Recent Results on Semialgebraic Range Searching Lower Bounds

Some Overview of Data Structure Lower Bounds

Contents:

Introduction

- No general, unconditional framework

Introduction

- No general, unconditional framework (we can't even prove a $n^{\omega(1)}$ lower bound for 3-SAT)

Introduction

- No general, unconditional framework (we can't even prove a $n^{\omega(1)}$ lower bound for 3-SAT)
- Conditional: Conjecture Problem A is hard, then use reductions

Introduction

- No general, unconditional framework (we can't even prove a $n^{\omega(1)}$ lower bound for 3-SAT)
- Conditional: Conjecture Problem A is hard, then use reductions
- Pointer Machine: Disallows random access, only applies when we need to report a large list. A Navigation bottleneck, free information/computation

Introduction

- No general, unconditional framework (we can't even prove a $n^{\omega(1)}$ lower bound for 3-SAT)
- Conditional: Conjecture Problem A is hard, then use reductions
- Pointer Machine: Disallows random access, only applies when we need to report a large list. A Navigation bottleneck, free information/computation
- Cell-probe: Can't go beyond $\Omega(\log n)$ static query time; Information bottleneck, free computation

Introduction

- No general, unconditional framework (we can't even prove a $n^{\omega(1)}$ lower bound for 3-SAT)
- Conditional: Conjecture Problem A is hard, then use reductions
- Pointer Machine: Disallows random access, only applies when we need to report a large list. A Navigation bottleneck, free information/computation
- Cell-probe: Can't go beyond $\Omega(\log n)$ static query time; Information bottleneck, free computation
- Semi-group: Limits what DS can store and do. Only for weighted counting, weights from a semi-group, i.e., no subtractions
- Group: Limits what DS can store and do. Allows subtractions but we only know how to do dynamic lower bounds

Introduction

- No general, unconditional framework (we can't even prove a $n^{\omega(1)}$ lower bound for 3-SAT)
- Conditional: Conjecture Problem A is hard, then use reductions
- Pointer Machine: Disallows random access, only applies when we need to report a large list. A Navigation bottleneck, free information/computation
- Cell-probe: Can't go beyond $\Omega(\log n)$ static query time; Information bottleneck, free computation
- Semi-group: Limits what DS can store and do. Only for weighted counting, weights from a semi-group, i.e., no subtractions
- Group: Limits what DS can store and do. Allows subtractions but we only know how to do dynamic lower bounds

Must avoid icebergs!

The Pointer Machine Model

Range Reporting

Range Reporting:

- A general class of Computational Geometric problems
- Input: A set of n objects, e.g., points, given by coordinates.
  - In 2D we have (x_i, y_i), $1 \leq i \leq n$
- We want to build a Data Structure:
  - Process the data using some preprocessing time, $P(n)$
  - Store the process data using $S(n)$ units of storage, i.e., space

Range Reporting

Range Reporting:

- A general class of Computational Geometric problems
- Input: A set of n objects, e.g., points, given by coordinates.
  - In 2D we have (x_i, y_i), $1 \leq i \leq n$
- We want to build a Data Structure:
  - Process the data using some preprocessing time, $P(n)$
  - Store the process data using $S(n)$ units of storage, i.e., space

The Goal:

- Answer queries
- A query is a geometric region or object.
  - Triangle
  - Circle
  - Point
  - ...
- Output: List of all the input objects that intersect the query object

Range Reporting

Range Reporting:

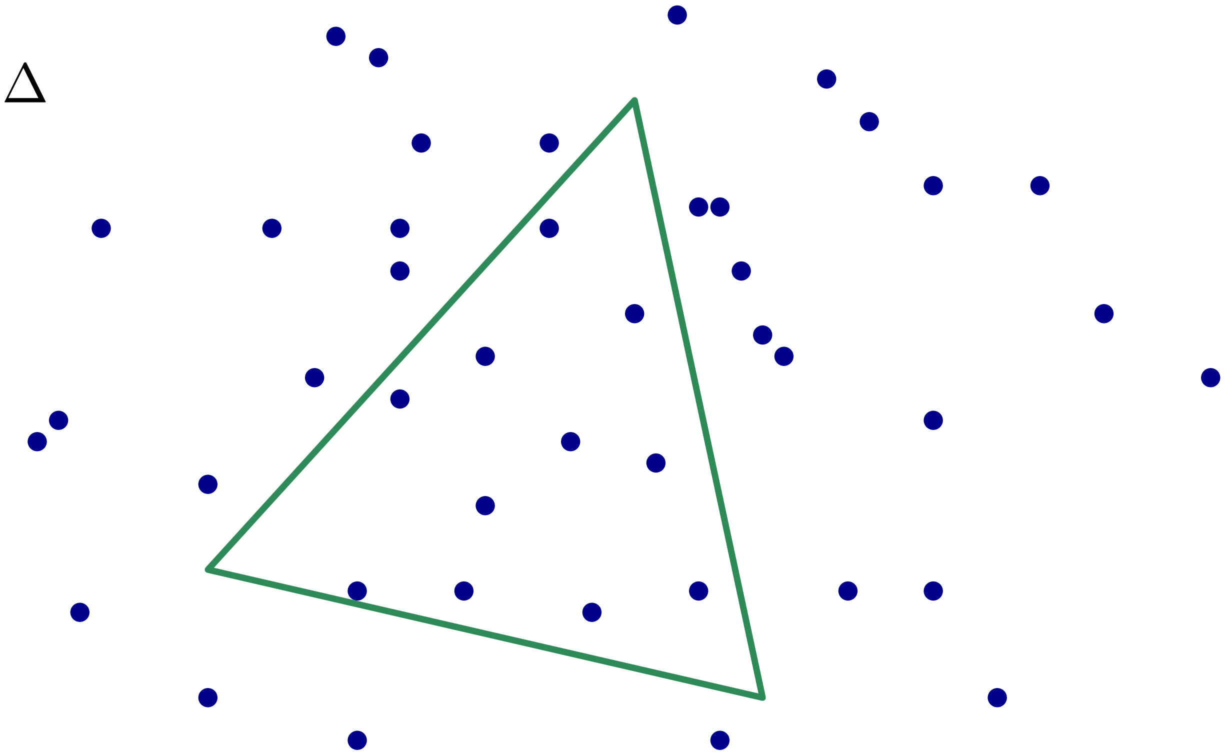
- A general class of Computational Geometric problems
- Input: A set of n objects, e.g., points, given by coordinates.
  - In 2D we have (x_i, y_i), $1 \leq i \leq n$
- We want to build a Data Structure:
  - Process the data using some preprocessing time, $P(n)$
  - Store the process data using $S(n)$ units of storage, i.e., space

