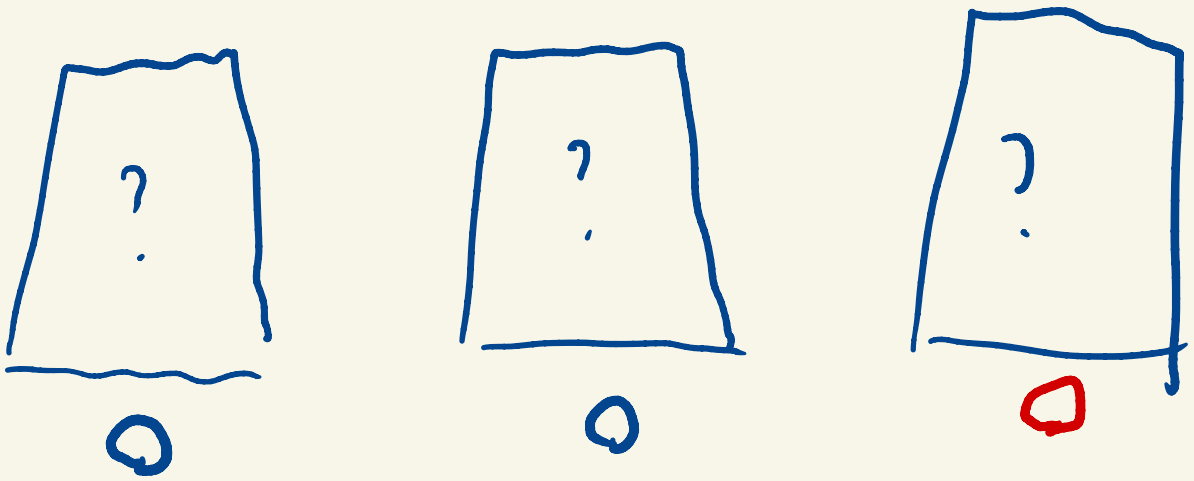


Adversary Argument: A simple example

2 Blue Balls

1 Red

Find the red!

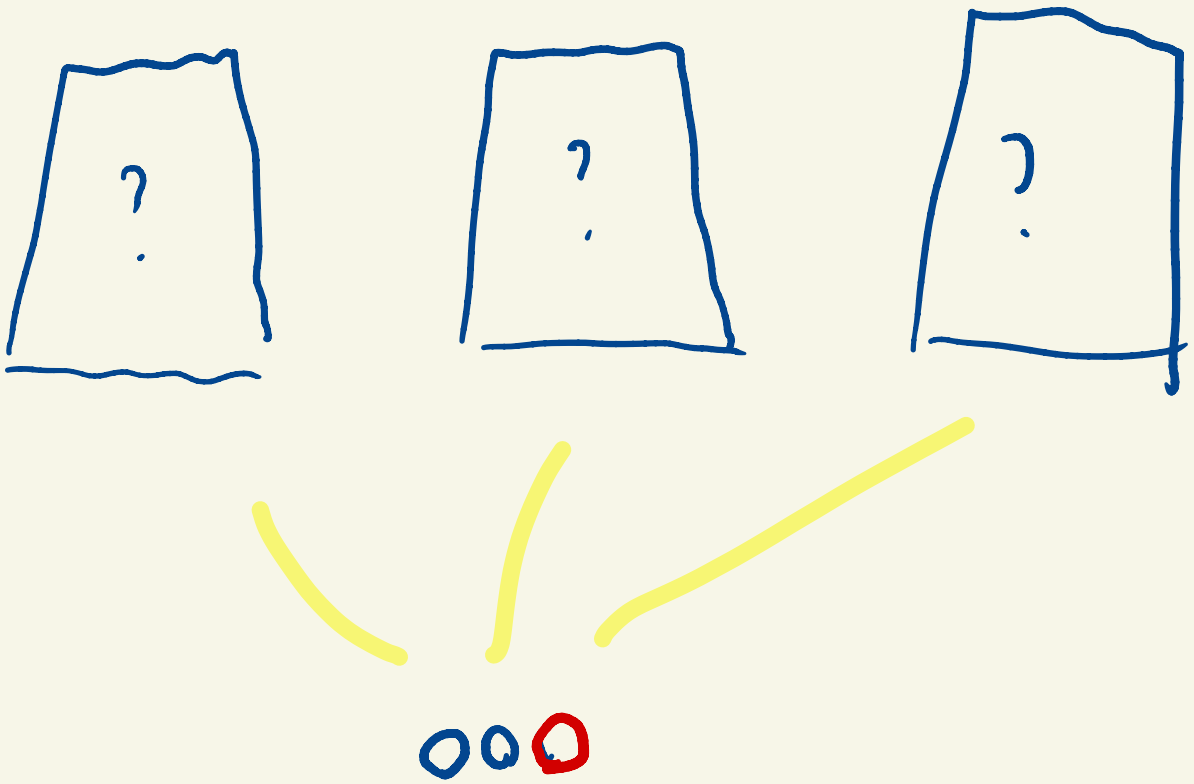


Adversary Argument: A simple example

2 Blue Balls

1 Red

Find the red!

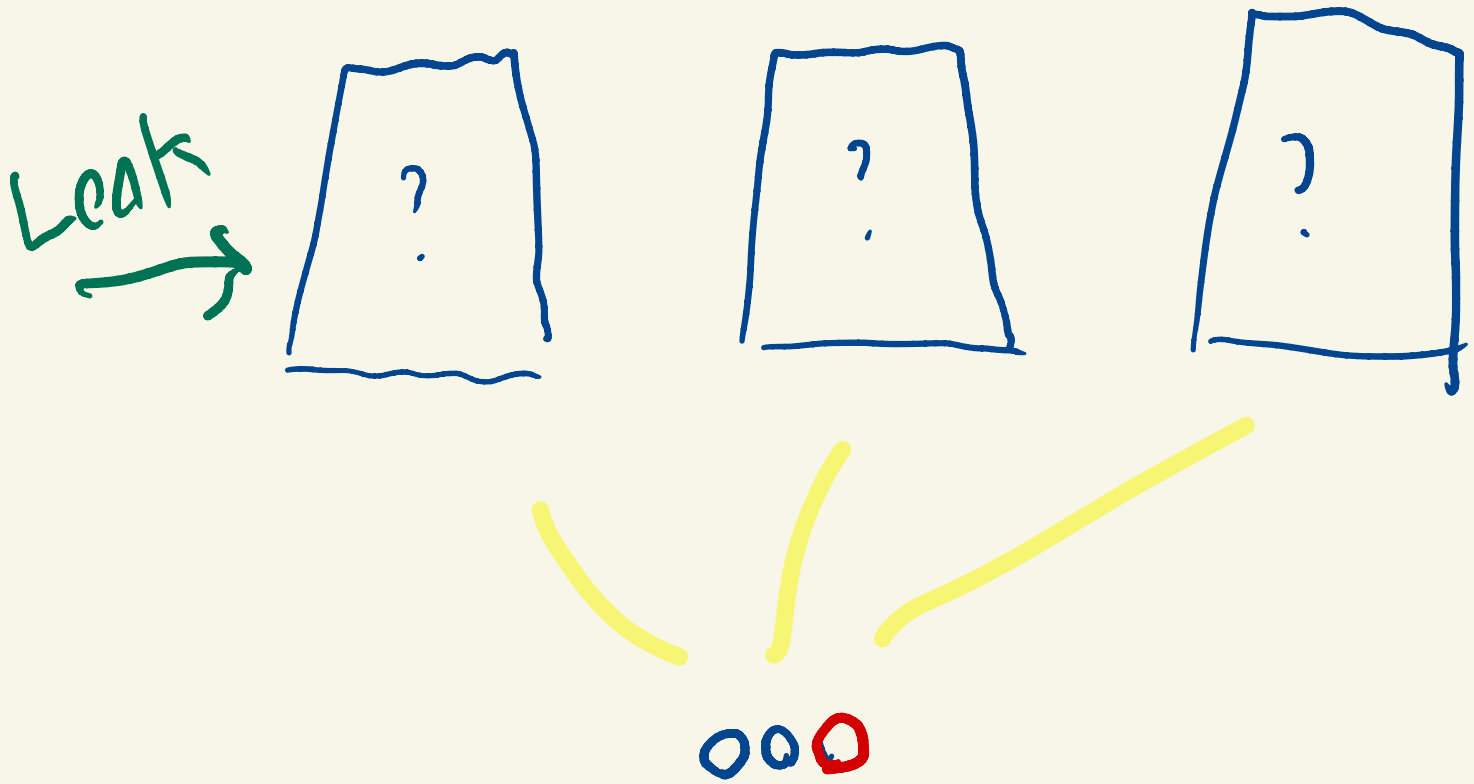


Adversary Argument: A simple example

2 Blue Balls

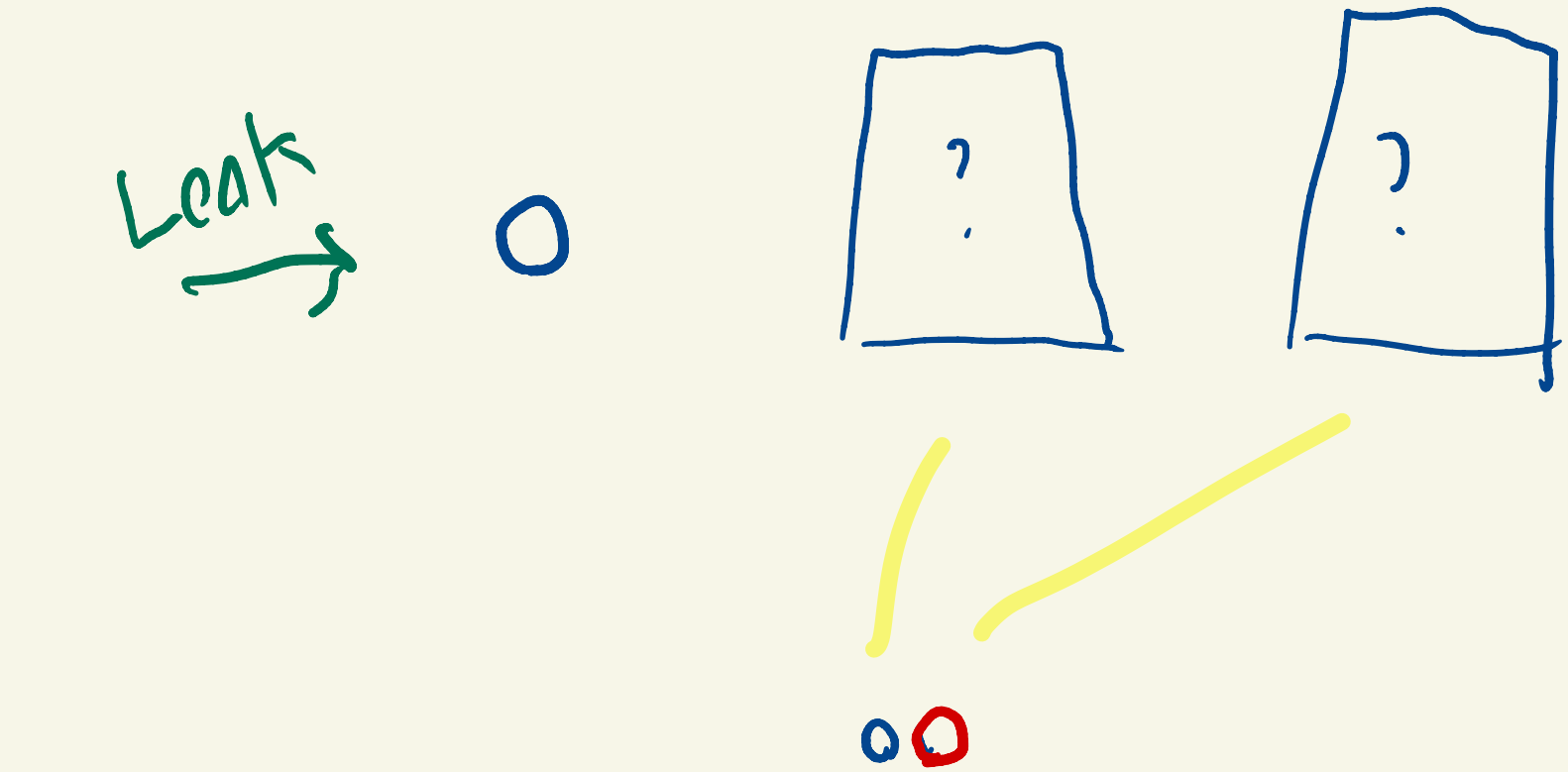
1 Red

Find the red!