The Goal:

- Answer queries
- A query is a geometric region or object.
  - Triangle
  - Circle
  - Point
  - ...
- Output: List of all the input objects that intersect the query object
- k: Output size

A 2D Range Reporting Example

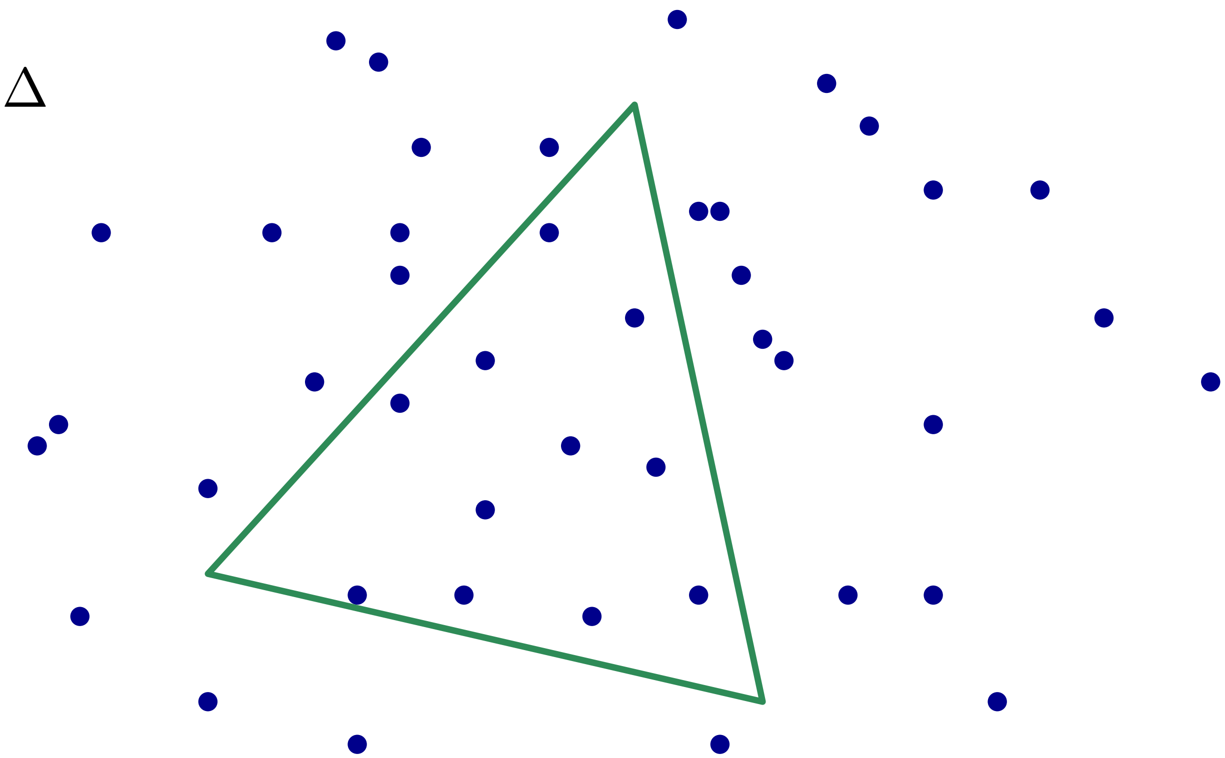
- Input: n points in 2D
- Query: A triangle Δ
- Output: List of k points inside Δ



A 2D Range Reporting Example

- Input: n points in 2D
- Query: A triangle Δ
- Output: List of k points inside Δ

- We want to spend $O(n)$ space
- Query time?

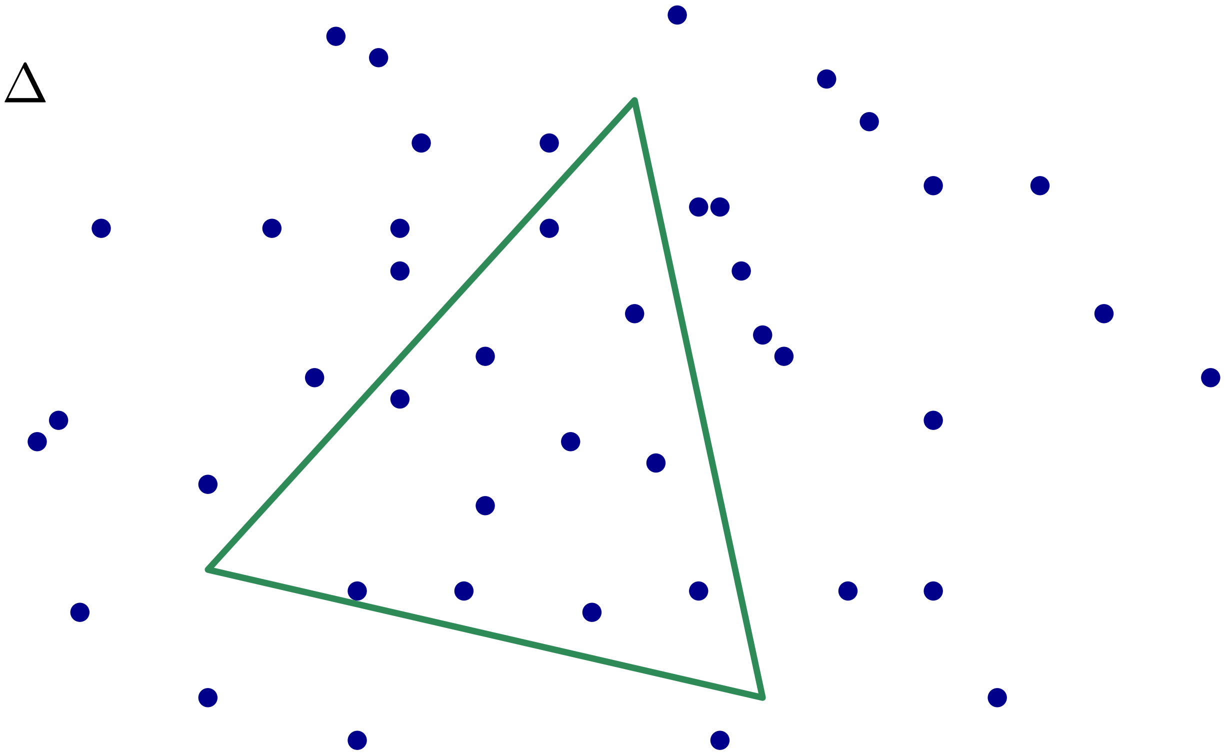


A 2D Range Reporting Example

- Input: n points in 2D
- Query: A triangle Δ
- Output: List of k points inside Δ

- We want to spend $O(n)$ space
- Query time?