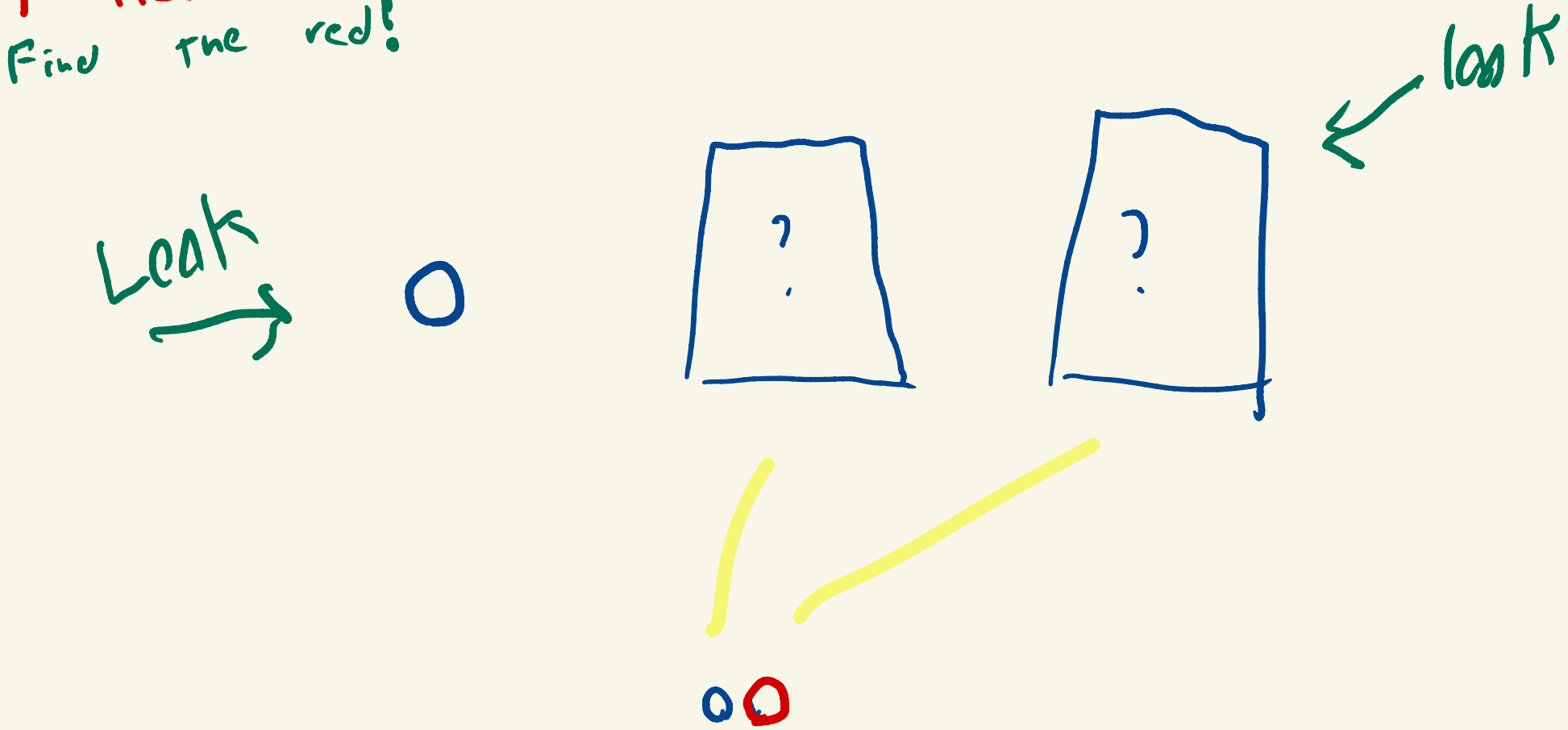
Adversary Argument: A simple example

2 Blue Balls
1 Red
Find the red!



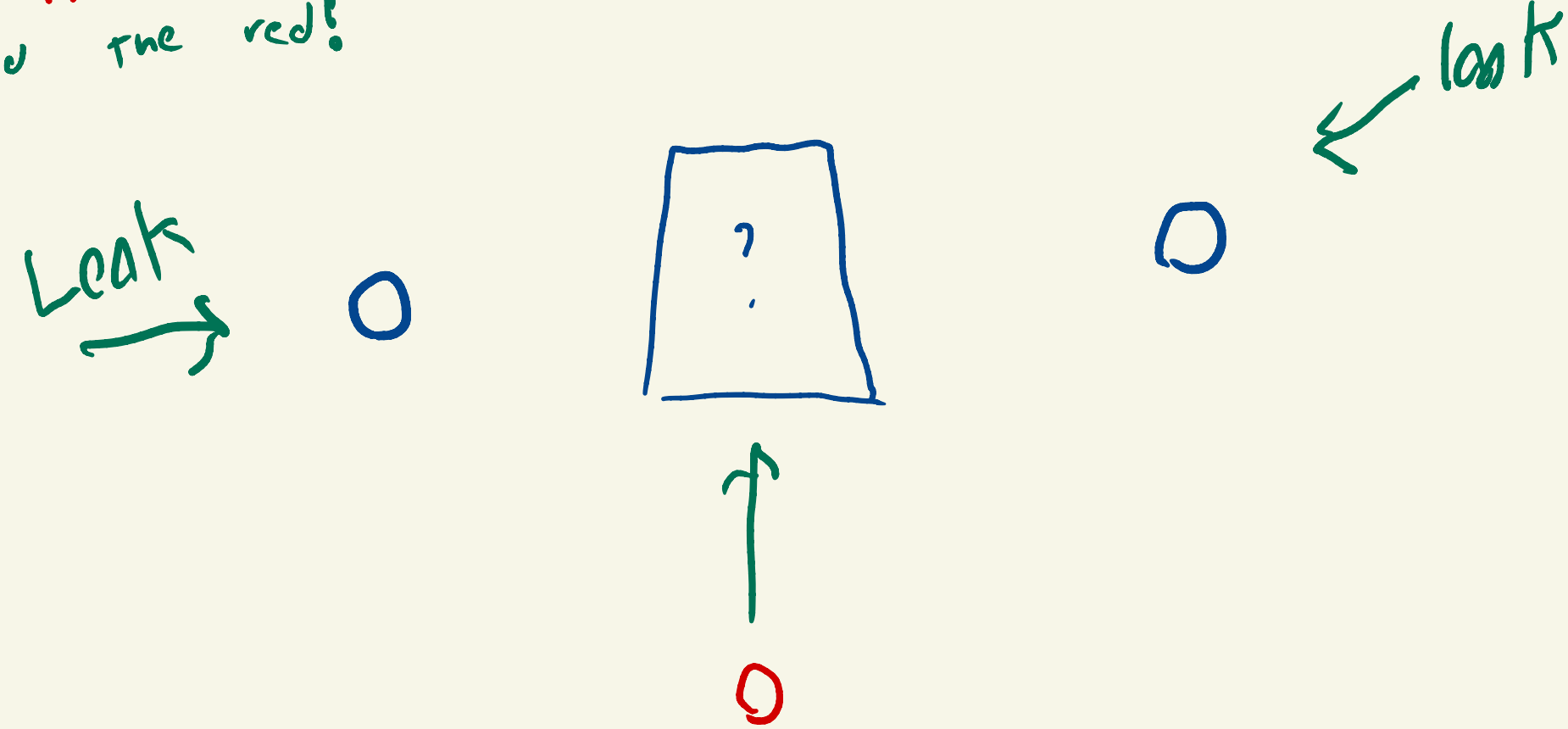
Adversary Argument: A simple example

2 Blue Balls
1 Red
Find the red!



Adversary Argument: A simple example

2 Blue Balls
1 Red
Find the red!



Moral: I can change the instance so long as

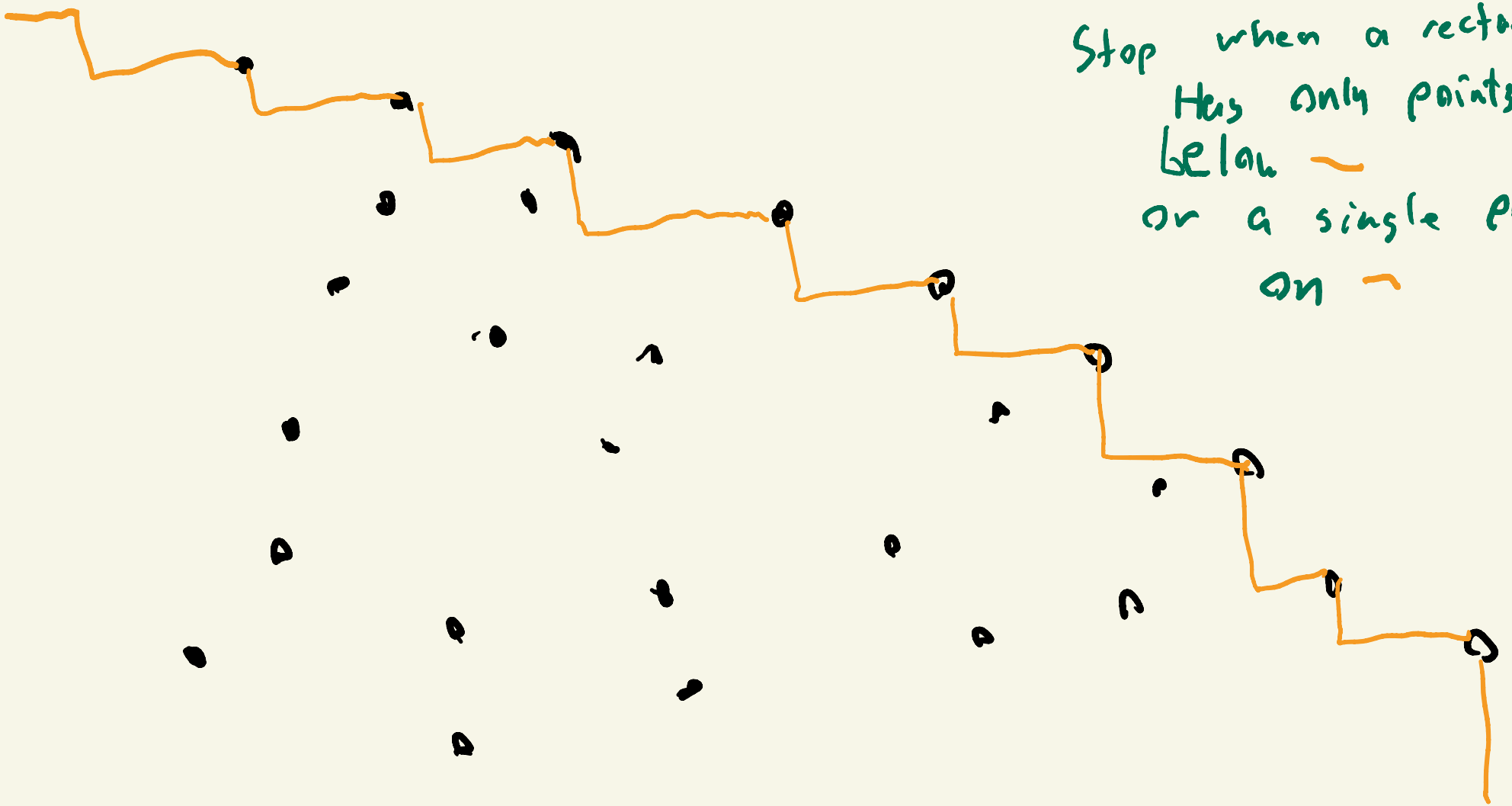
- It is still valid
- It is consistant with what the algorithm knows.

Idea: Give one rectangular partition Π and show that any algorithm* must use $\Omega(\mathcal{H}(\Pi))$ time

Idea: Give one rectangular partition π and show that any algorithm* must use $\Omega(\mathcal{H}(\pi))$ time

* any algorithm that works by comparing x or y coordinates of points

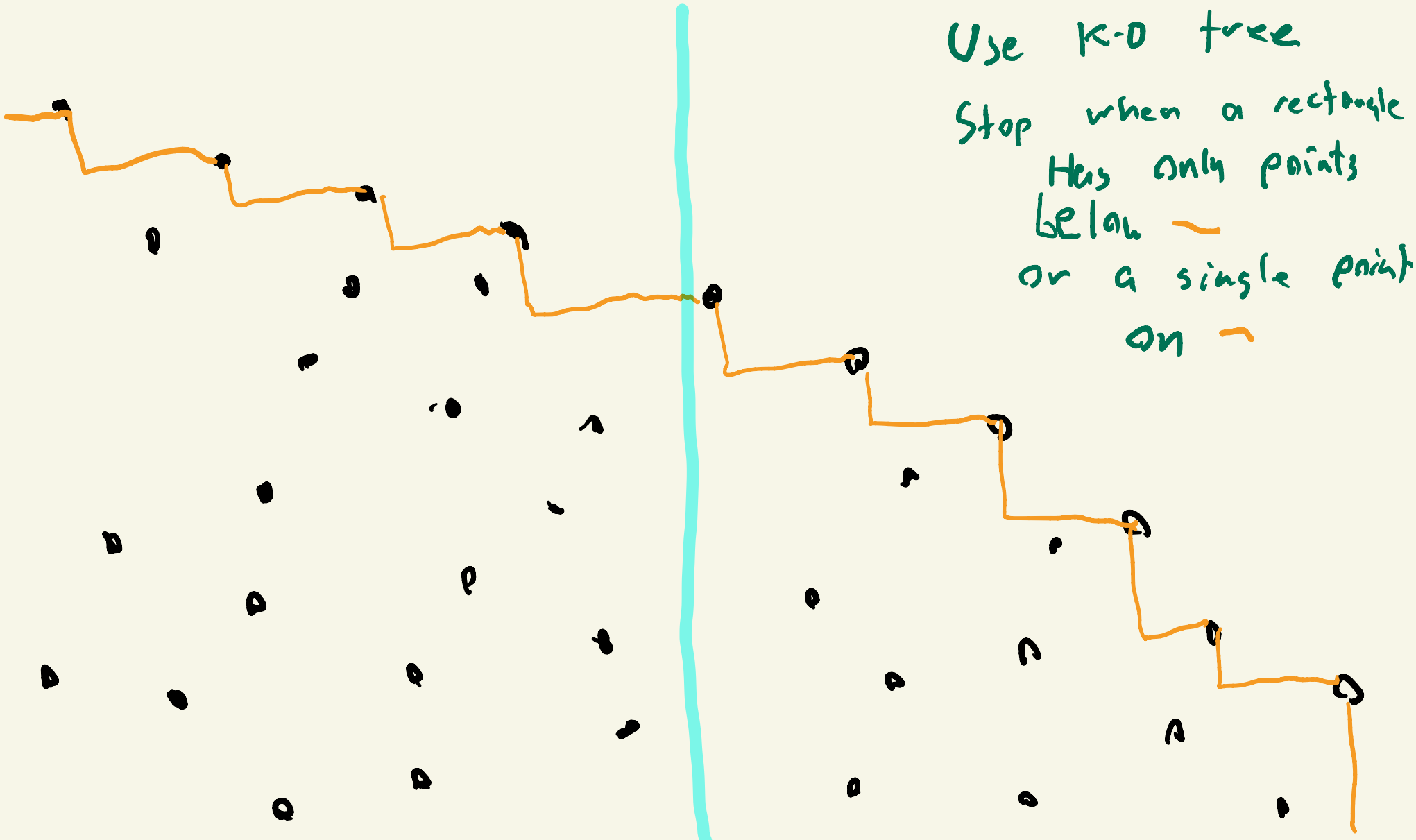
Idea: Give one rectangular Partition π and show that any algorithm* must use $\Omega(\mathcal{H}(\pi))$ time



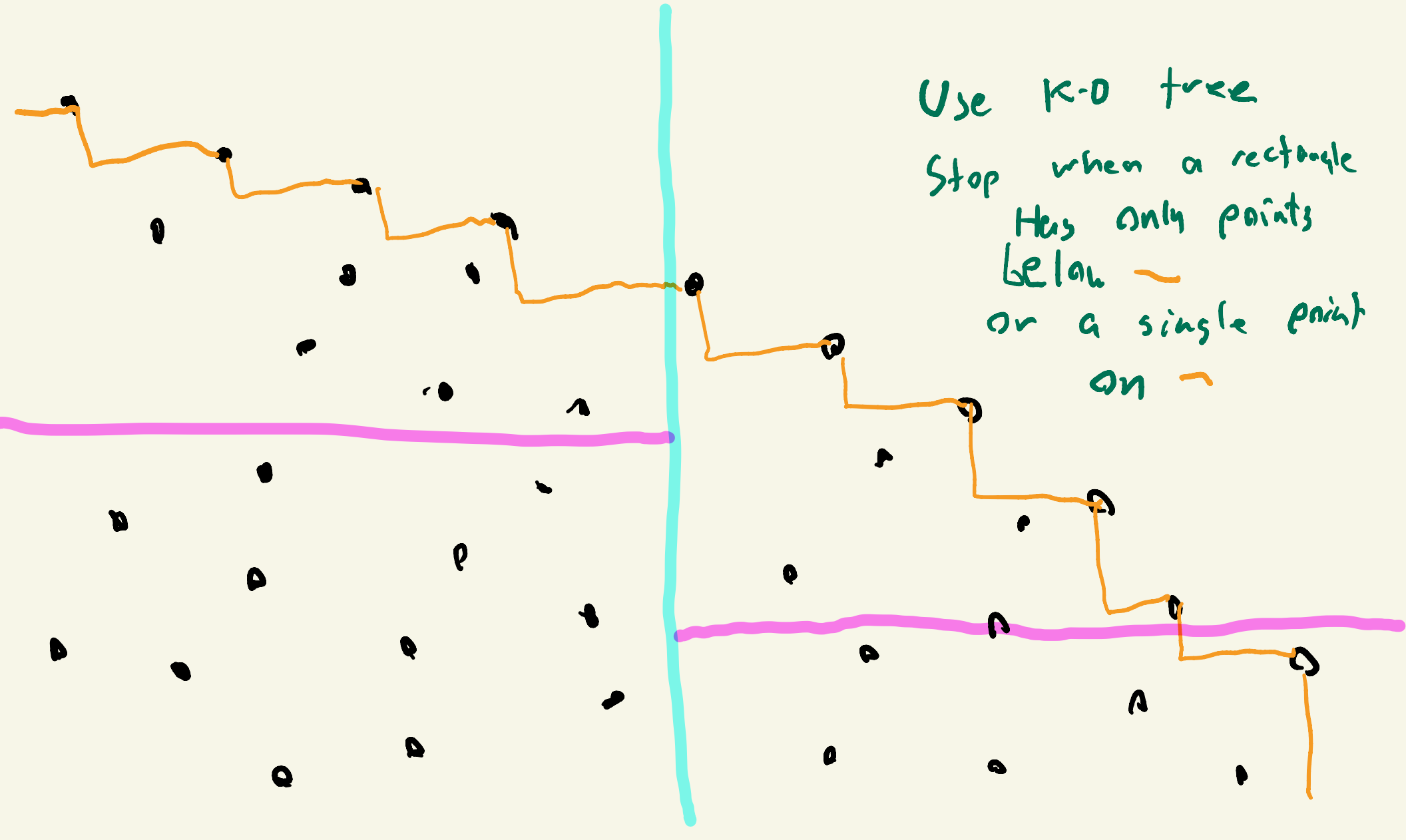
Use K-D tree

Stop when a rectangle has only points below — or a single point on —

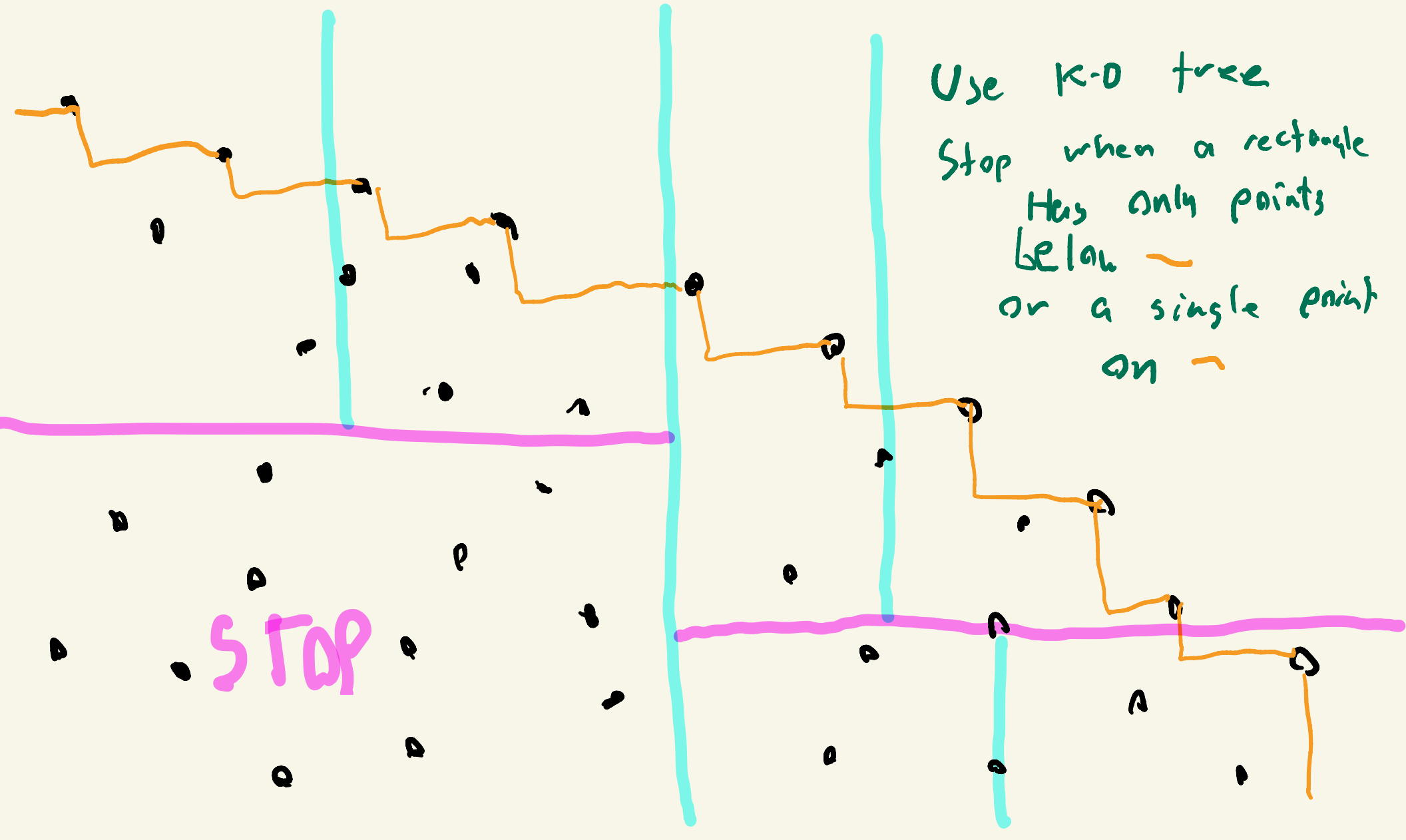
Idea: Give one rectangular Partition π and show that any algorithm* must use $\Omega(\mathcal{H}(\pi))$ time



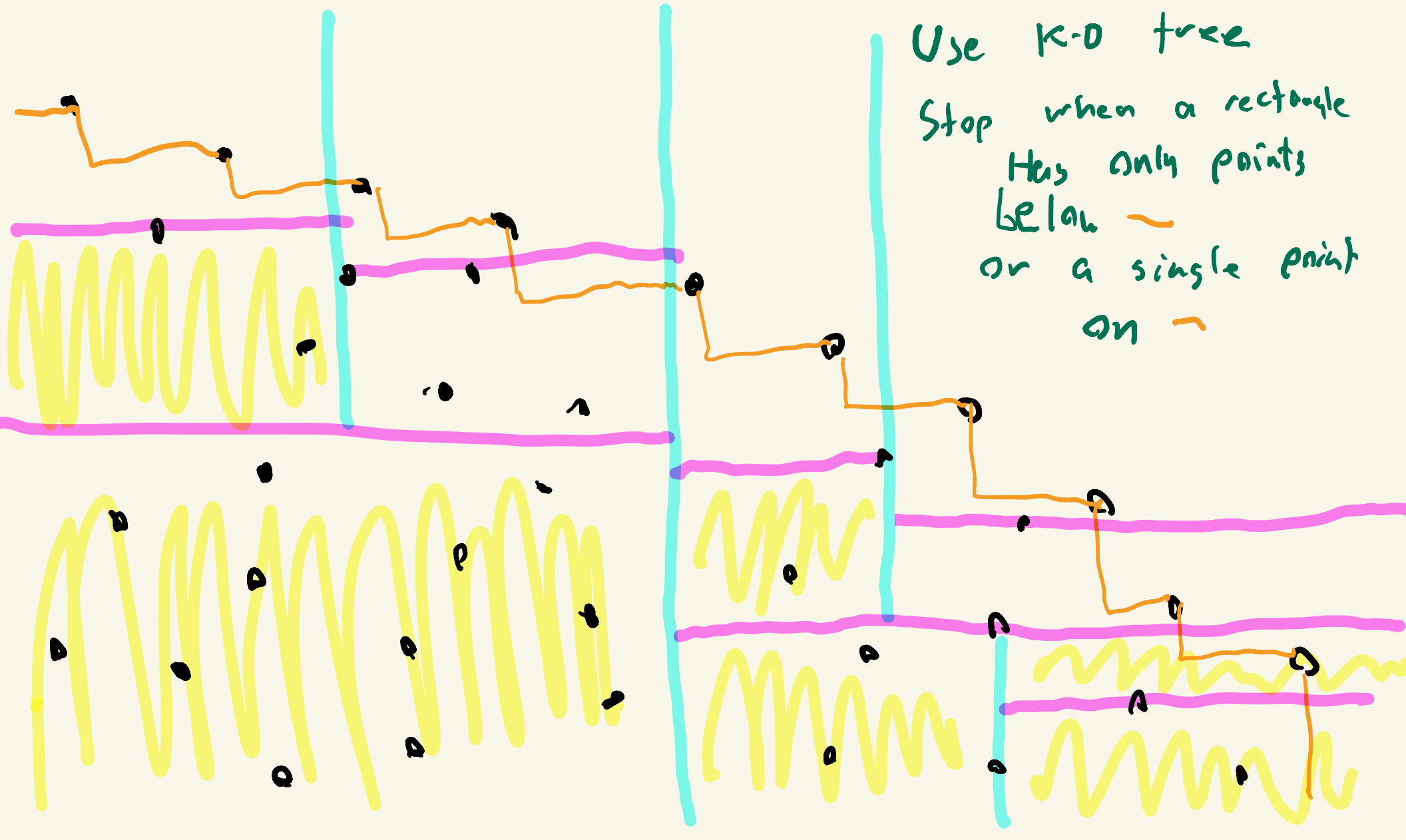
Idea: Give one rectangular Partition Π and show that any algorithm* must use $\Omega(\mathcal{H}(\Pi))$ time



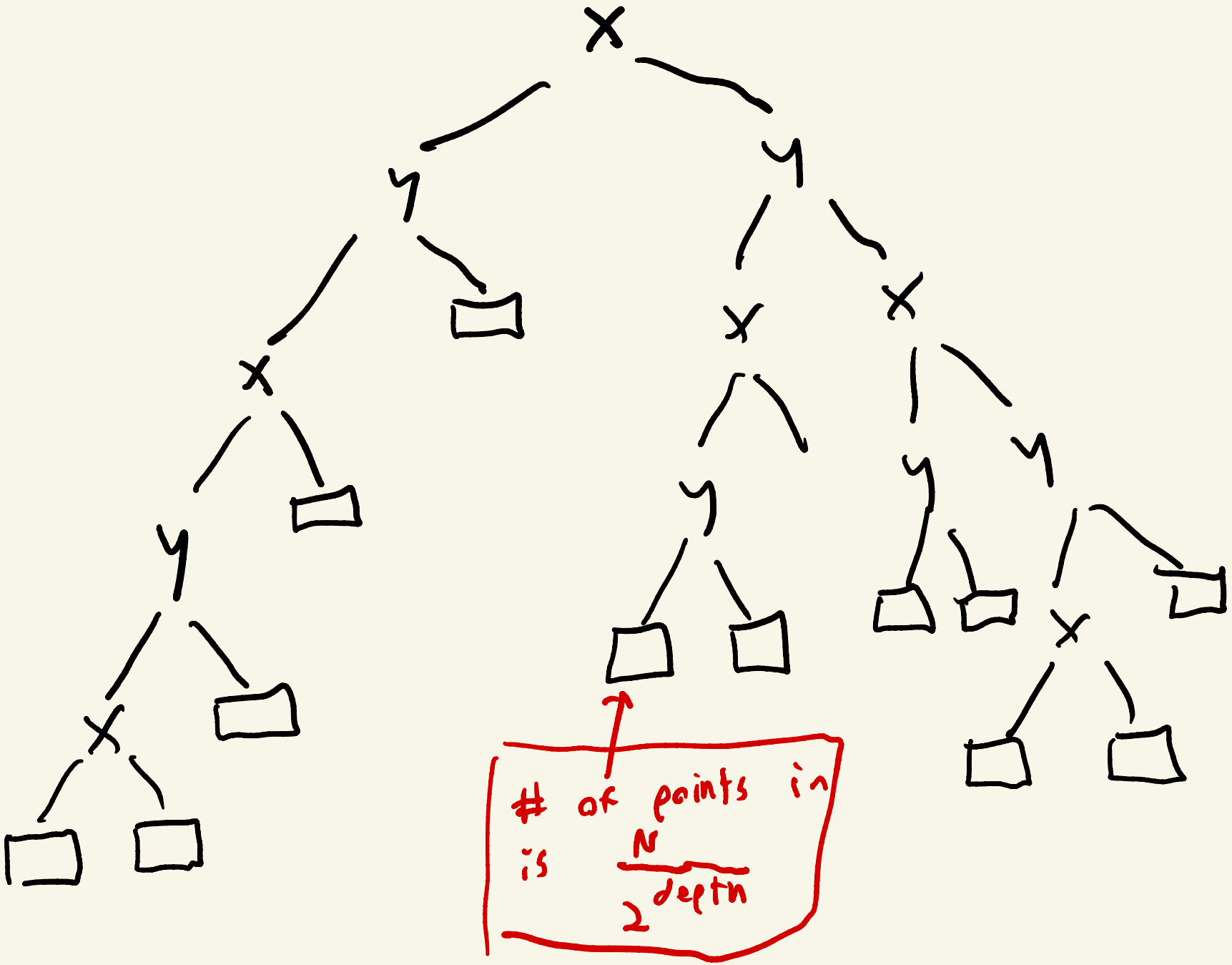
Idea: Give one rectangular Partition π and show that any algorithm* must use $\Omega(\mathcal{H}(\pi))$ time



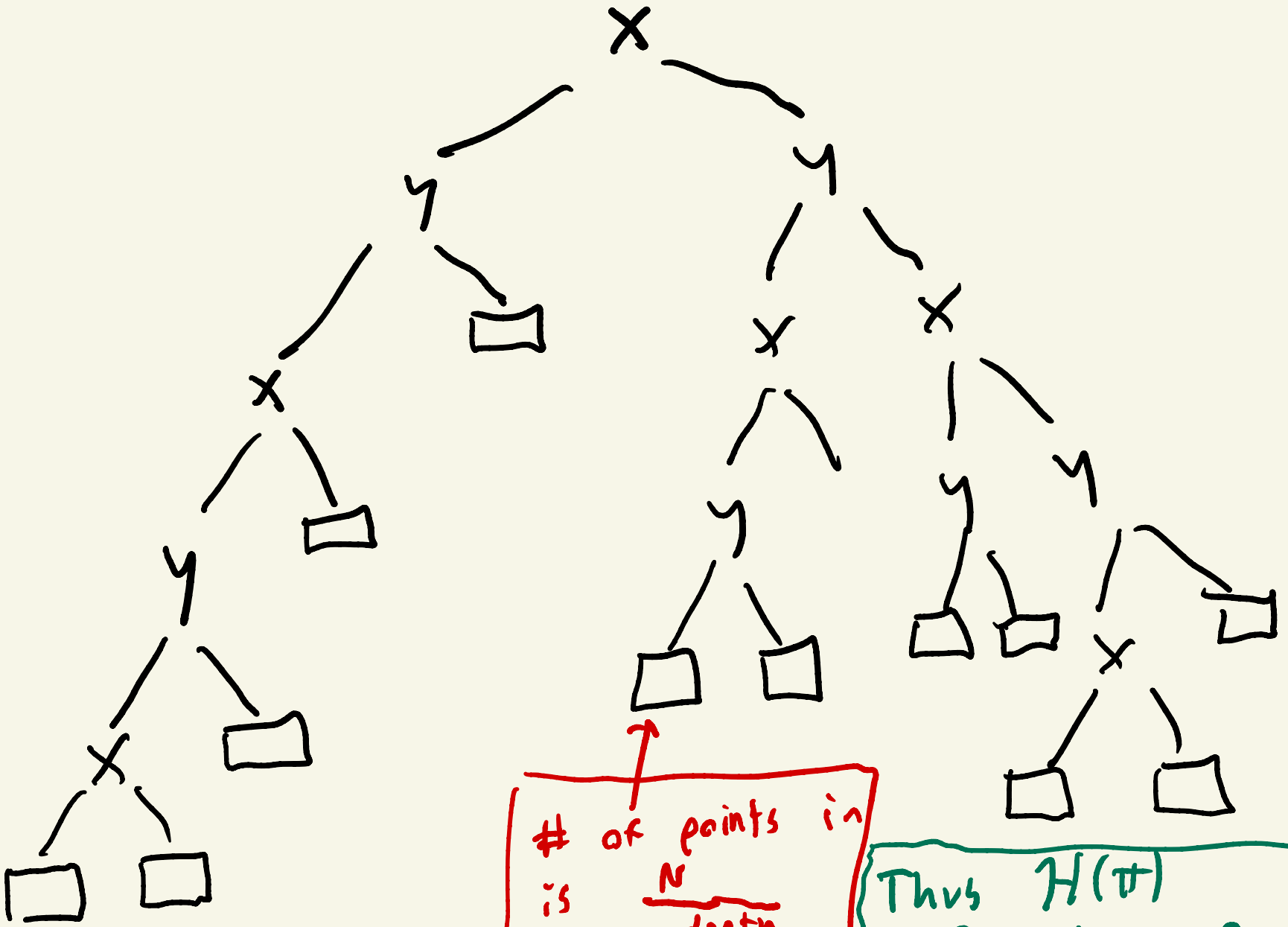
Idea: Give one rectangular Partition Π and show that any algorithm* must use $\Omega(\mathcal{H}(\Pi))$ time



Tree View of K-D tree



Tree View of K-D tree

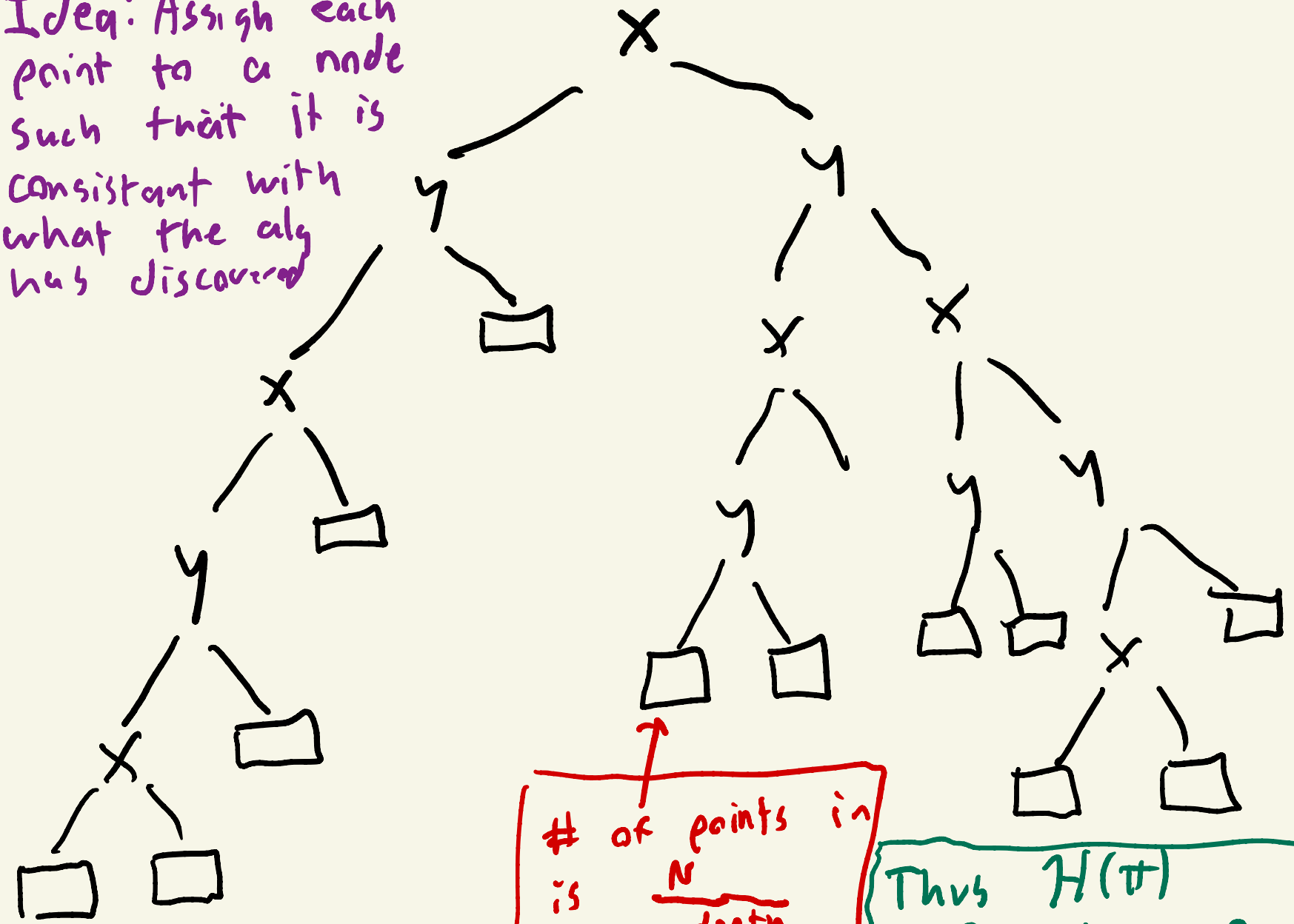


# of points in is $\frac{N}{2^{depth}}$

Thus $H(\pi)$ is Σ points in R · depth R

Tree View of K-D tree

Idea: Assign each point to a node such that it is consistant with what the alg has discovered



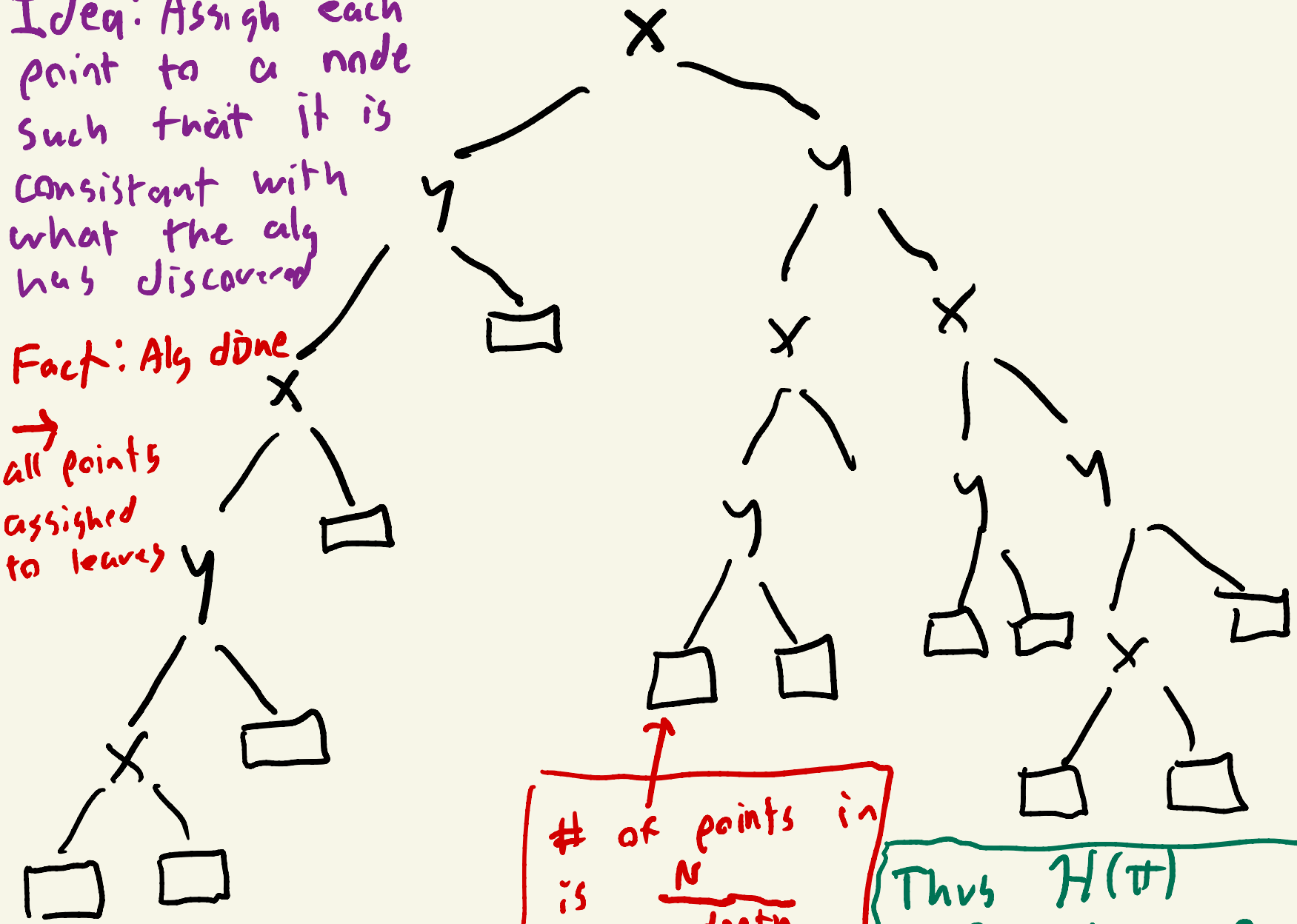
# of points in is $\frac{N}{2^{depth}}$

Thus $H(\pi)$ is $\sum$ points in R • depth R

Tree View of K-D tree

Idea: Assign each point to a node such that it is consistant with what the alg has discovered

Fact: Alg done → all points assigned to leaves



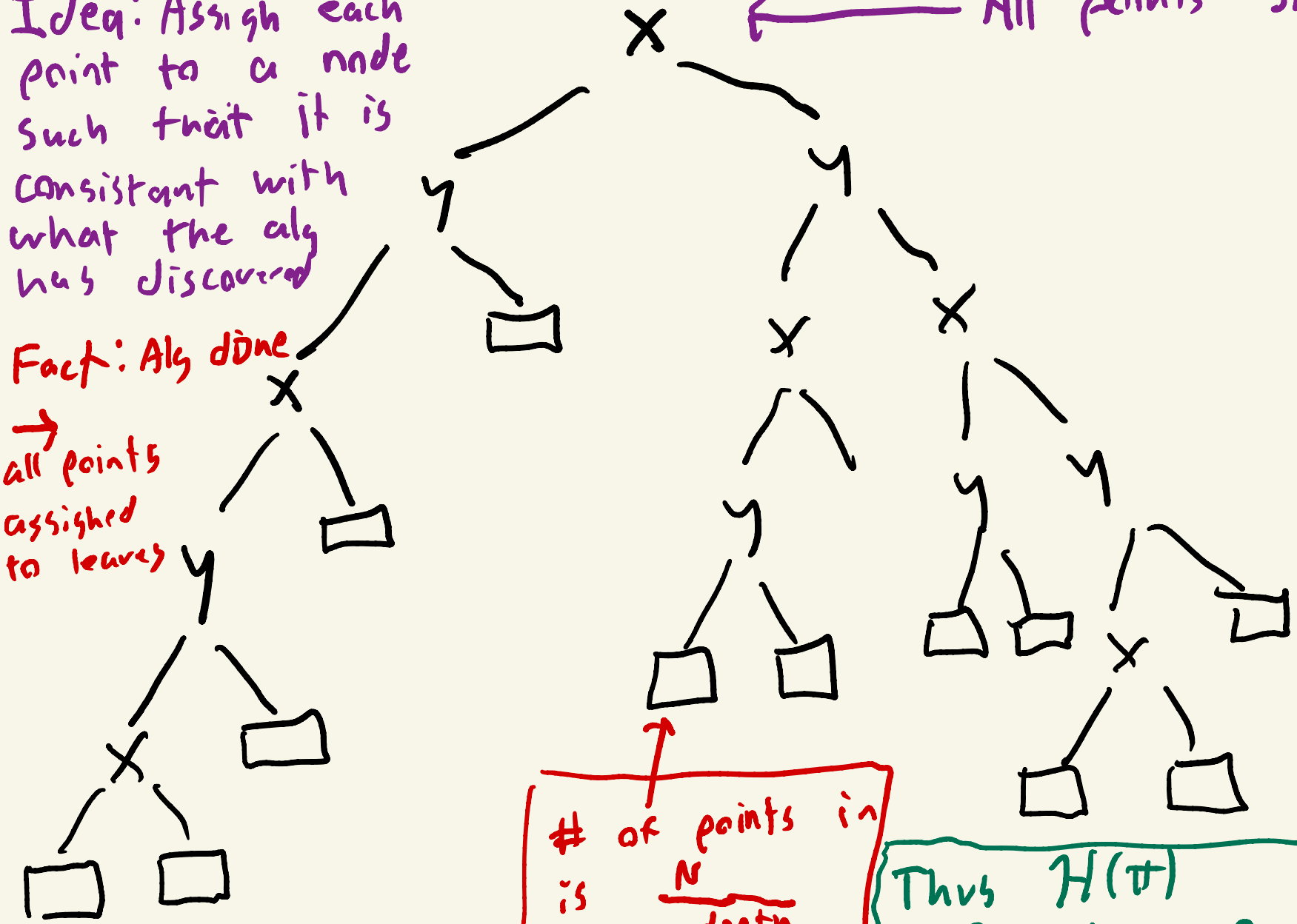
# of points in is $\frac{N}{2^{depth}}$

Thus $H(T)$ is Σ points in $R \cdot$ depth R

Tree View of K-D tree

Idea: Assign each point to a node such that it is consistant with what the alg has discovered