- Answer: $O(\sqrt{n} + k)$

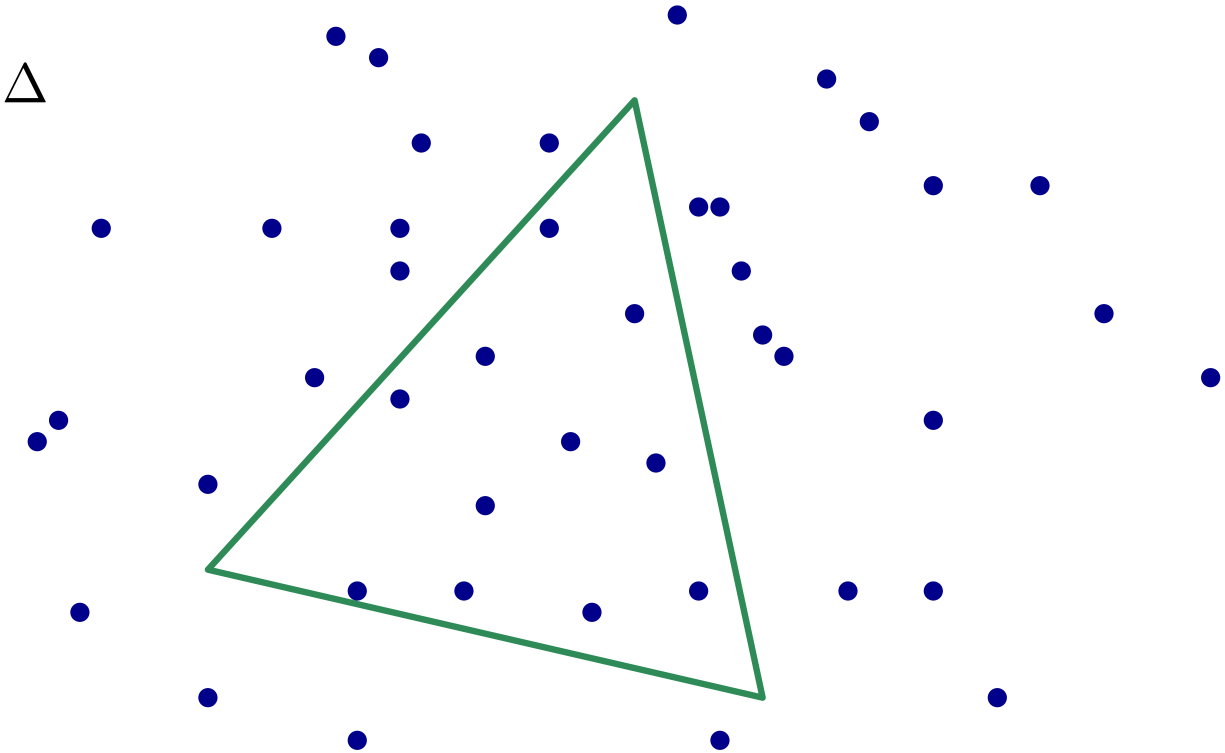


A 2D Range Reporting Example

- Input: n points in 2D
- Query: A triangle Δ
- Output: List of k points inside Δ

- We want to spend $O(n)$ space
- Query time?

- Answer: $O(\sqrt{n} + k)$
  - Some people invented crazy techniques: cutting lemma, partition theorem, partition trees, etc.



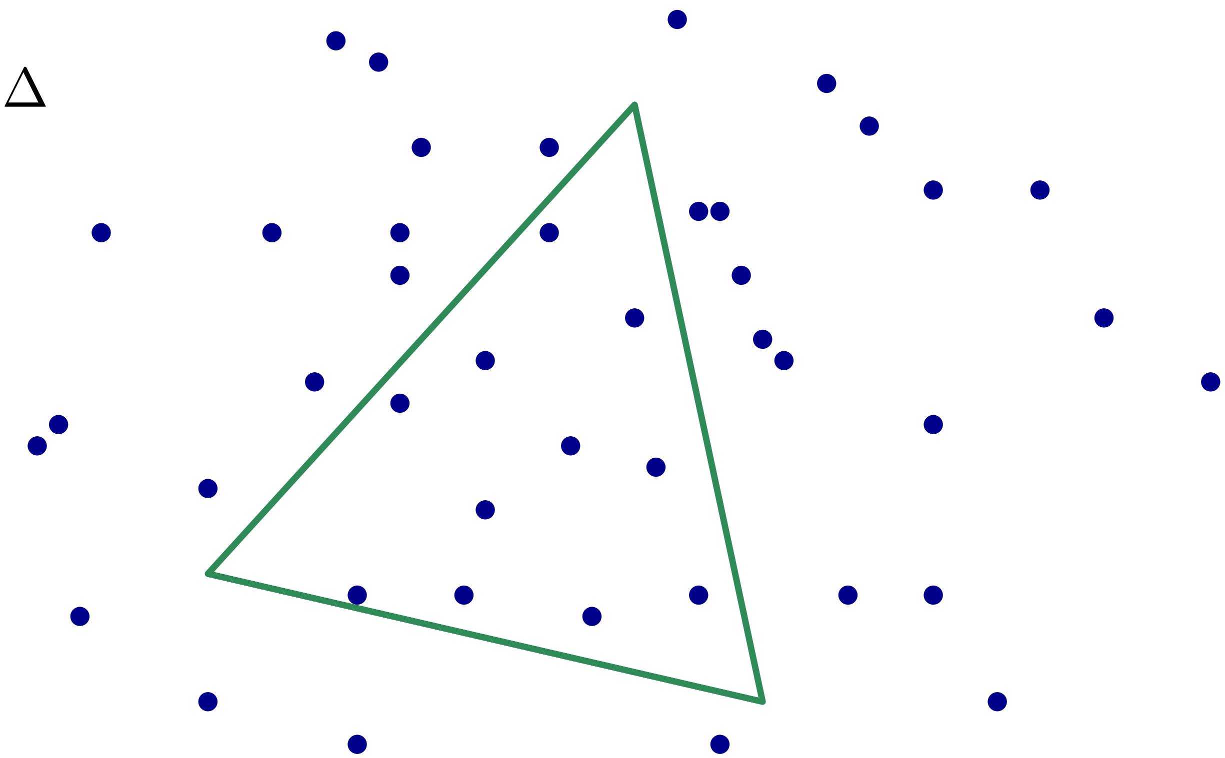
A 2D Range Reporting Example

- Input: n points in 2D
- Query: A triangle Δ
- Output: List of k points inside Δ

- We want to spend $O(n)$ space
- Query time?