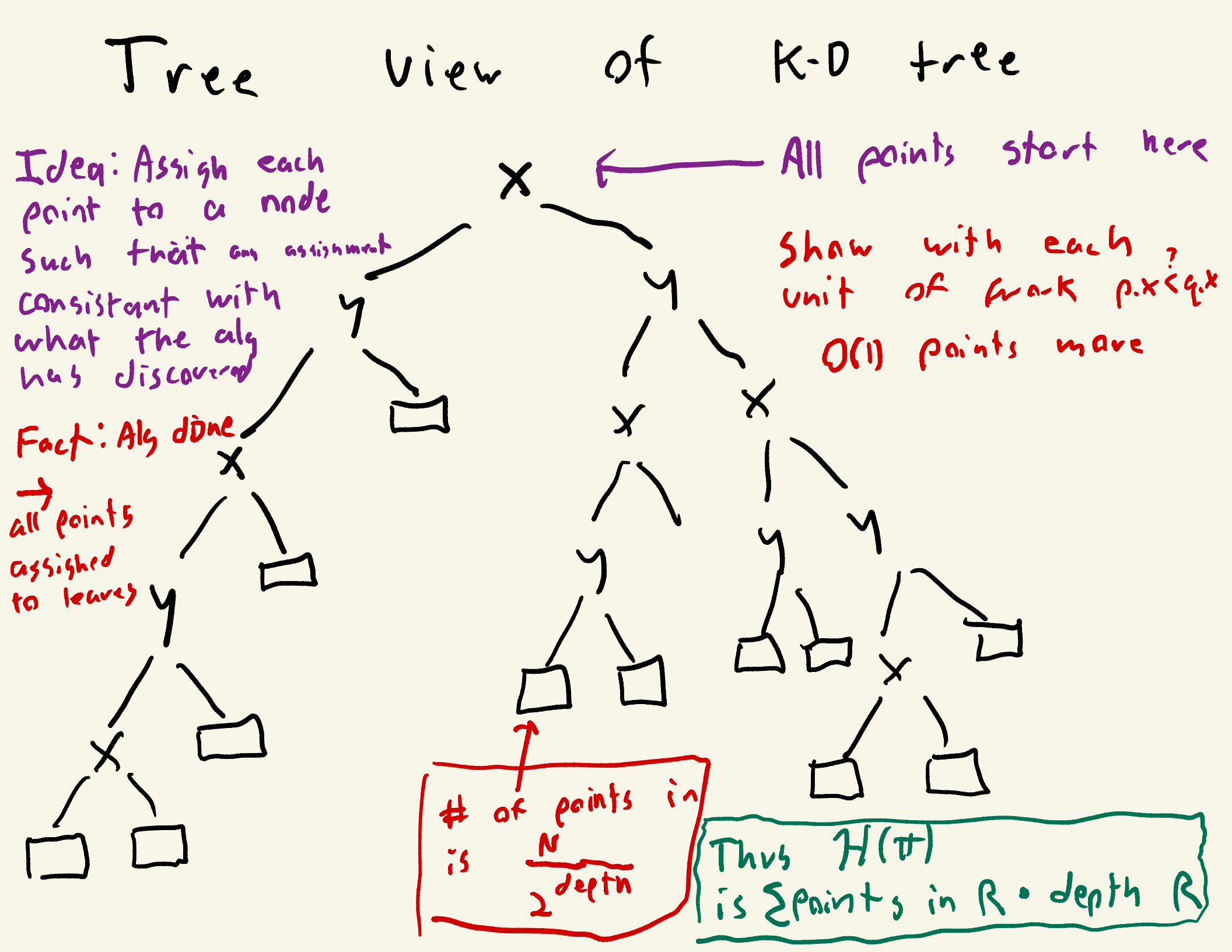
Fact: Alg done → all points assigned to leaves



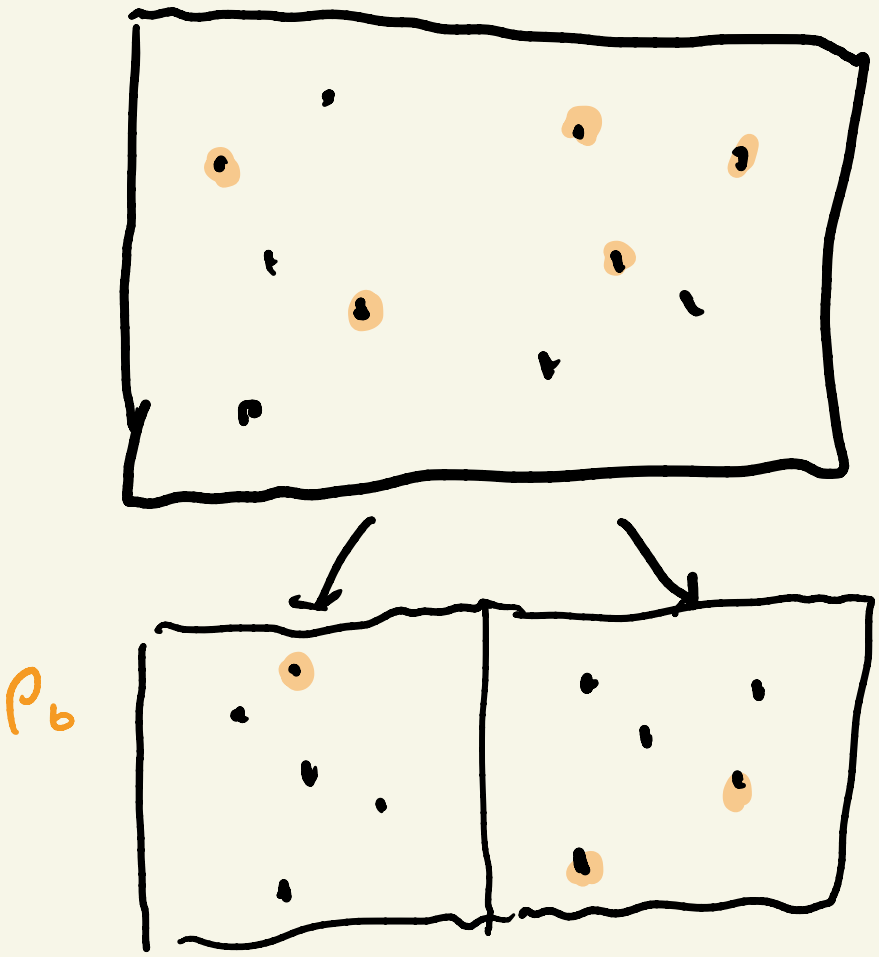
# of points in is $\frac{N}{2^{depth}}$

Thus $H(\pi)$ is $\sum$ points in R · depth R

Tree View of K-D tree



Basic Idea



Affiliated with this rectangle

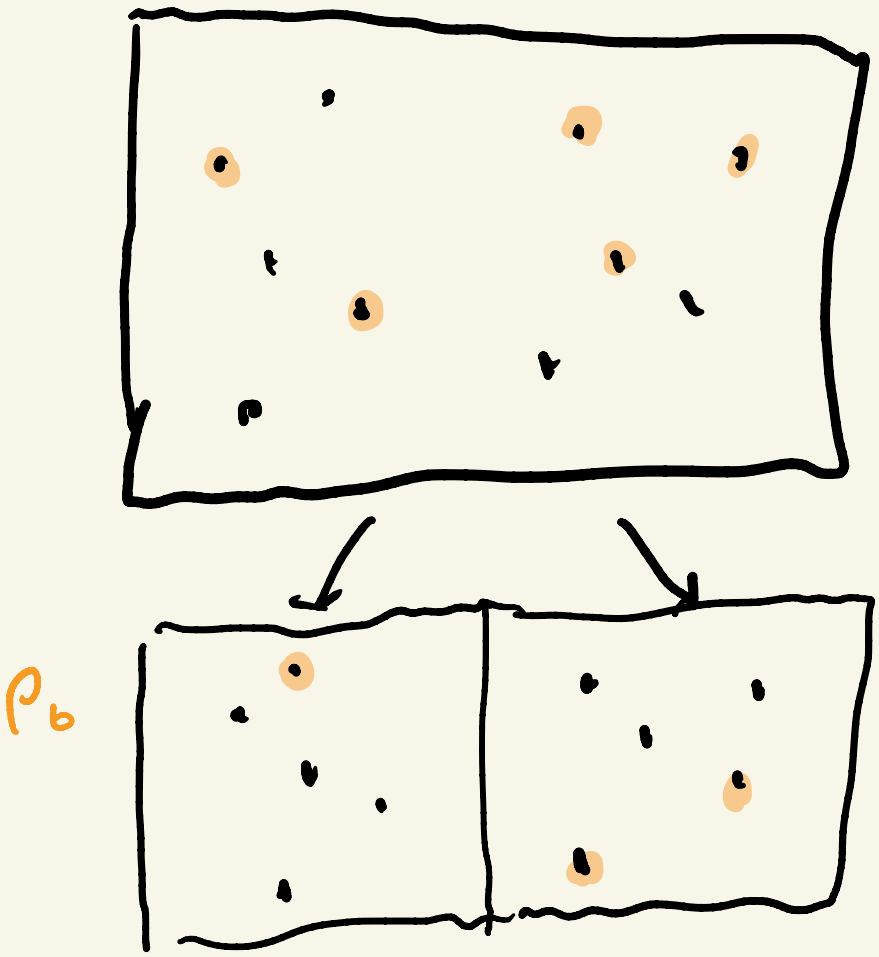
p_1 p_2 p_3 p_4 p_5

← could map to orange in any order

p_6

p_7 p_8

Basic Idea



Affiliated with this rectangle

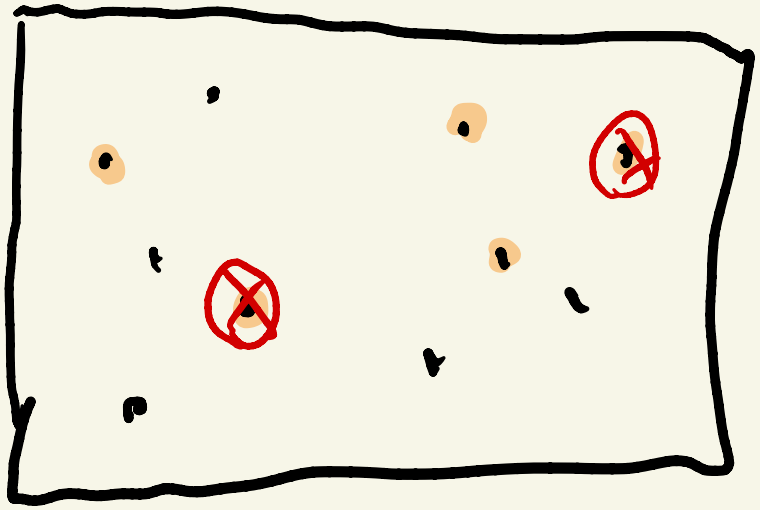
p_1 p_2 p_3 p_4 p_5

← could map to orange in any order

Algorithm compares

$p_1.x \overset{?}{\leq} p_2.x$

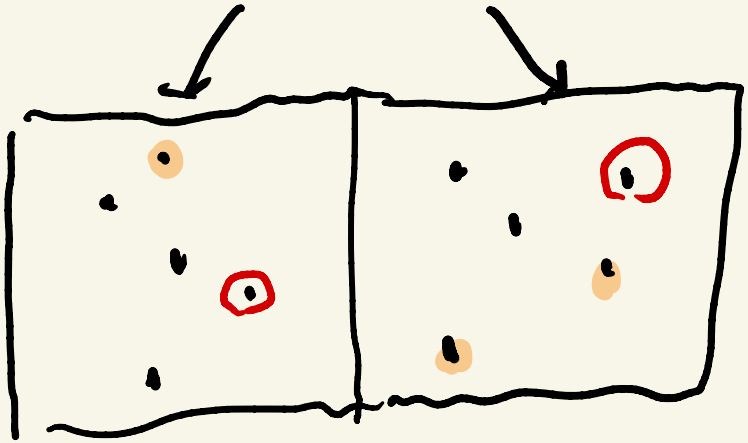
Basic Idea



Affiliated with this rectangle

p_2 p_3 p_4 p_5

← could map to orange in any order



p_1 p_6

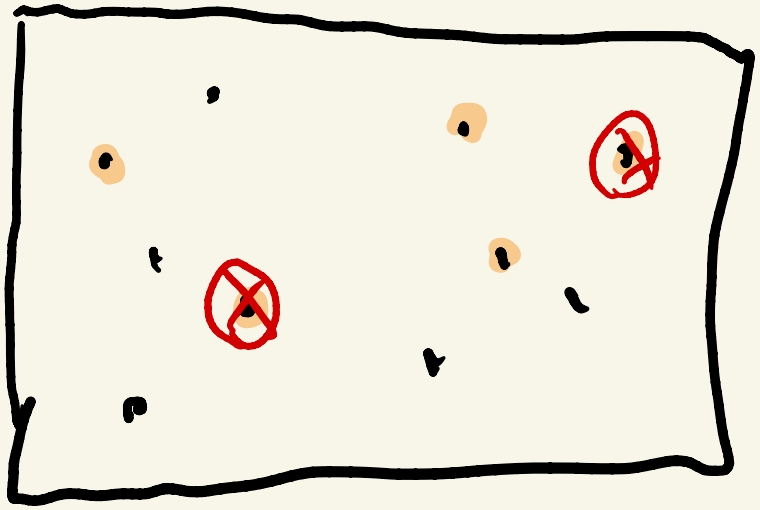
p_7 p_8 p_1

report $p_1.x < p_2.x$

Algorithm compares

$p_1.x \overset{?}{\leq} p_2.x$

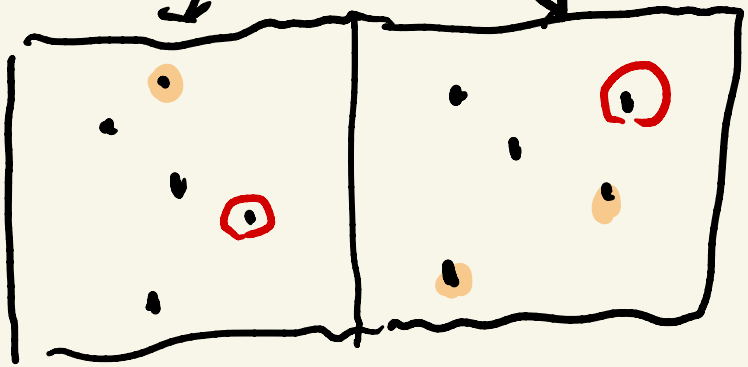
Basic Idea



Affiliated with this rectangle

p_2 p_3 p_4 p_5

← could map to orange in any order



p_1 p_6

p_7 p_8 p_1

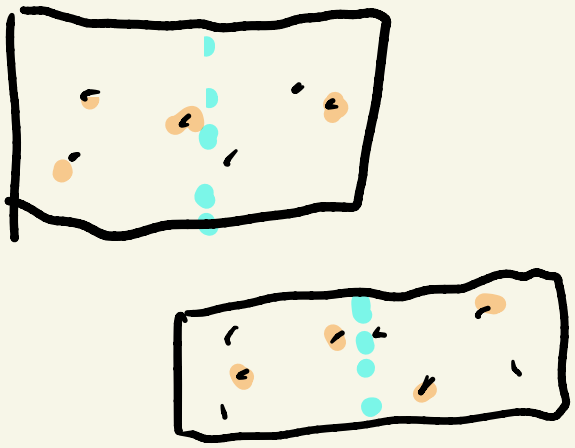
report $p_1.x < p_2.x$

This moves 2 points down 1 in Tree

Algorithm compares

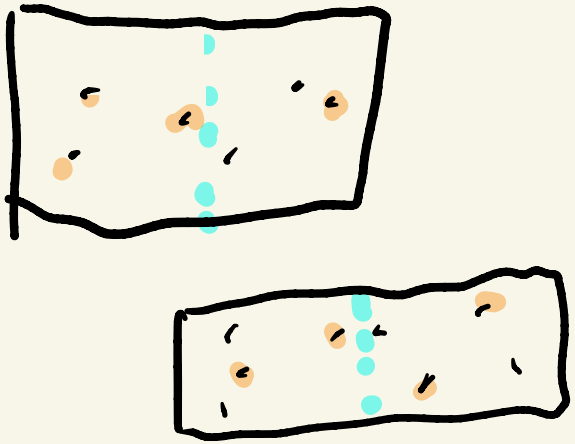
$p_1.x \overset{?}{\leq} p_2.x$

Other Cases



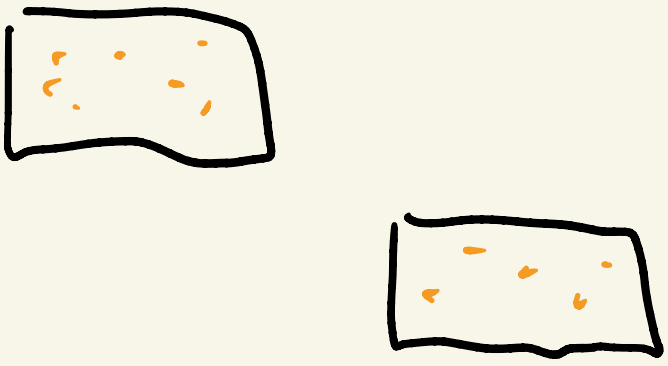
Suppose you compare x coordinate of points with overlapping x ranges

Other Cases

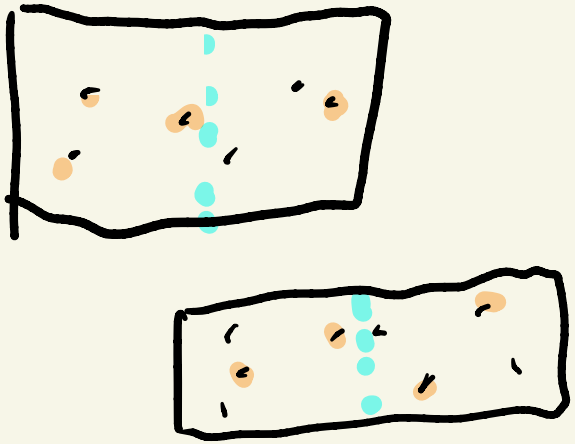


Suppose you compare x coordinate of points with overlapping x ranges

If No overlap

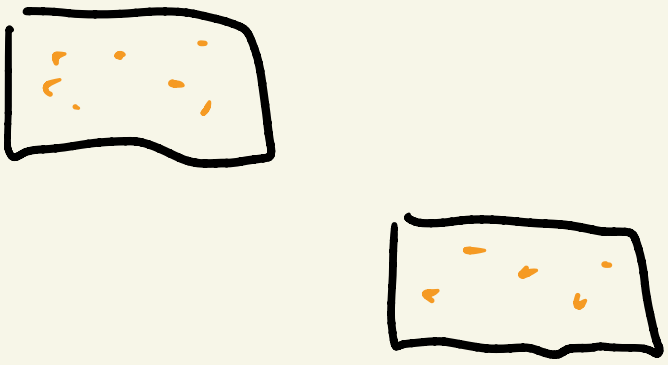


Other Cases

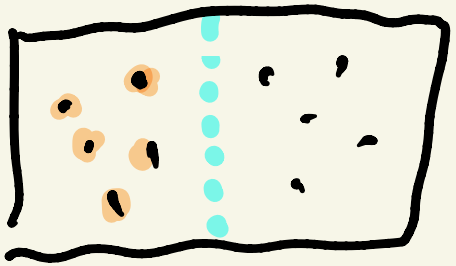


Suppose you compare x coordinate of points with overlapping x ranges

If No overlap



If all remaining are on one side



Push down (Amortize)