- Answer: $O(\sqrt{n} + k)$
  - Some people invented crazy techniques: cutting lemma, partition theorem, partition trees, etc.

- This is optimal!



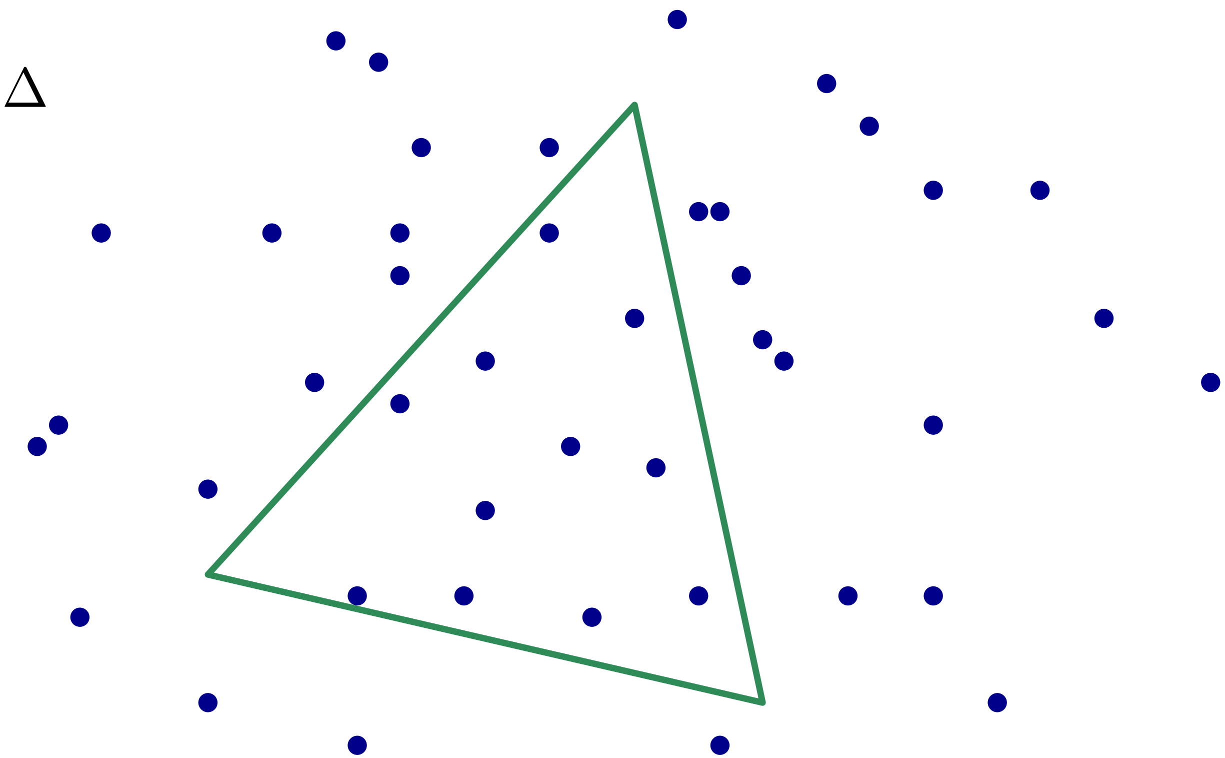
A 2D Range Reporting Example

- Input: n points in 2D
- Query: A triangle Δ
- Output: List of k points inside Δ

- We want to spend $O(n)$ space
- Query time?

- Answer: $O(\sqrt{n} + k)$
  - Some people invented crazy techniques: cutting lemma, partition theorem, partition trees, etc.

- This is optimal!

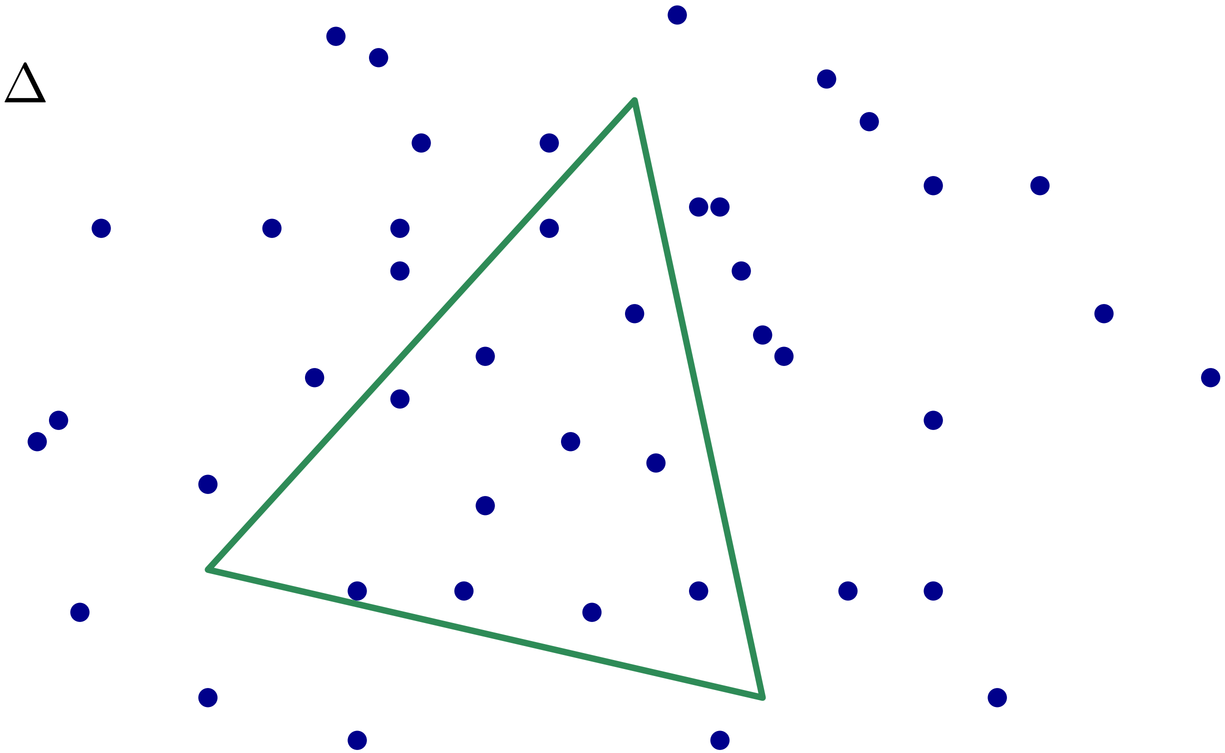


Assume we have a data structure:

1. Works on any input of n points
2. Uses $O(n)$ space
3. Finds all the points inside any triangle
4. Query time is $O(Q(n) + k)$

A 2D Range Reporting Example

- Input: n points in 2D
- Query: A triangle Δ
- Output: List of k points inside Δ



- We want to spend $O(n)$ space
- Query time?
- Answer: $O(\sqrt{n} + k)$
  - Some people invented crazy techniques: cutting lemma, partition theorem, partition trees, etc.
- This is optimal!

Assume we have a data structure:

1. Works on any input of n points
2. Uses $O(n)$ space
3. Finds all the points inside any triangle
4. Query time is $O(Q(n) + k)$ $\implies Q(n) = \Omega(\sqrt{n})$

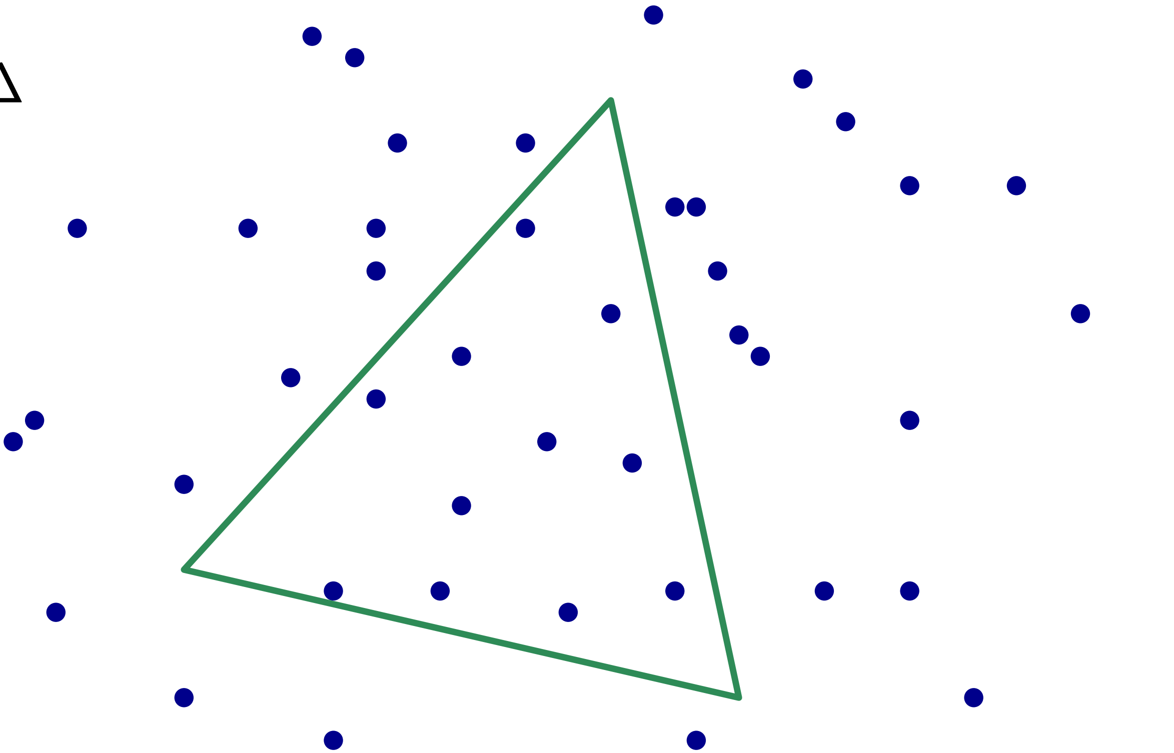
A 2D Range Reporting Example

- Input: n points in 2D
- Query: A triangle Δ
- Output: List of k points inside Δ

- We want to spend $O(n)$ space
- Query time?

- Answer: $O(\sqrt{n} + k)$
  - Some people invented crazy techniques: cutting lemma, partition theorem, partition trees, etc.

- This is optimal!



Assume we have a data structure:

1. Works on any input of n points
2. Uses $O(n)$ space
3. Finds all the points inside any triangle
4. Query time is $O(Q(n) + k)$

This is a claim that holds for *any* data structure that satisfies 1-4!!