2D Dominance - Summary

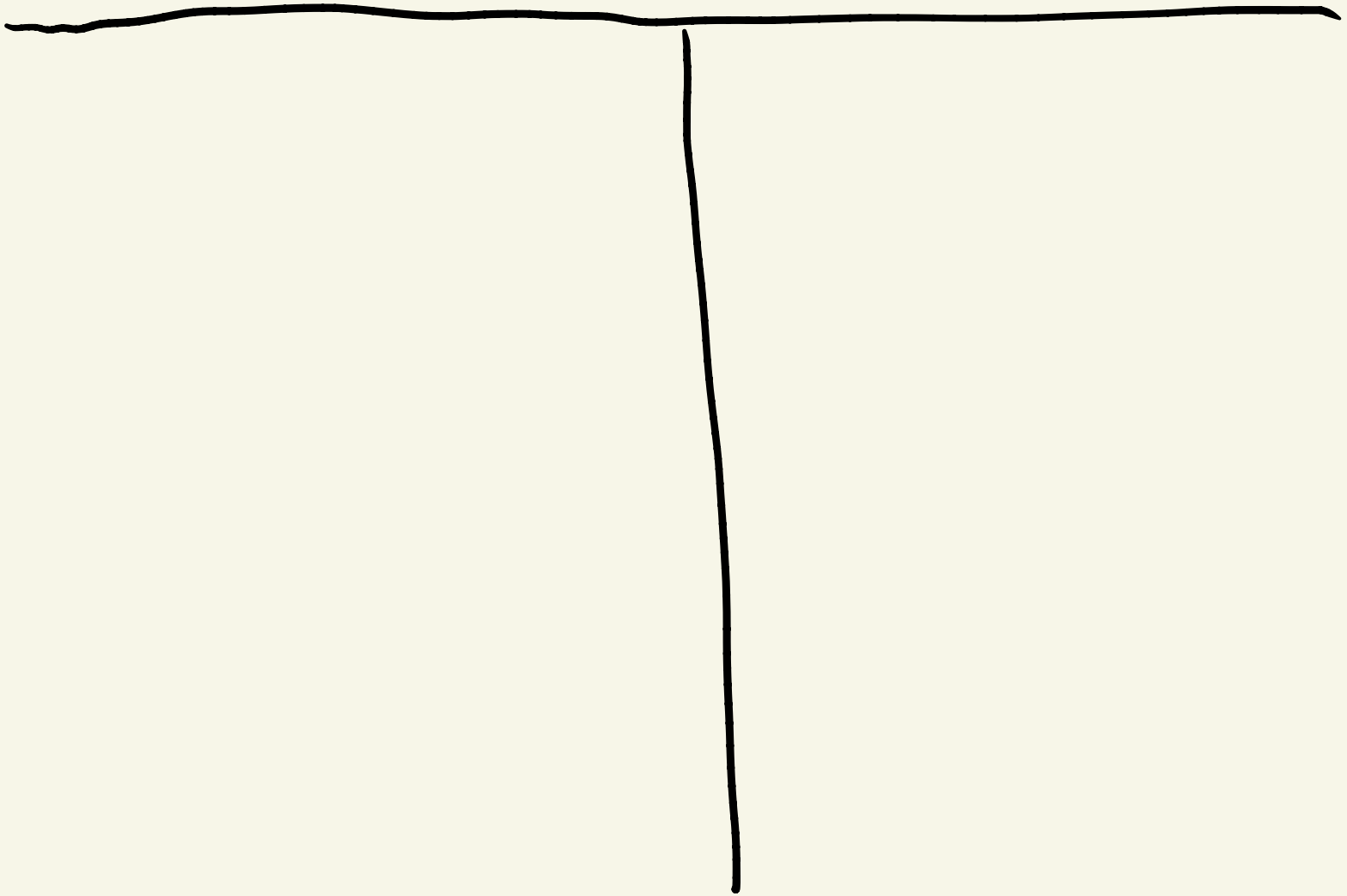
- Simple algorithm
- Structural entropy gives upper and lower bound
- Instance based optimality:
  Best possible alg on an instance among all permutations of the input

Instance-Based

Other

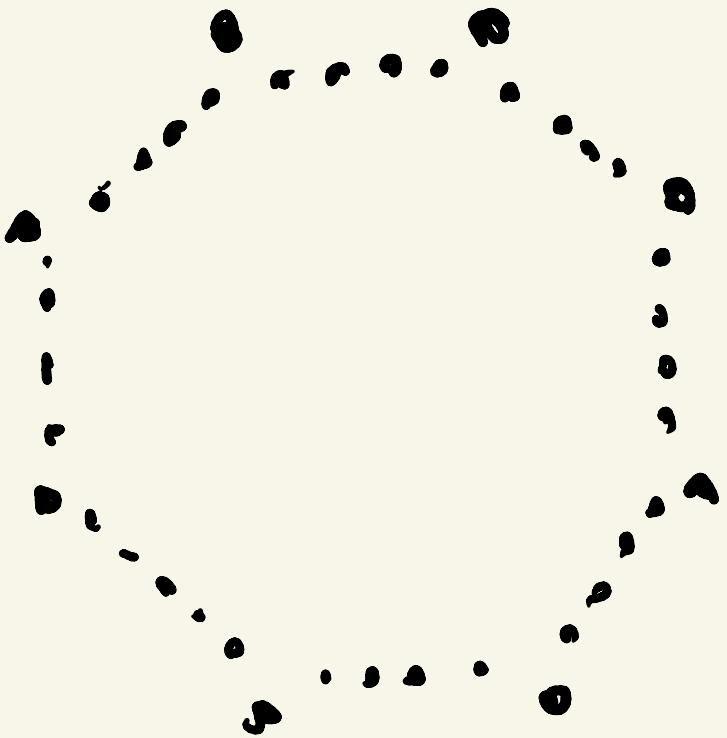
Optimality

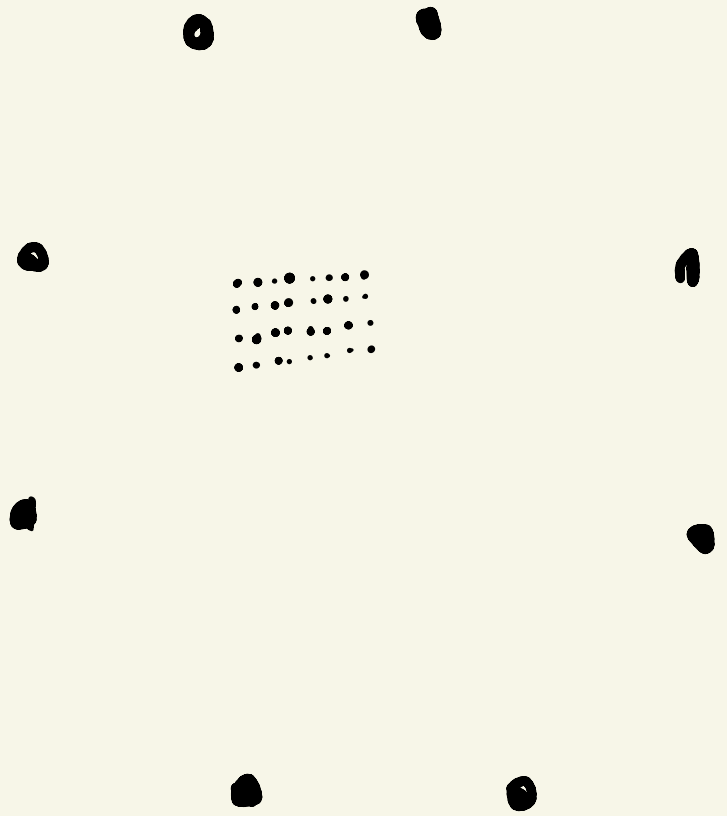
problems



Instance-Based Optimality

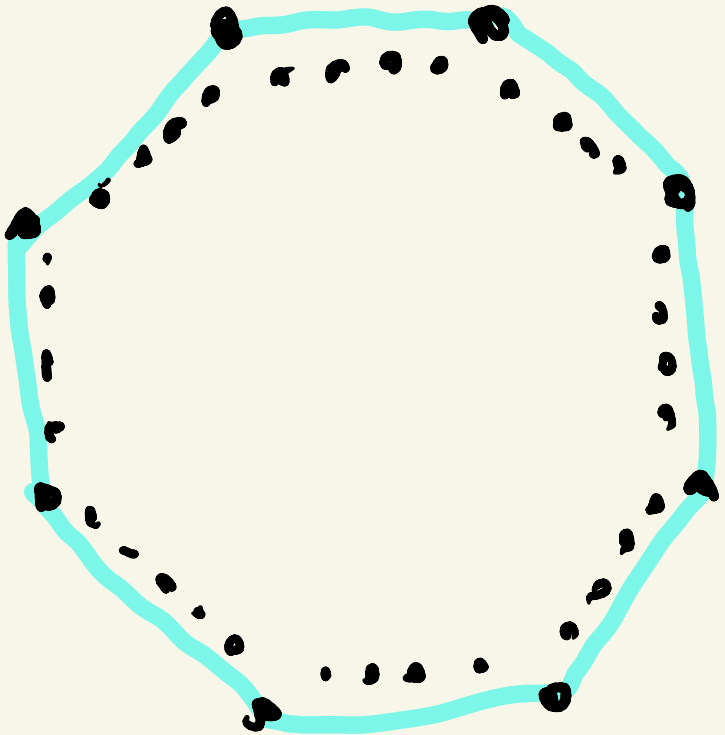
Other problems

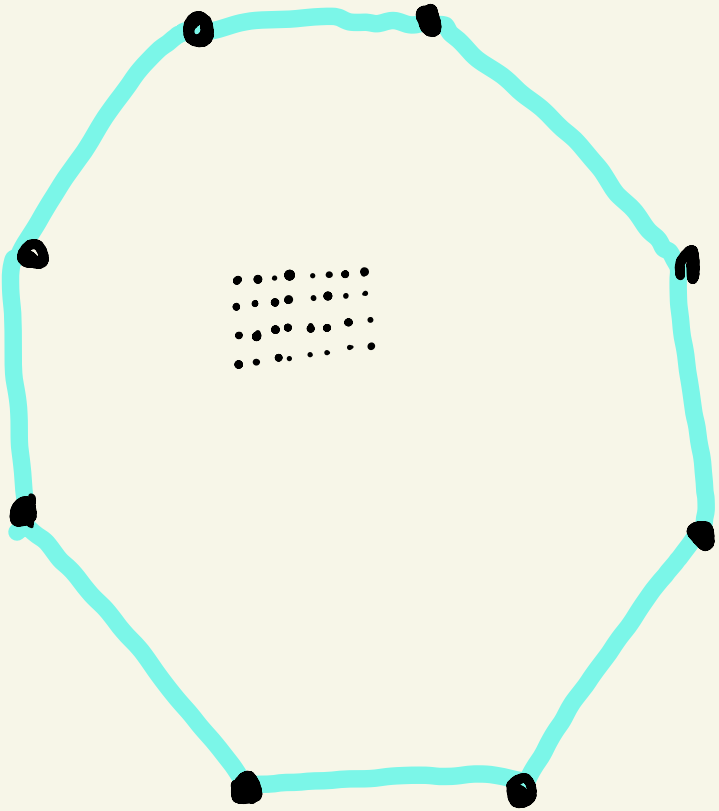




Instance-Based Optimality

Other problems





Same K

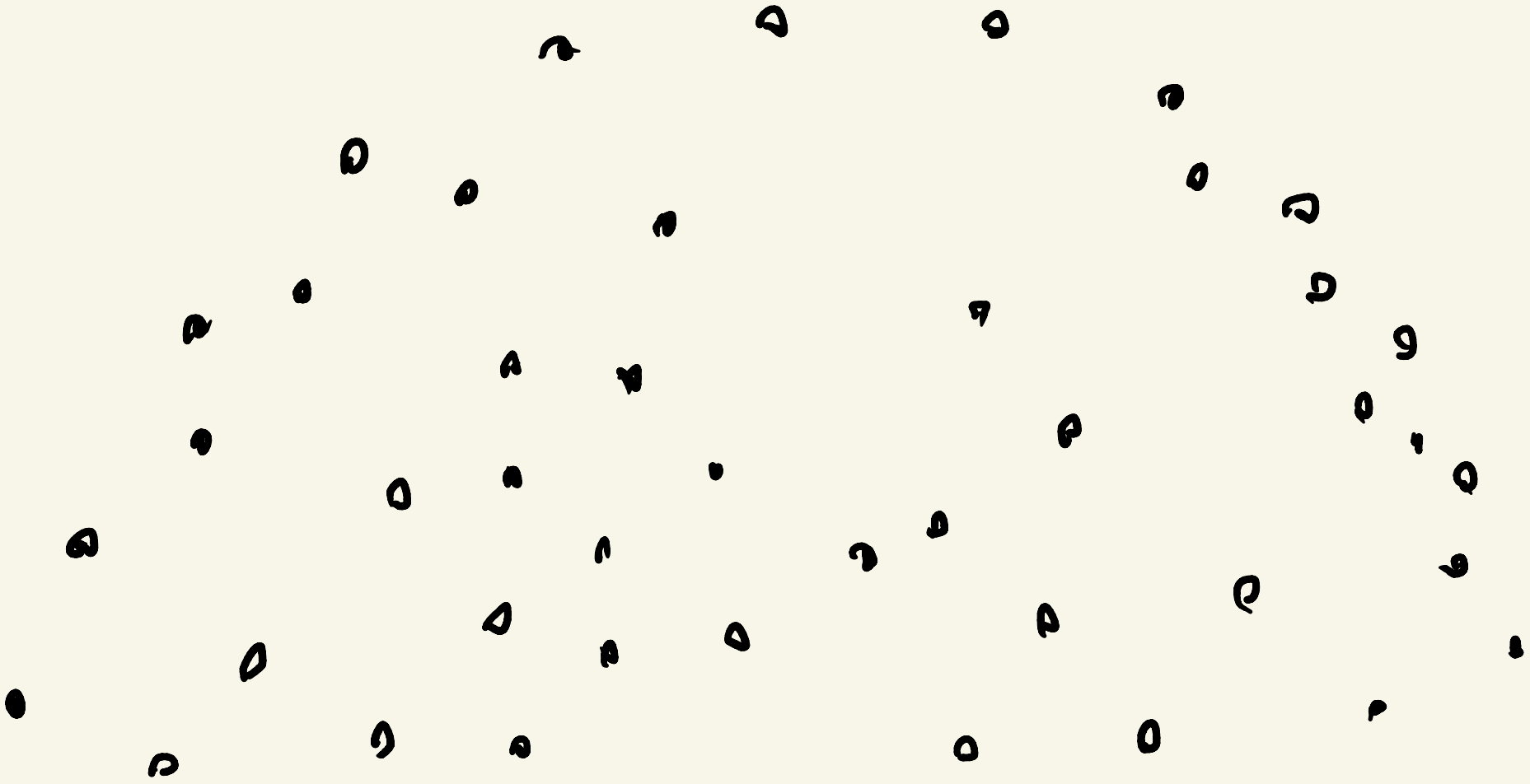
Hard

Easy

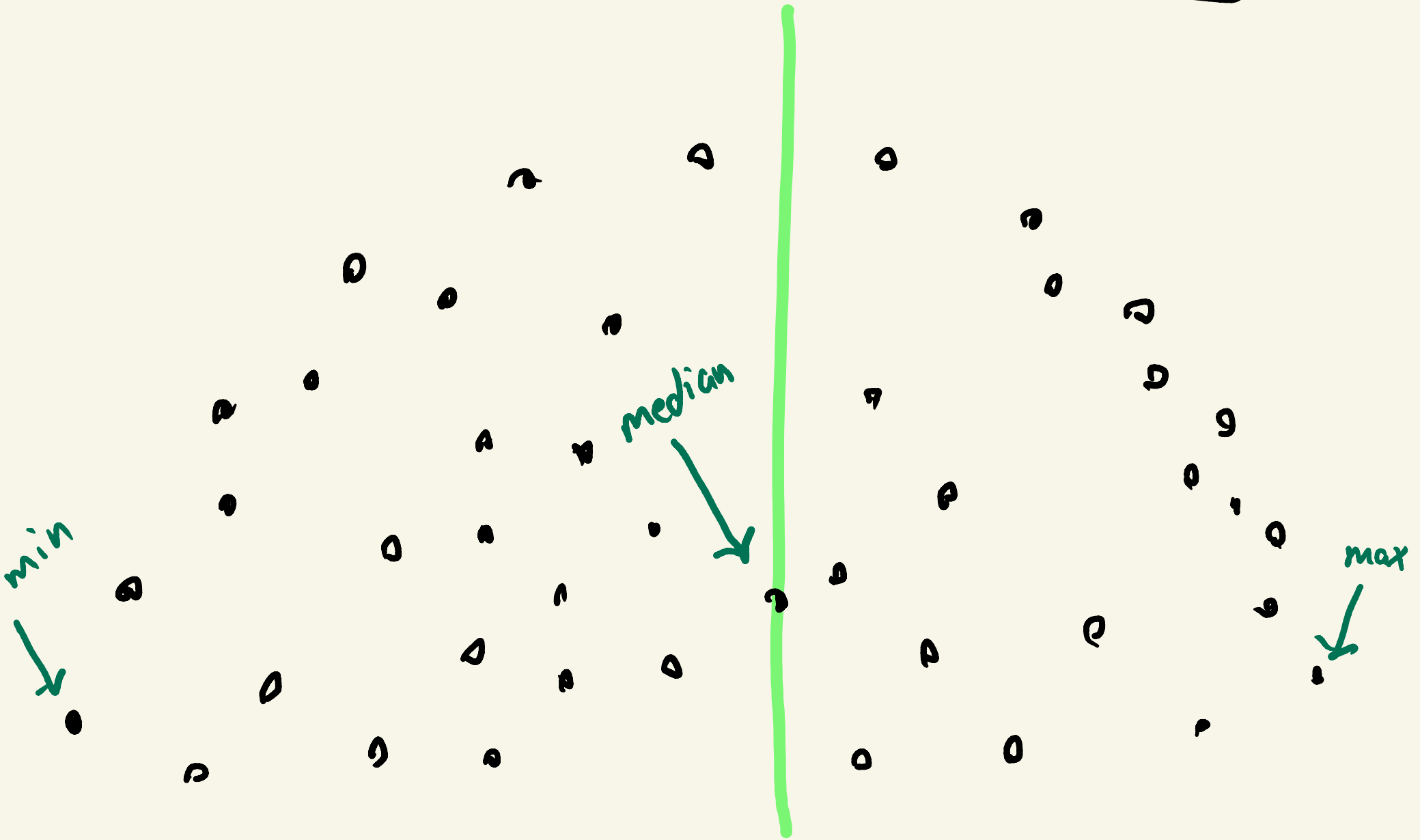
Convex Hull

- Need bound
- Need algorithm
- Need proof of instance-based optimality.