$\implies Q(n) = \Omega(\sqrt{n})$

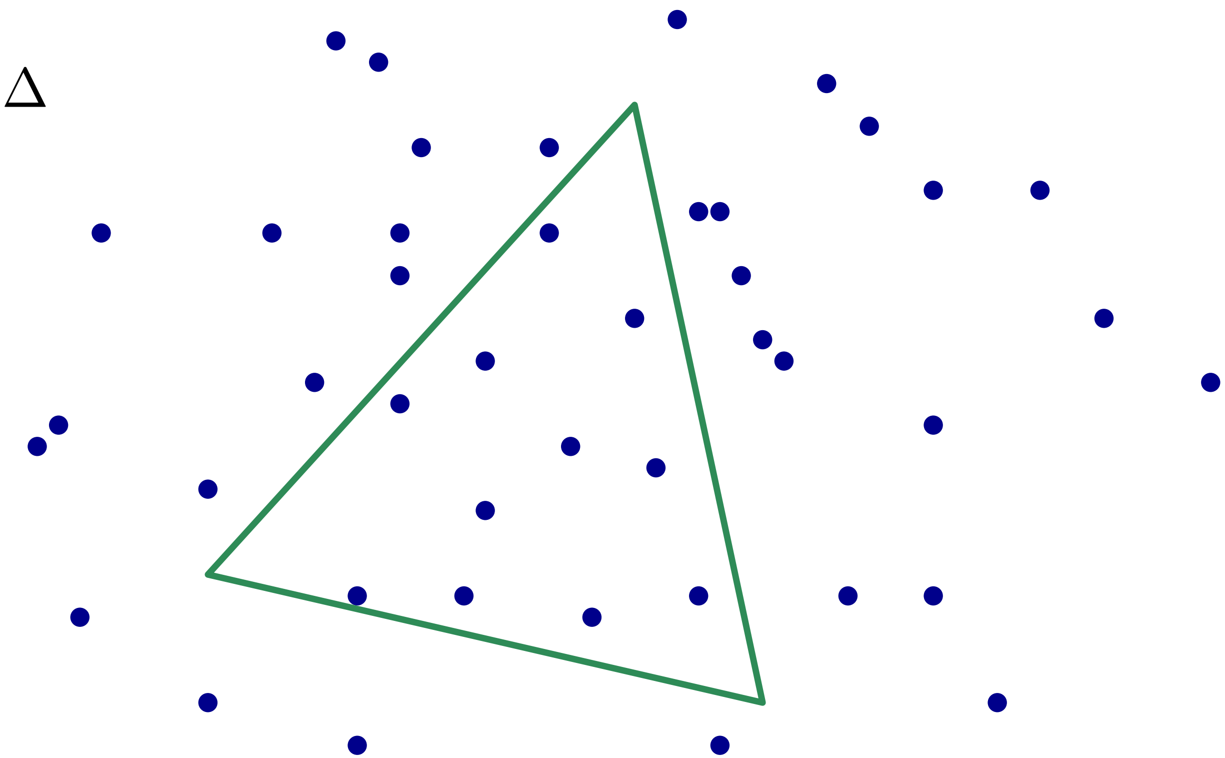
A 2D Range Reporting Example

- Input: n points in 2D
- Query: A triangle Δ
- Output: List of k points inside Δ

- We want to spend $O(n)$ space
- Query time?

- Answer: $O(\sqrt{n} + k)$
  - Some people invented crazy techniques: cutting lemma, partition theorem, partition trees, etc.

- This is optimal!



Assume we have a data structure:

1. Works on any input of n points
2. Uses $O(n)$ space
3. Finds all the points inside any triangle
4. Query time is $O(Q(n) + k)$ $\implies Q(n) = \Omega(\sqrt{n})$

This is a claim that holds for *any* data structure that satisfies 1-4!!

How do we prove it?

Data Structure Lower Bounds

Theorem we want to prove

Assume we have a data structure:

1. Given any input of n points in 2D,
2. stores them using $O(n)$ space, s.t., it
3. finds all the points inside any given query triangle, using
4. query time of $O(Q(n) + k)$.

Then, we must have $Q(n) = \Omega(\sqrt{n})$

Data Structure Lower Bounds

Theorem we want to prove

Assume we have a data structure:

1. Given any input of n points in 2D,
2. stores them using $O(n)$ space, s.t., it
3. finds all the points inside any given query triangle, using
4. query time of $O(Q(n) + k)$.

Then, we must have $Q(n) = \Omega(\sqrt{n})$

$\Updownarrow$

Theorem we want to prove

It is impossible to have a data structure that:

1. Given any input of n points in 2D,
2. stores them using $O(n)$ space, s.t., it
3. finds all the points inside any given query triangle, using
4. query time of $o(\sqrt{n}) + O(k)$.

Data Structure Lower Bounds

Theorem we want to prove

Assume we have a data structure:

1. Given any input of n points in 2D,
2. stores them using $O(n)$ space, s.t., it
3. finds all the points inside any given query triangle, using
4. query time of $O(Q(n) + k)$.

Then, we must have $Q(n) = \Omega(\sqrt{n})$

$\Updownarrow$

Theorem we want to prove **Impossibility result!**

It is impossible to have a data structure that:

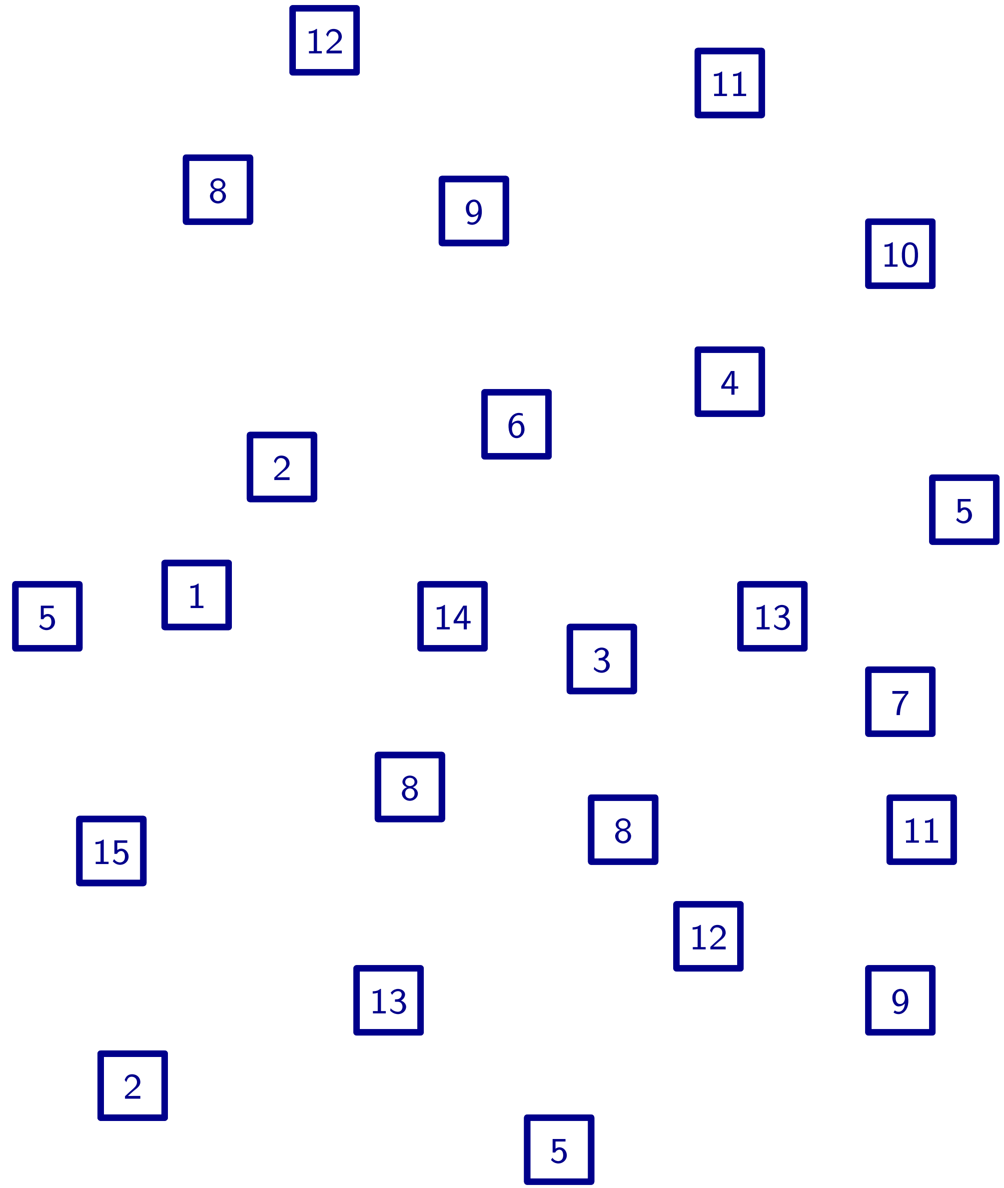
1. Given any input of n points in 2D,
2. stores them using $O(n)$ space, s.t., it
3. finds all the points inside any given query triangle, using
4. query time of $o(\sqrt{n}) + O(k)$.

The Model of Computation: A Pointer Machine

Assume, the input is a set P of n items (e.g., points)

DS:

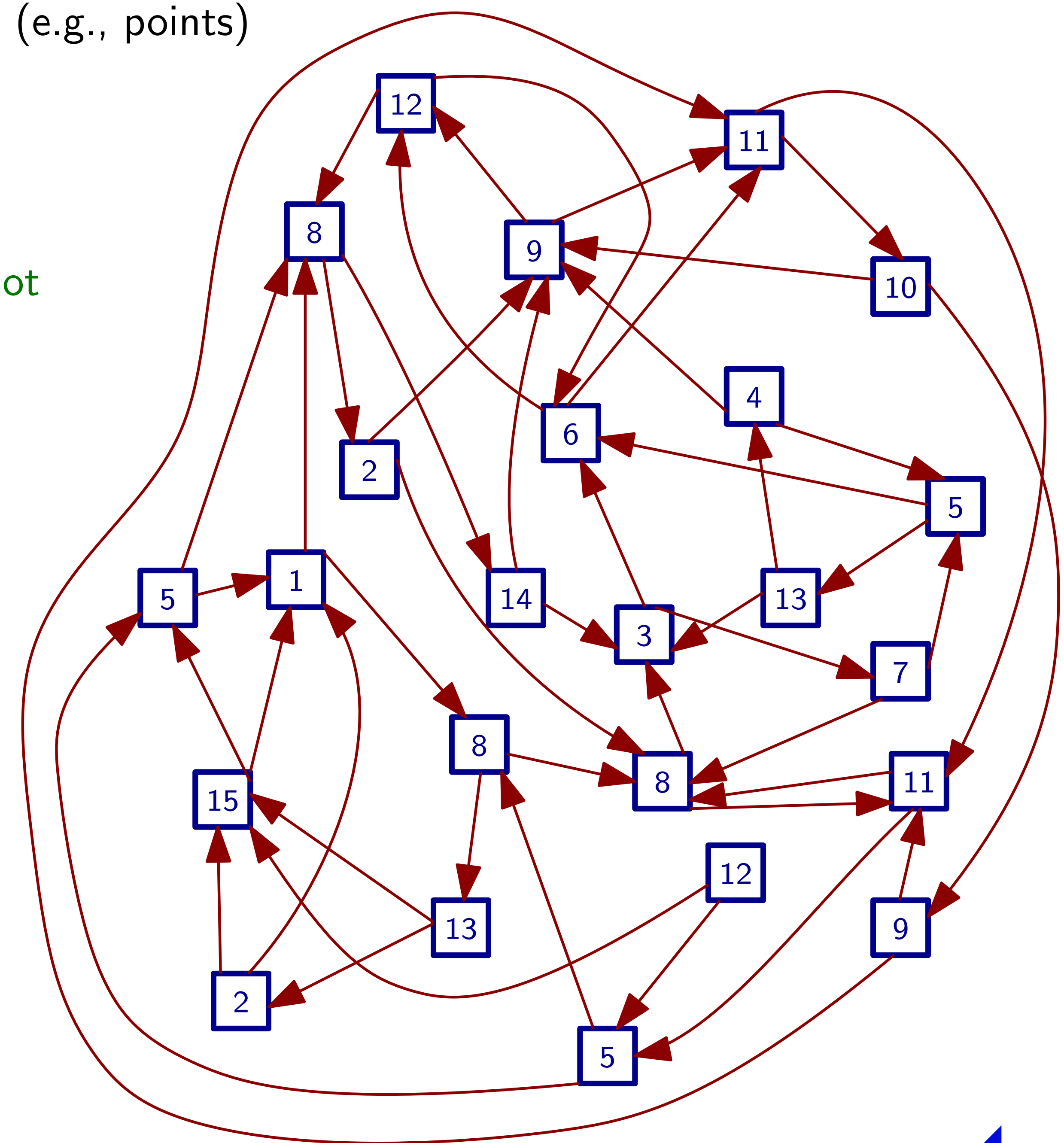
- Storage is a collection of cells
- A cell stores **one** item
- A cell points to two other cells
- There is a special node called the root



The Model of Computation: A Pointer Machine

Assume, the input is a set P of n items (e.g., points)
DS:

- Storage is a collection of cells
- A cell stores **one** item
- A cell points to two other cells
- There is a special node called the root