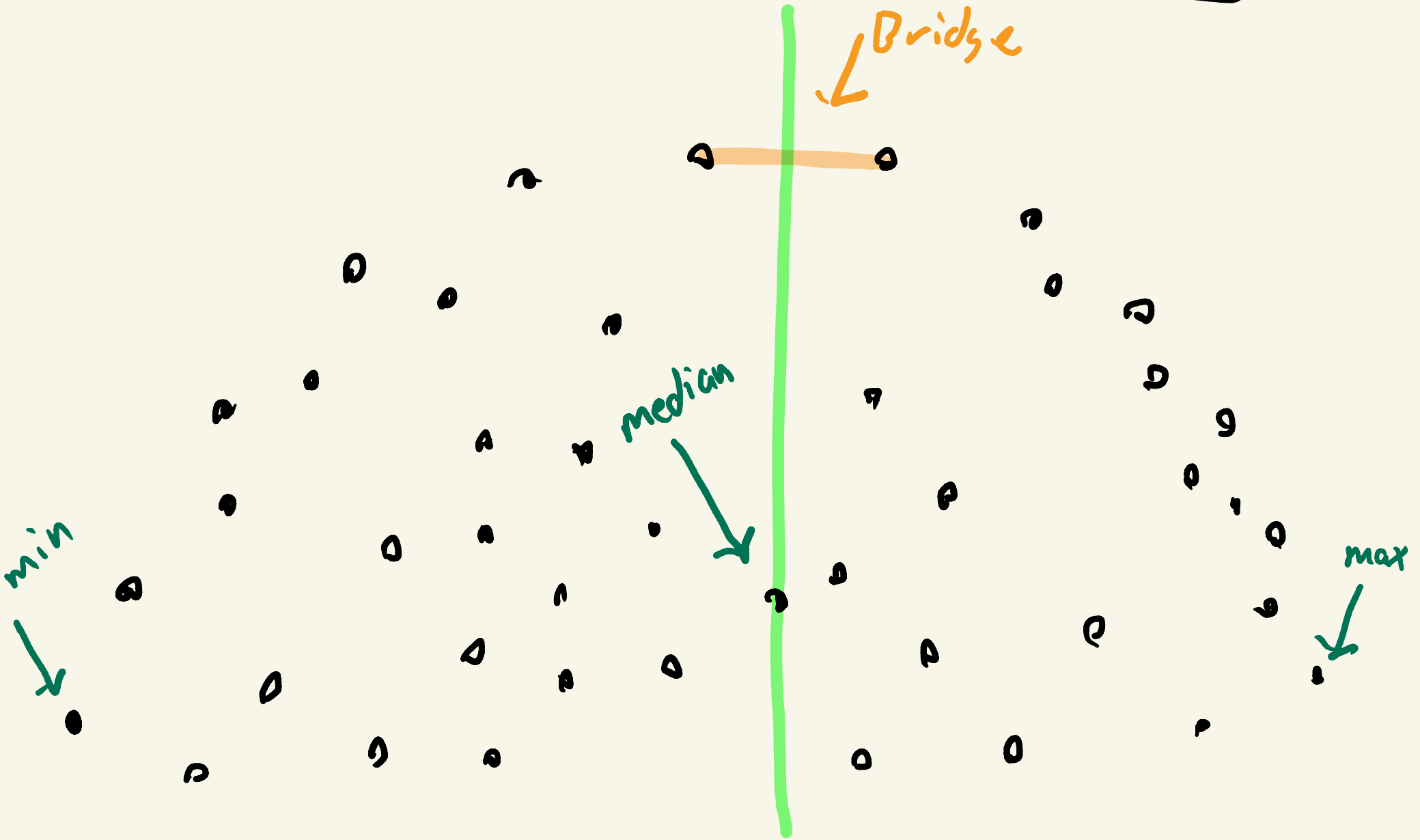
An Instance-Based Optimal Algorithm for upper Hull



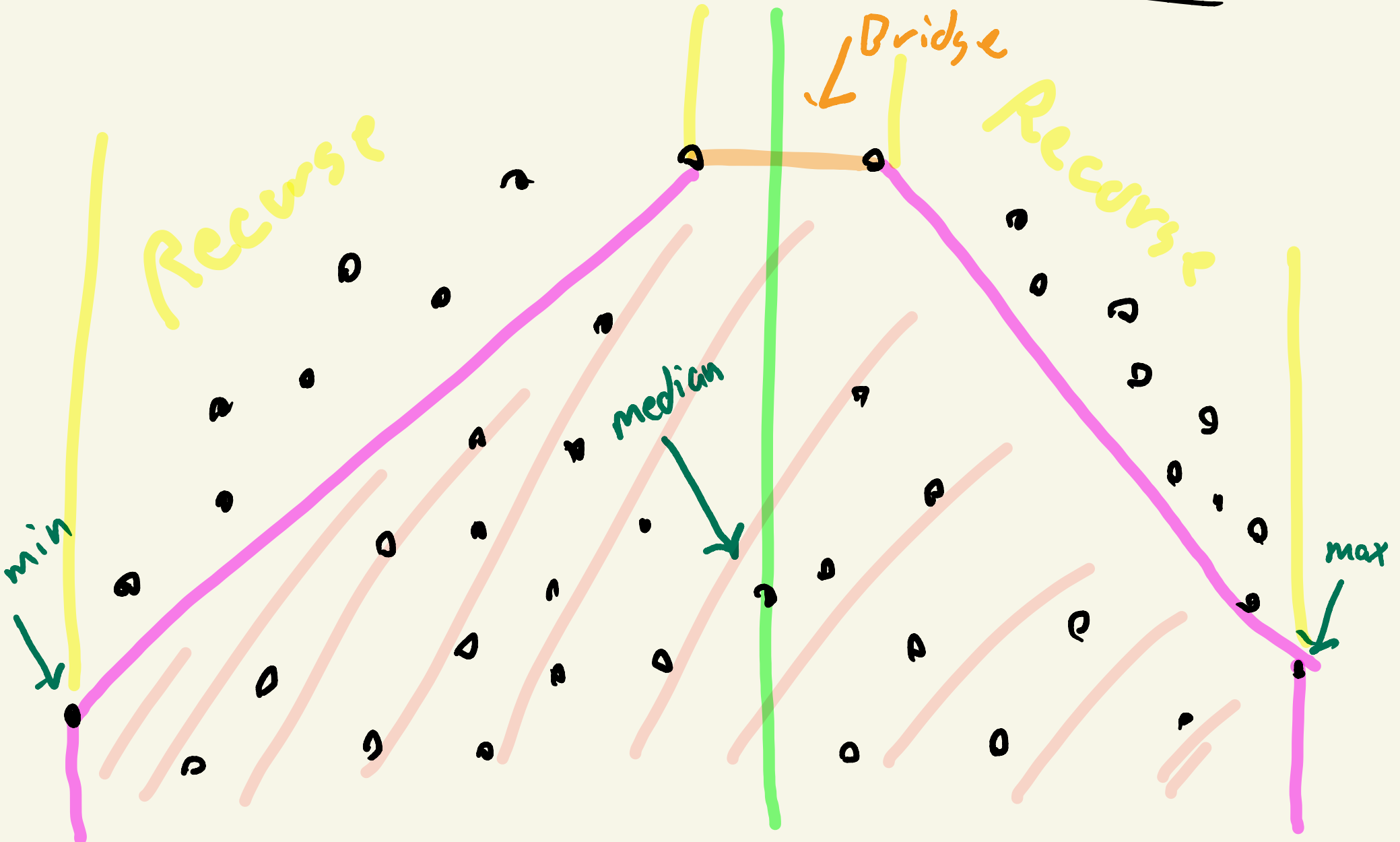
An Instance-Based Optimal Algorithm for upper Hull



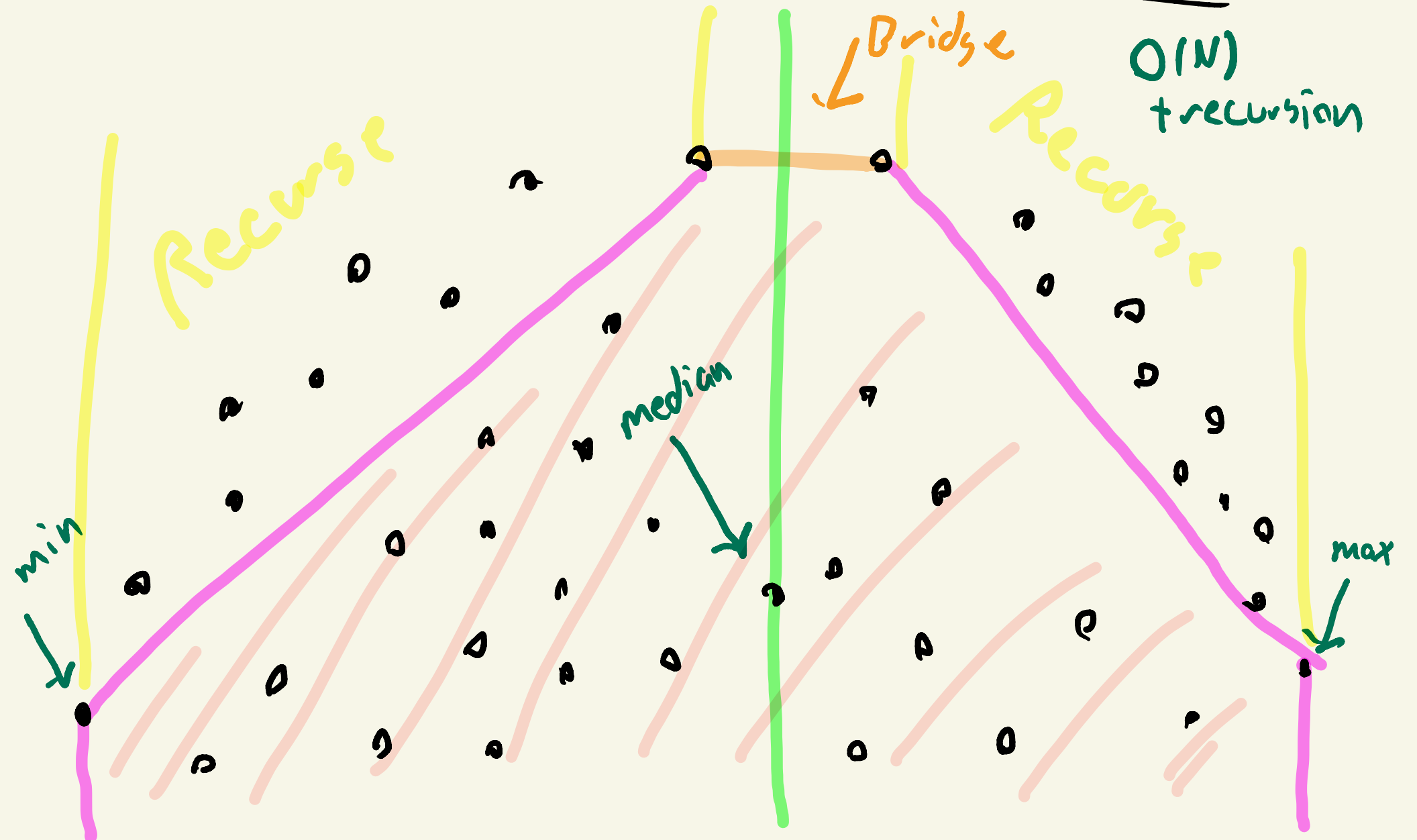
An Instance-Based Optimal Algorithm for upper Hull



An Instance-Based optimal Algorithm for upper Hull



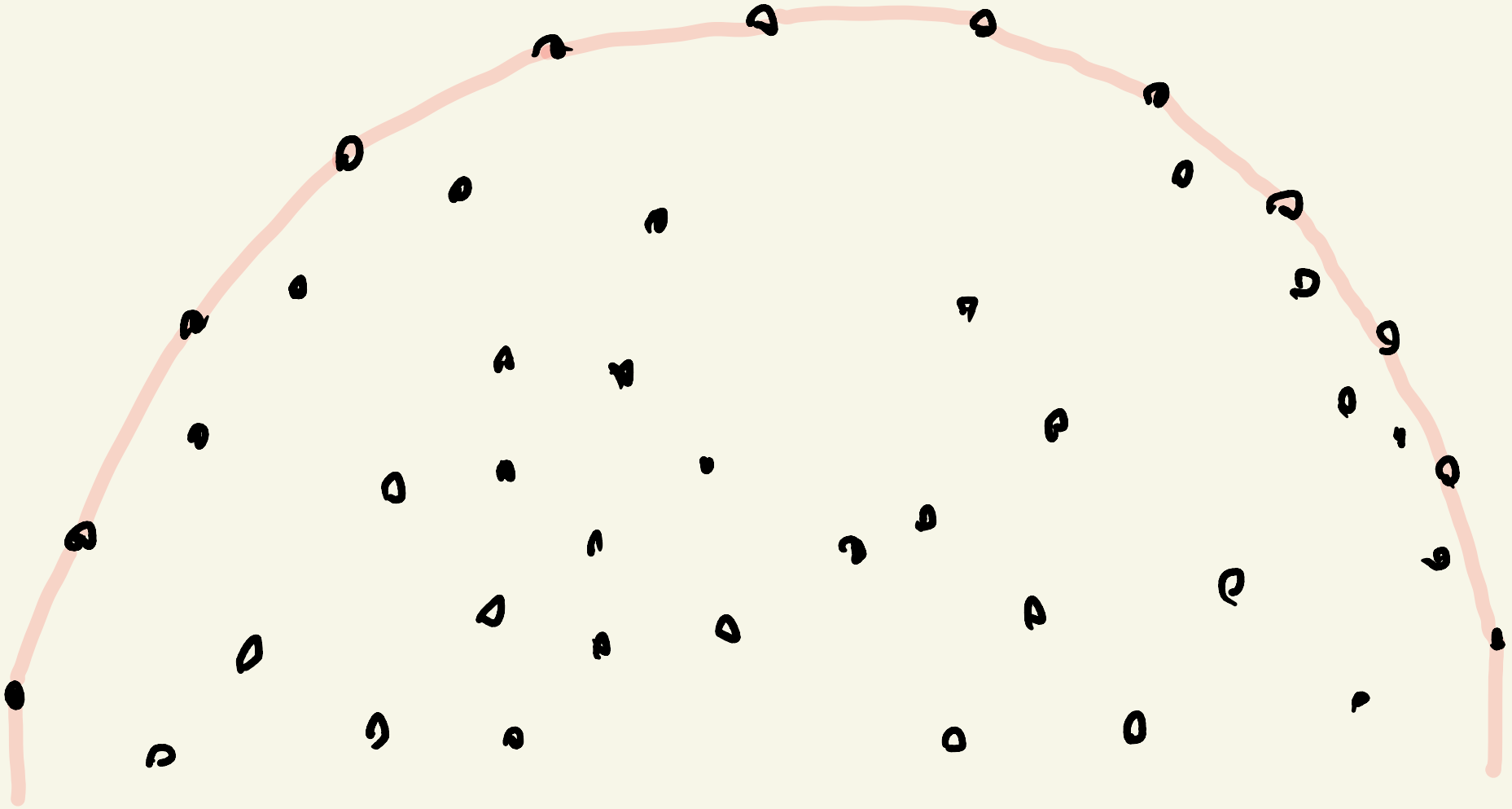
An Instance-Based Optimal Algorithm for upper Hull



Band

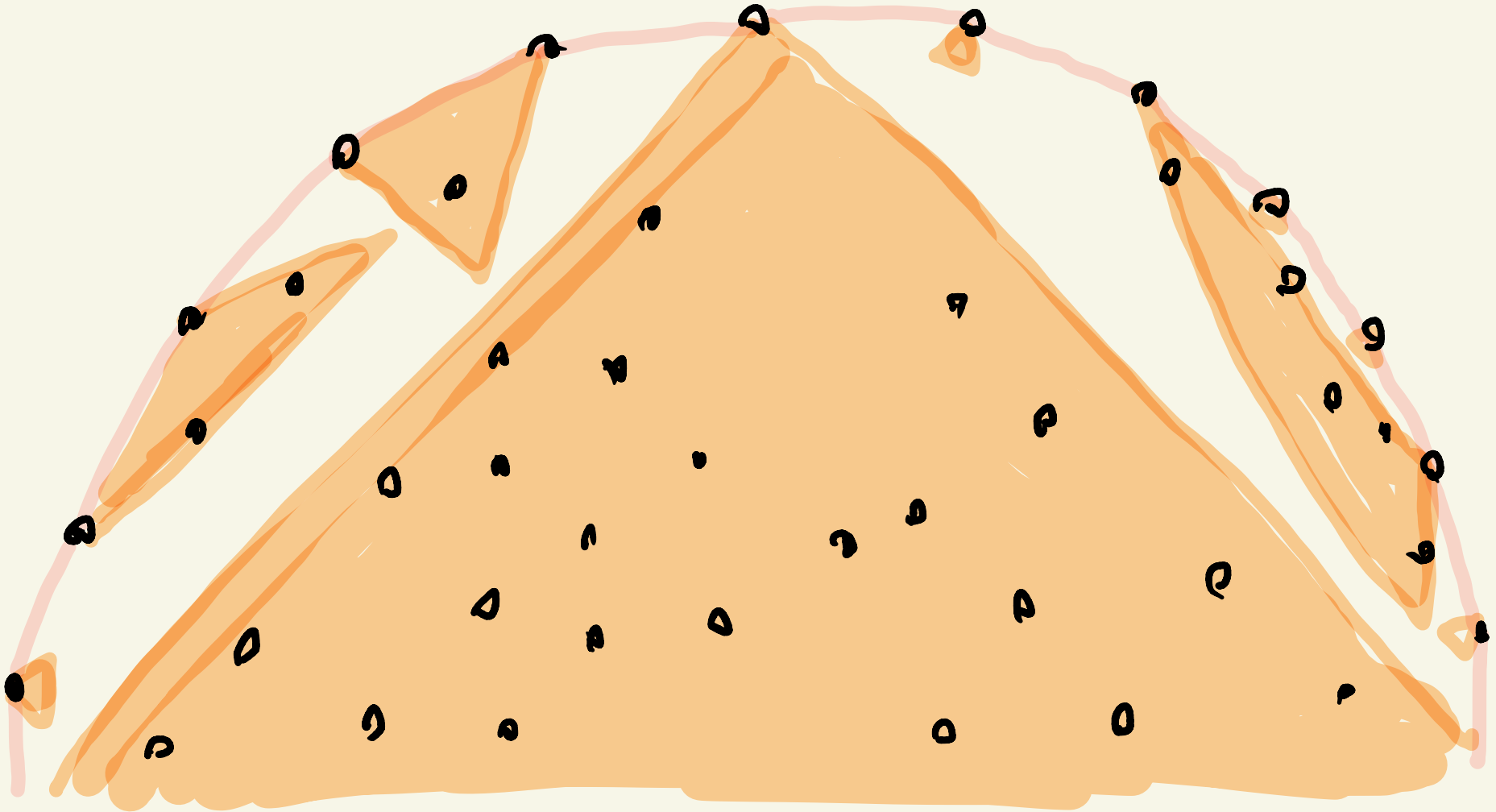
Bound

Cover points with triangles $T_1, T_2 \ldots$
That don't go outside of
the convex hull to maximize $\sum |T_i| \log \frac{N}{|T_i|}$



Bound

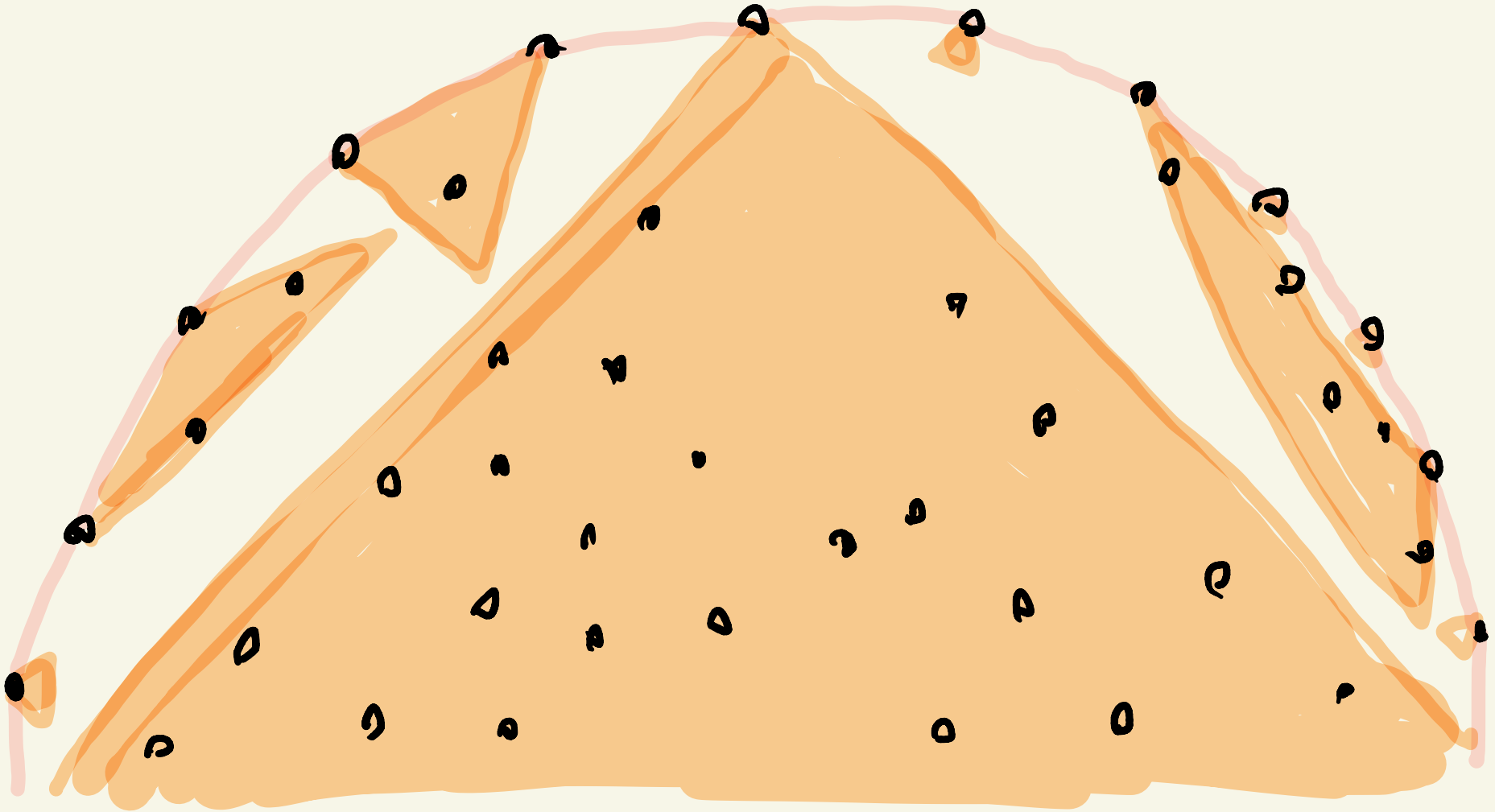
Cover points with triangles $T_1, T_2 \ldots$
That don't go outside of
the convex hull to minimize $\sum |T_i| \log \frac{N}{|T_i|}$



Bound

$O(n \log k)$

Cover points with triangles $T_1, T_2 \ldots$
That don't go outside of
the convex hull to minimize $\sum |T_i| \log \frac{N}{|T_i|}$



2D Convex hull

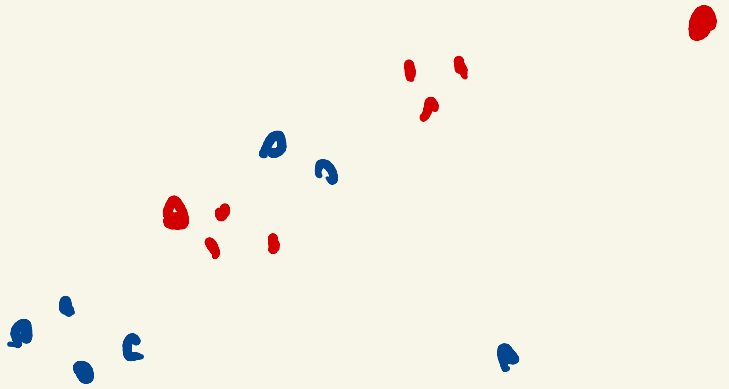
- Proving Upper Bound similar to 2D dominance
- k-D tree of lower bound does not
  → Uses partition thm of [Matoušek 92]

3D Convex hull

- Median Idea no longer works
- Use partitioning also, along with LP

Red/Blue Reporting

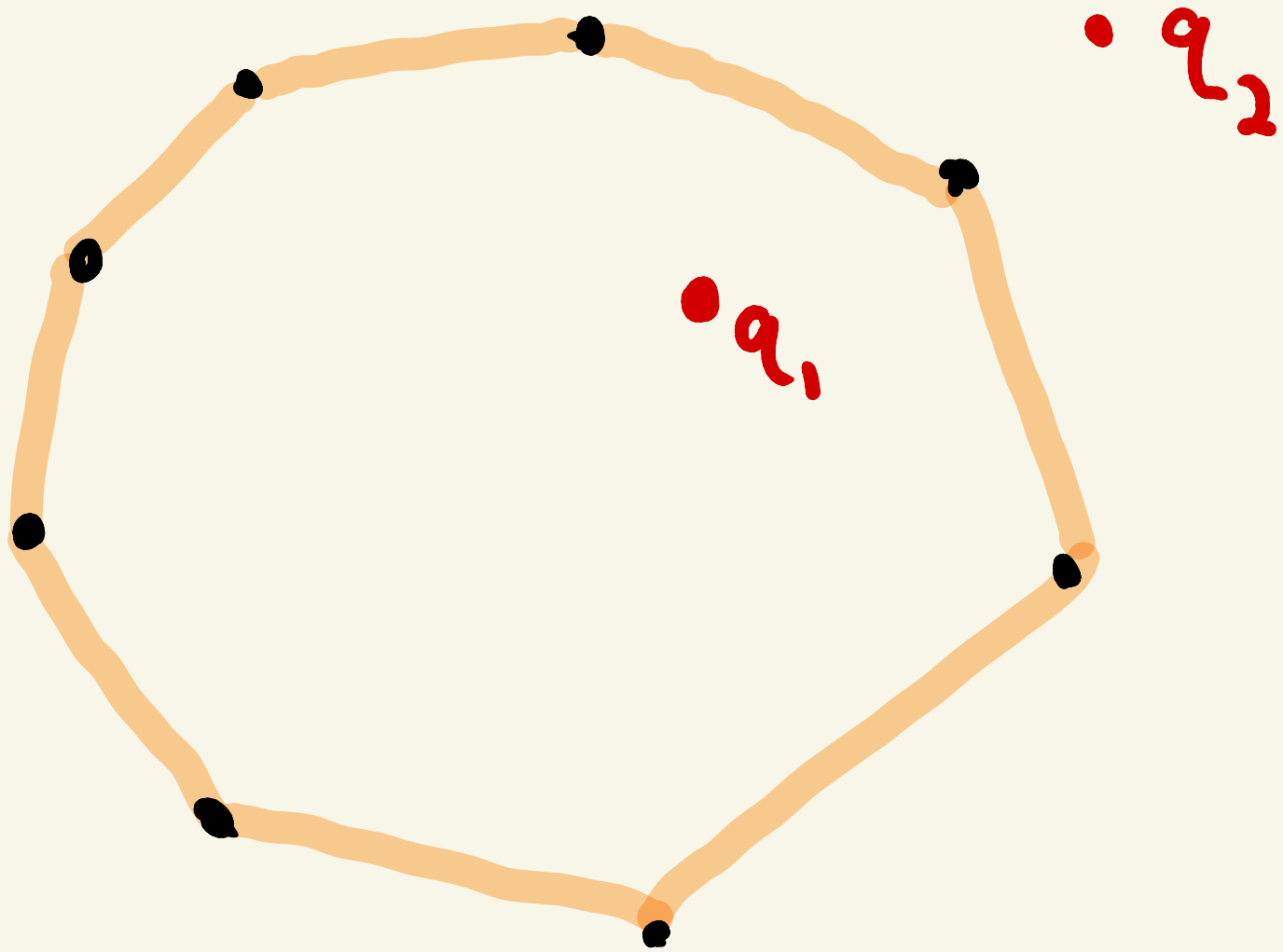
- Similar



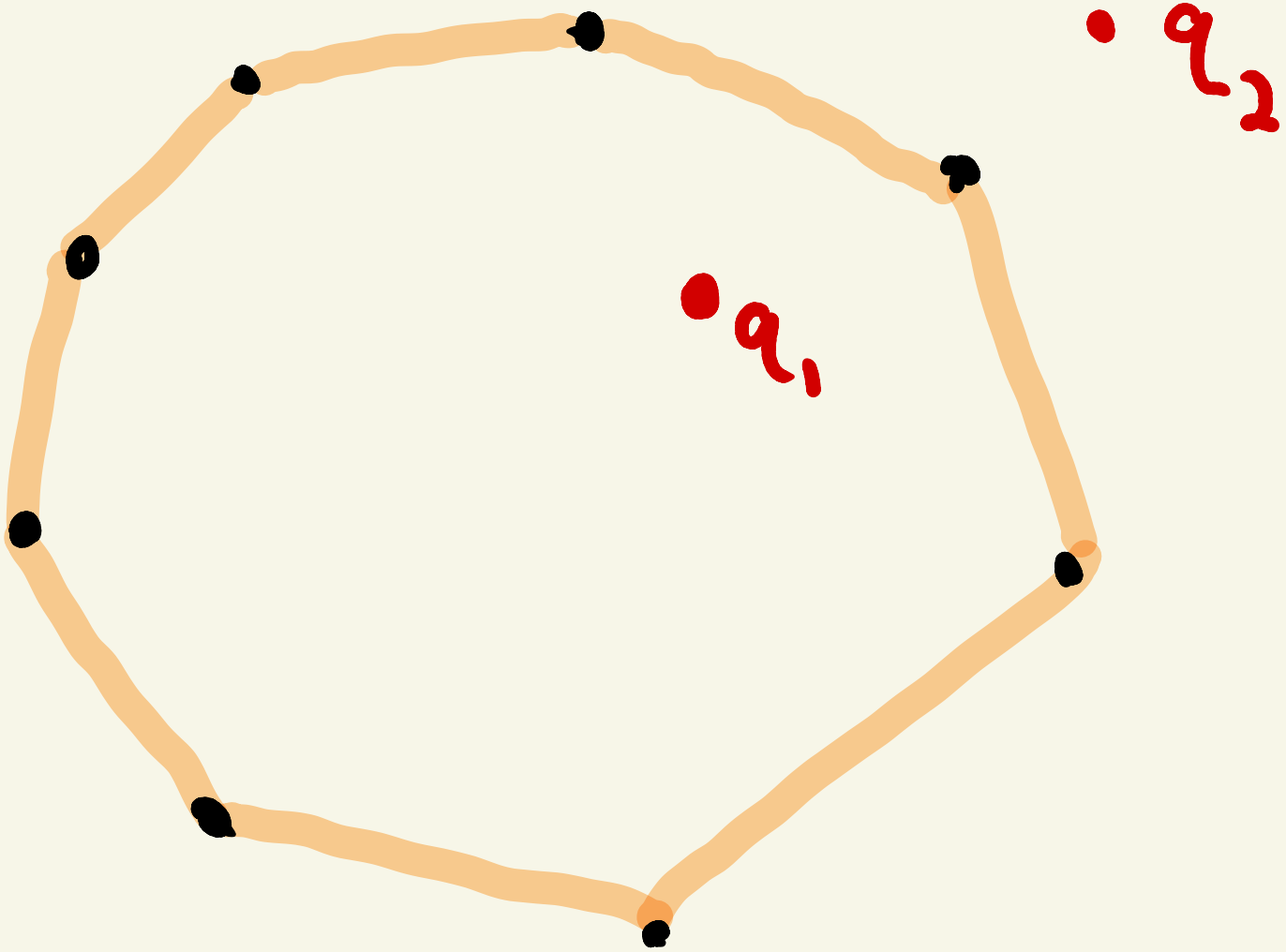
Part 4

Optimality through Intelligence

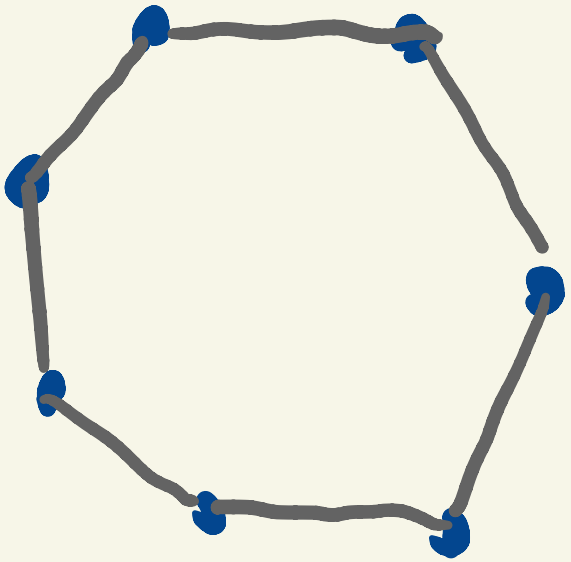
Point in Polygon



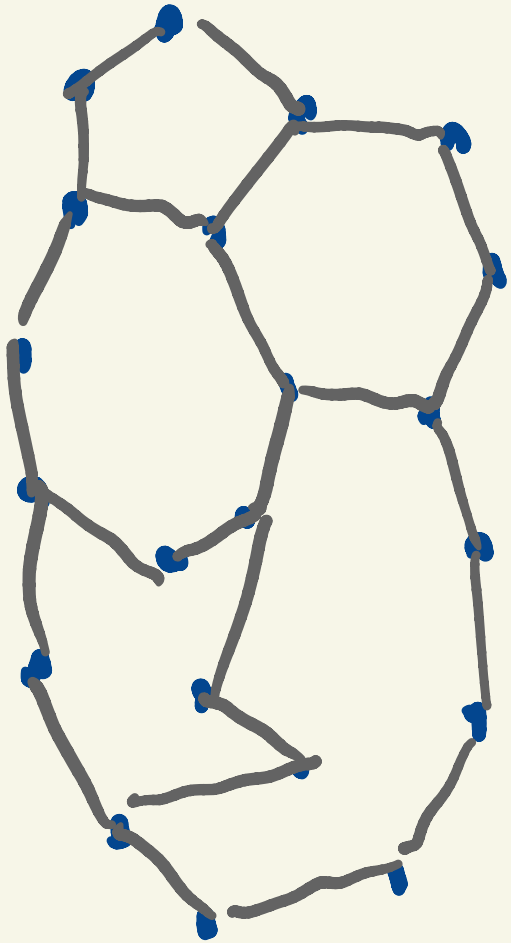
Point in Polygon

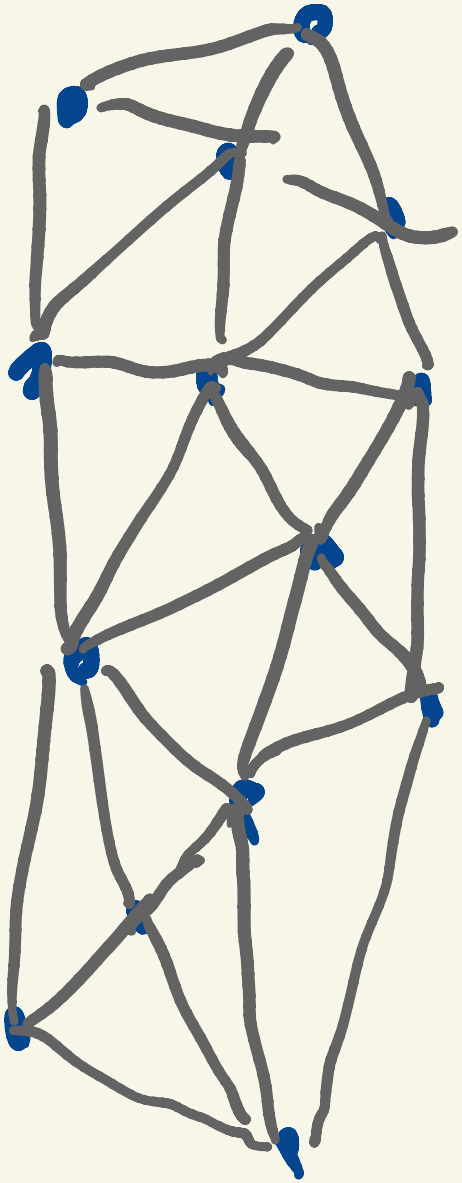


Suppose You Knew the distribution of q



Only two regions!
Thus constant entropy.

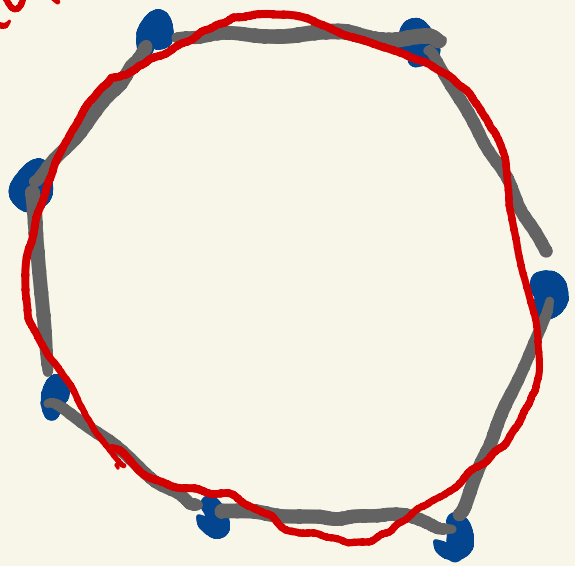




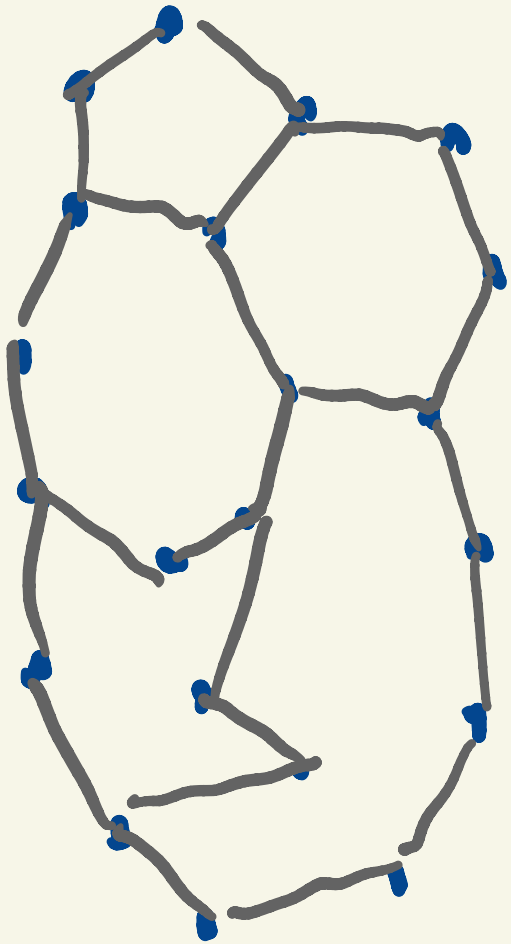
Entropy

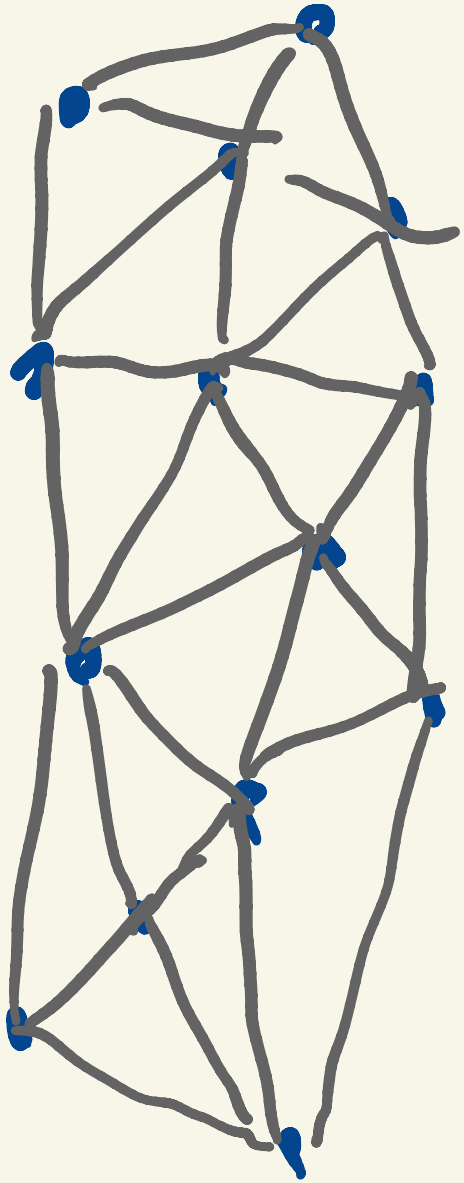
$$\Theta\left(\sum p(T_i)\log\frac{1}{p(T_i)}\right)$$

If searches are equiprobable on the circle, $\Theta(\log N)$ if needed



Only two regions! Thus constant entropy.

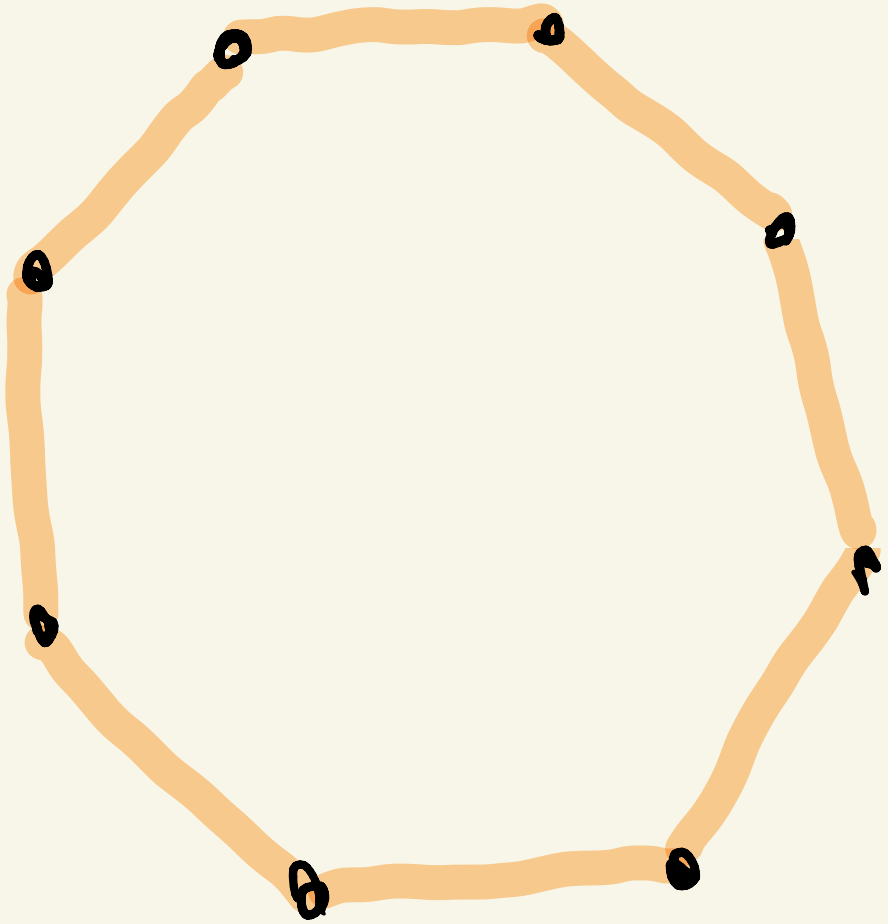




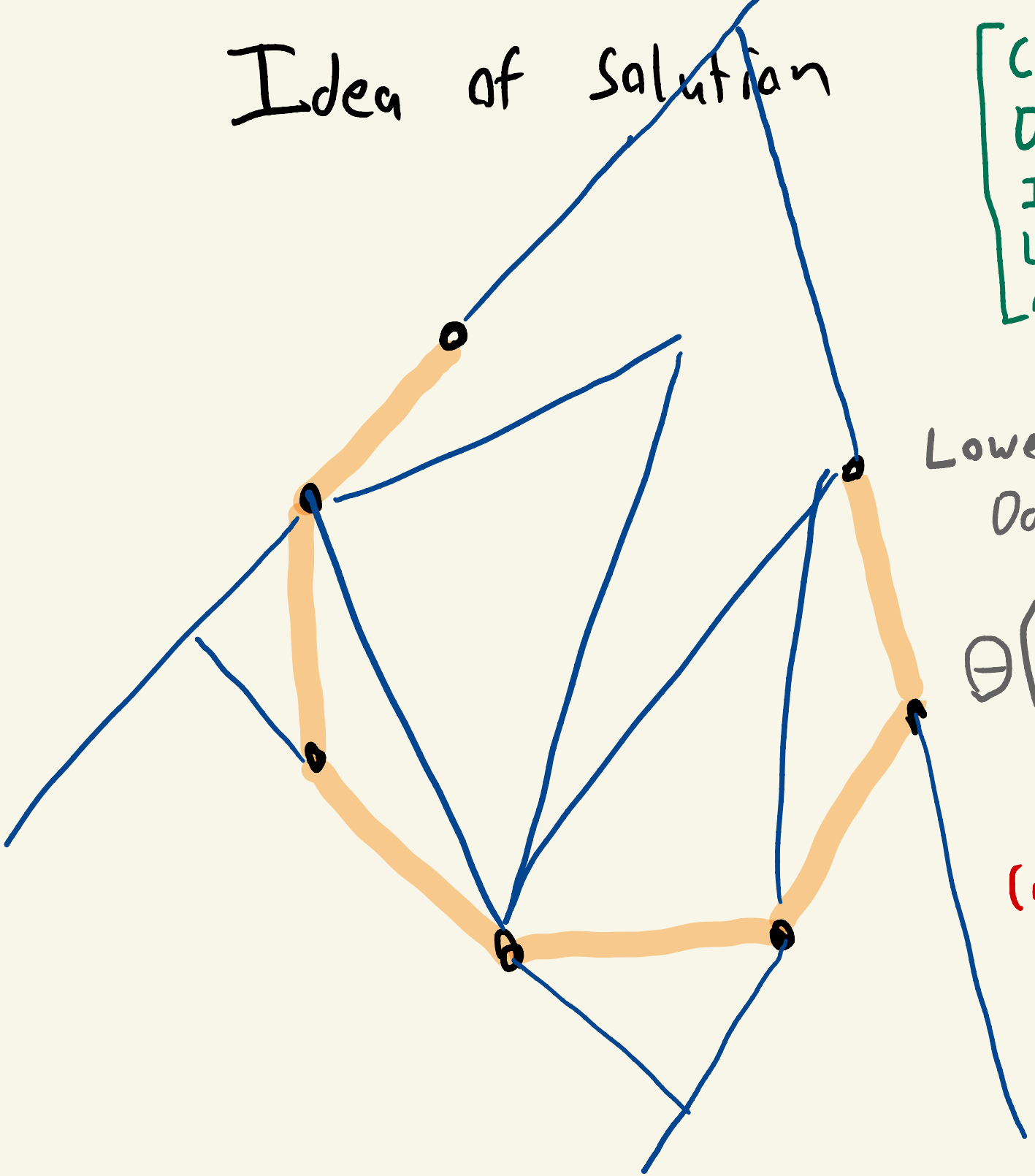
Entropy

$$\Theta\left[\sum p(T_i)\log_{\frac{1}{p(T_i)}}\right]$$

Cole He
Dujmović
Icaza
Langerman
Morin 08



Idea of Solution



[Cole He
Dusmovic
Iacono
Langerman
Marin 08]

Lower Bound t
Data structure

Θ(minimum entropy triangular subdivision

(remember, probability distribution is given)

But what if you dont know the probability distribution?

But what if you dont know the probability distribution?

For Triangulations

Remember this slide?

Result [I 01 Sadc] [I04]

Given a known distribution, PPL on ind queries takes $O(H) = O(\sum p_i \log \frac{1}{p_i})$

But wait, what if I dont know the distribution?

Is OK! Just periodically rebuild with the observed frequencies.

[I Mulzer 12] [I 11 SoCG]

But what if you dont know the probability distribution?

For Triangulations

Remember this slide?

Not clear what to do. Observed frequencies not so simple

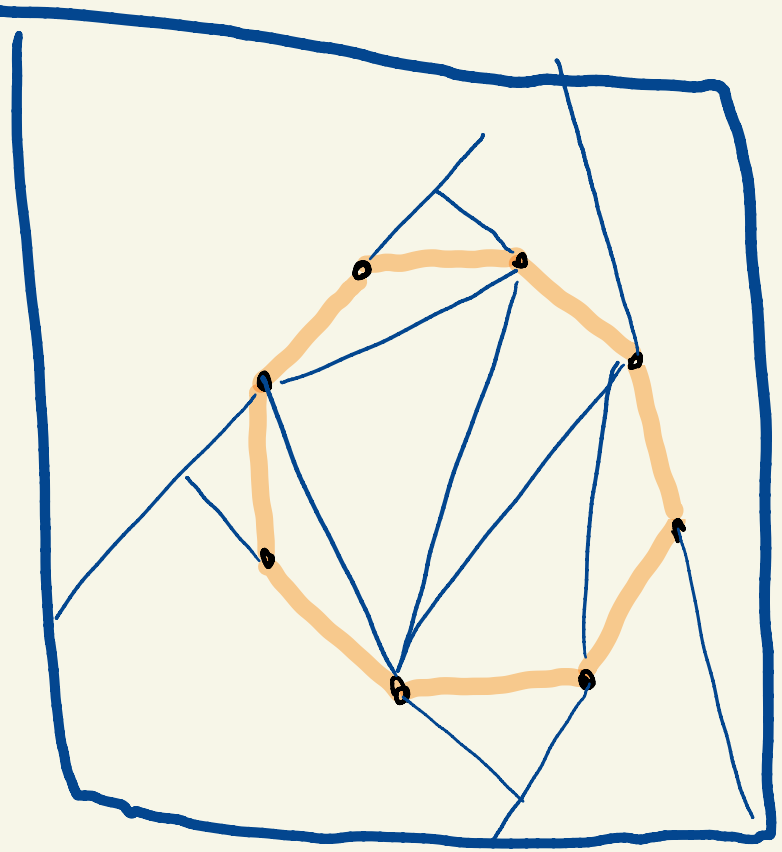
Result [I 01 Sadeo] [I04]

Given a known distribution, PPL on ind queries takes $O(H) = O(\sum p_i \log \frac{1}{p_i})$

But wait, what if I dont know the distribution?

Is OK! Just periodically rebuild with the observed frequencies.

[I Mulzer 12] [I 11 SoCG]



Bandits [ACBFS02]

Events

C	v_1	v_2	v_3	..	v_K
e_1	1	5	2	3	7
e_2	8	12	4	1	1
⋮					
e_ℓ					

experts

Bandits [ACBFS02]

Events

C	v_1	v_2	v_3	..	v_K
e_1	1	5	2	3	7
e_2	8	12	4	1	1
⋮					
e_ℓ					

experts

EXP3 algorithm

Input: events (online)

x_1 x_2 x_3 ... x_m

Output: experts (online)

e_1 e_2 e_3

Such that $\sum_{i=1}^{m} C(x_i, e_{i-1}) + f(\ell, k)$

$\leq \min_{e} \sum_{i=1}^{m} C(x_i, e)$

Bandits [ACBFS02]

Events = different queries $\Theta(N^2)$

C	v_1	v_2	v_3	..	v_K
e_1	1	5	2	3	7
e_2	8	12	4	1	1
.					
.					
.					
e_ℓ					

experts

# of different linear decision algs $\leq 2^{n^2}$

EXP3 algorithm

Input: events (online)

x_1 x_2 x_3 .... x_m

Output: experts (online)

e_1 e_2 e_3

Such that $\sum_{i=1}^{m} C(x_i, e_{i-1}) + f(\ell, k)$ (does it matter for large m)

$\leq \min_{e} \sum_{i=1}^{m} C(x_i, e)$ (Best alg)

Summary

Algorithms

- Can use measures other than N
- Can give meaningfull tight bounds assuming data comes in arbitrary order

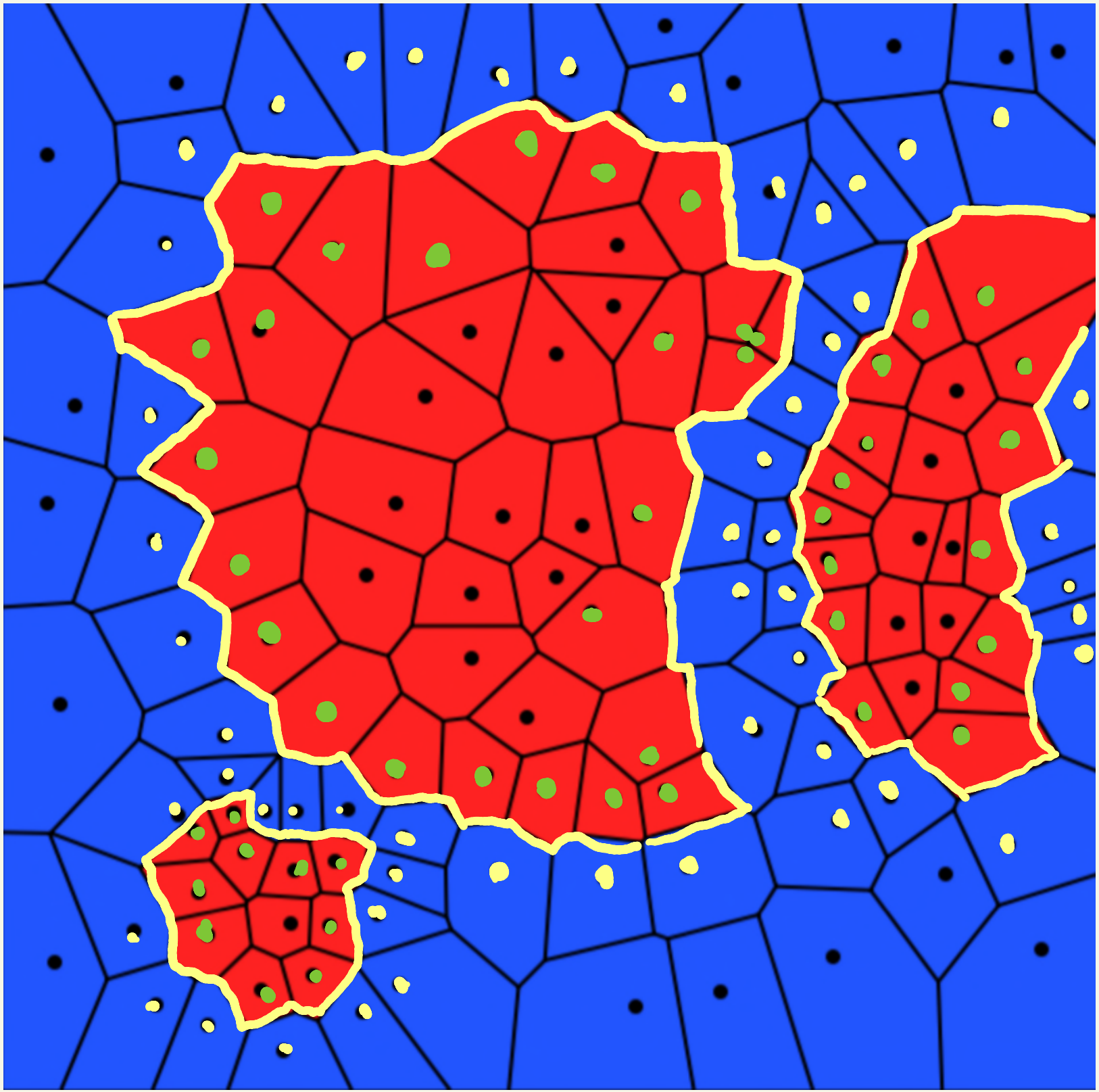
Data Stuctures

- Can use statistics on the data
- Independent queries are similar to ←
- EXP3 is a big hammer
- Have not talked about other types of struct
  EG prox point search [Demaine Iacono Langerman 2002]

An open problem

GEOMETRY

Find all points on the red-blue boundary



Do marine substrates 'look'and 'sound'the same? Supervised classification of multibeam acoustic data using autonomous underwater vehicle images

V Lucieer, NA Hill, NS Barrett, S Nichol - Estuarine, Coastal and Shelf ..., 2013 - Elsevier

In this study we outline the techniques used to transform multibeam acoustic data into spatial layers that can be used for predictive habitat modelling. The results allow us to identify multibeam attributes which may act as potential surrogates for environmental variables that ...

☆ 99 Cited by 89 Related articles All 6 versions

Native bees buffer the negative impact of climate warming on honey bee pollination of watermelon crops

R Rader, J Reilly, I Bartomeus... - Global change ..., 2013 - Wiley Online Library

If climate change affects pollinator-dependent crop production, this will have important implications for global food security because insect pollinators contribute to production for 75% of the leading global food crops. We investigate whether climate warming could result ...

☆ 99 Cited by 92 Related articles All 14 versions

N log K version is very well cited!

Imagine the impact of a instance-based optimal!

[PDF] Breast cancer diagnosis by using k-nearest neighbor with different distances and classification rules [PDF

SA Medjahed, TA Saadi... - International Journal of ..., 2013 - researchgate.net

Cancer diagnosis is one of the most studied problems in the medical domain. Several researchers have focused in order to improve performance and achieve to obtain satisfactory results. Breast cancer is one of cancer killer in the world. The diagnosis of this ...

☆ 99 Cited by 52 Related articles All 7 versions »

[HTML] Optimal model selection for posture recognition in home-based healthcare [HTM

S Zhang, P McCullagh, C Nugent, H Zheng... - International Journal of ..., 2011 - Springer

This paper investigates optimal model selection for posture recognition. Accuracy and computational time are related to the trained model in a supervised classification. An optimal model selection is important for a reliable activity monitoring system. Conventional guidance ...

☆ 99 Cited by 46 Related articles All 7 versions

Do marine substrates 'look'and 'sound'the same? Supervised classification of multibeam acoustic data using autonomous underwater vehicle images

V Lucieer, NA Hill, NS Barrett, S Nichol - Estuarine, Coastal and Shelf ..., 2013 - Elsevier

In this study we outline the techniques used to transform multibeam acoustic data into spatial layers that can be used for predictive habitat modelling. The results allow us to identify multibeam attributes which may act as potential surrogates for environmental variables that ...

☆ 99 Cited by 89 Related articles All 6 versions

Native bees buffer the negative impact of climate warming on honey bee pollination of watermelon crops

R Rader, J Reilly, I Bartomeus... - Global change ..., 2013 - Wiley Online Library

If climate change affects pollinator-dependent crop production, this will have important implications for global food security because insect pollinators contribute to production for 75% of the leading global food crops. We investigate whether climate warming could result ...

☆ 99 Cited by 92 Related articles All 14 versions

N log K version is very well cited!

Imagine the impact at a instance-based optimal!

* Not sure they realize it is 20

[PDF] Breast cancer diagnosis by using k-nearest neighbor with different distances and classification rules [PDF

SA Medjahed, TA Saadi... - International Journal of ..., 2013 - researchgate.net

Cancer diagnosis is one of the most studied problems in the medical domain. Several researchers have focused in order to improve performance and achieve to obtain satisfactory results. Breast cancer is one of cancer killer in the world. The diagnosis of this ...

☆ 99 Cited by 52 Related articles All 7 versions »

[HTML] Optimal model selection for posture recognition in home-based healthcare [HTM

S Zhang, P McCullagh, C Nugent, H Zheng... - International Journal of ..., 2011 - Springer

This paper investigates optimal model selection for posture recognition. Accuracy and computational time are related to the trained model in a supervised classification. An optimal model selection is important for a reliable activity monitoring system. Conventional guidance ...

☆ 99 Cited by 46 Related articles All 7 versions

Questions?