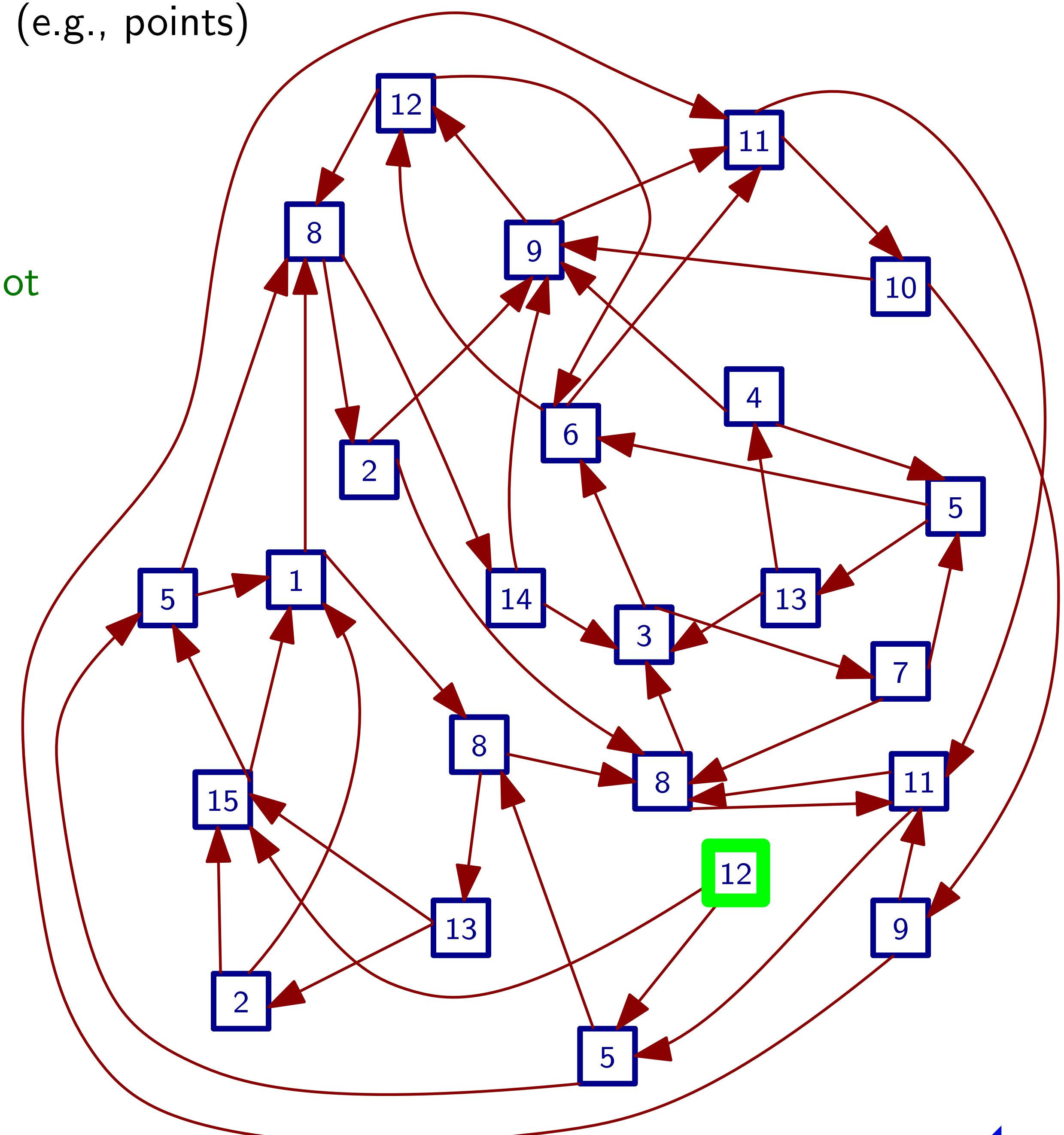


The Model of Computation: A Pointer Machine

Assume, the input is a set P of n items (e.g., points)

DS:

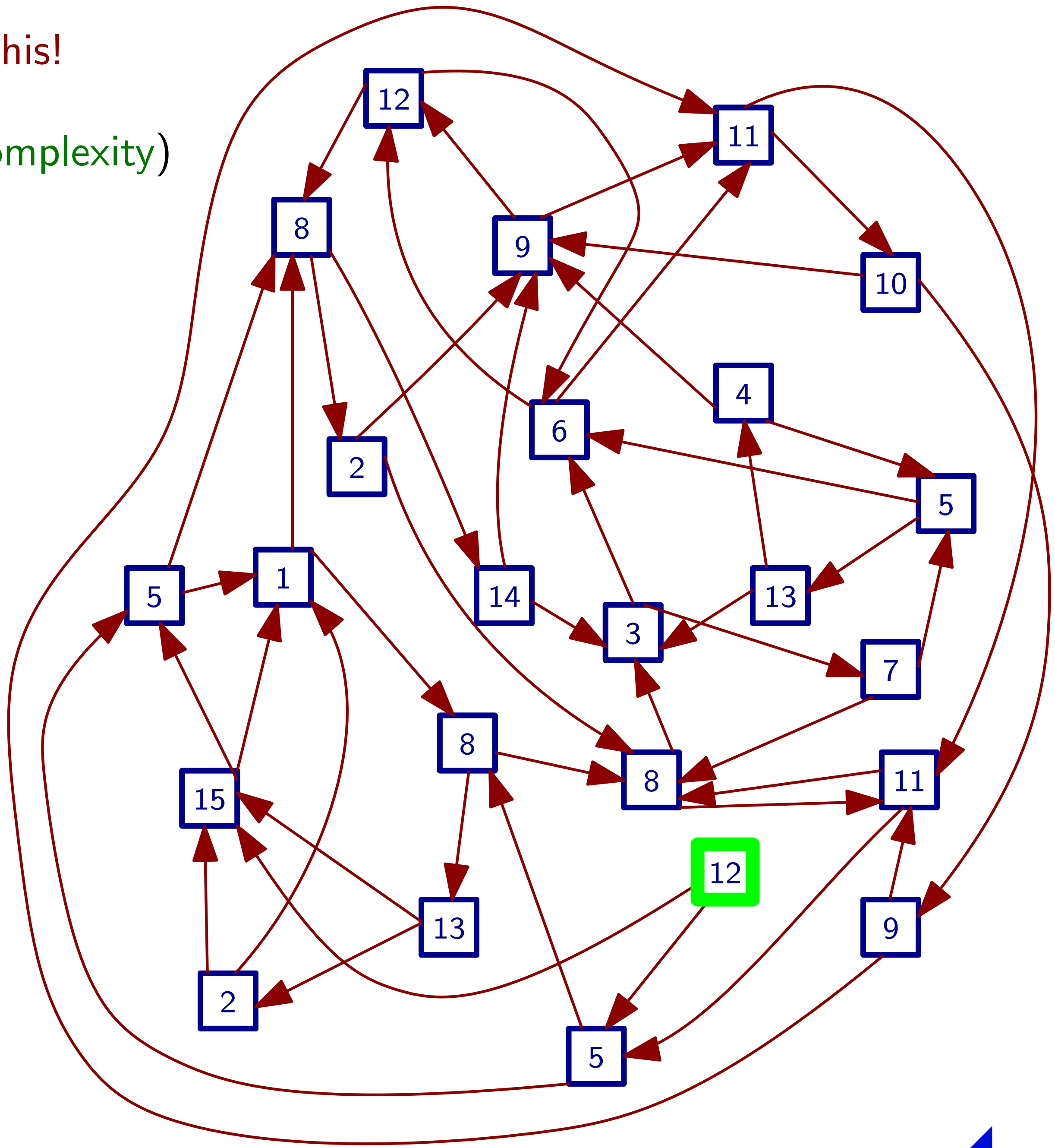
- Storage is a collection of cells
- A cell stores **one** item
- A cell points to two other cells
- There is a special node called the root



The Model of Computation: A Pointer Machine

Don't care how long it takes to build this!

# of cells is the space usage (space complexity)

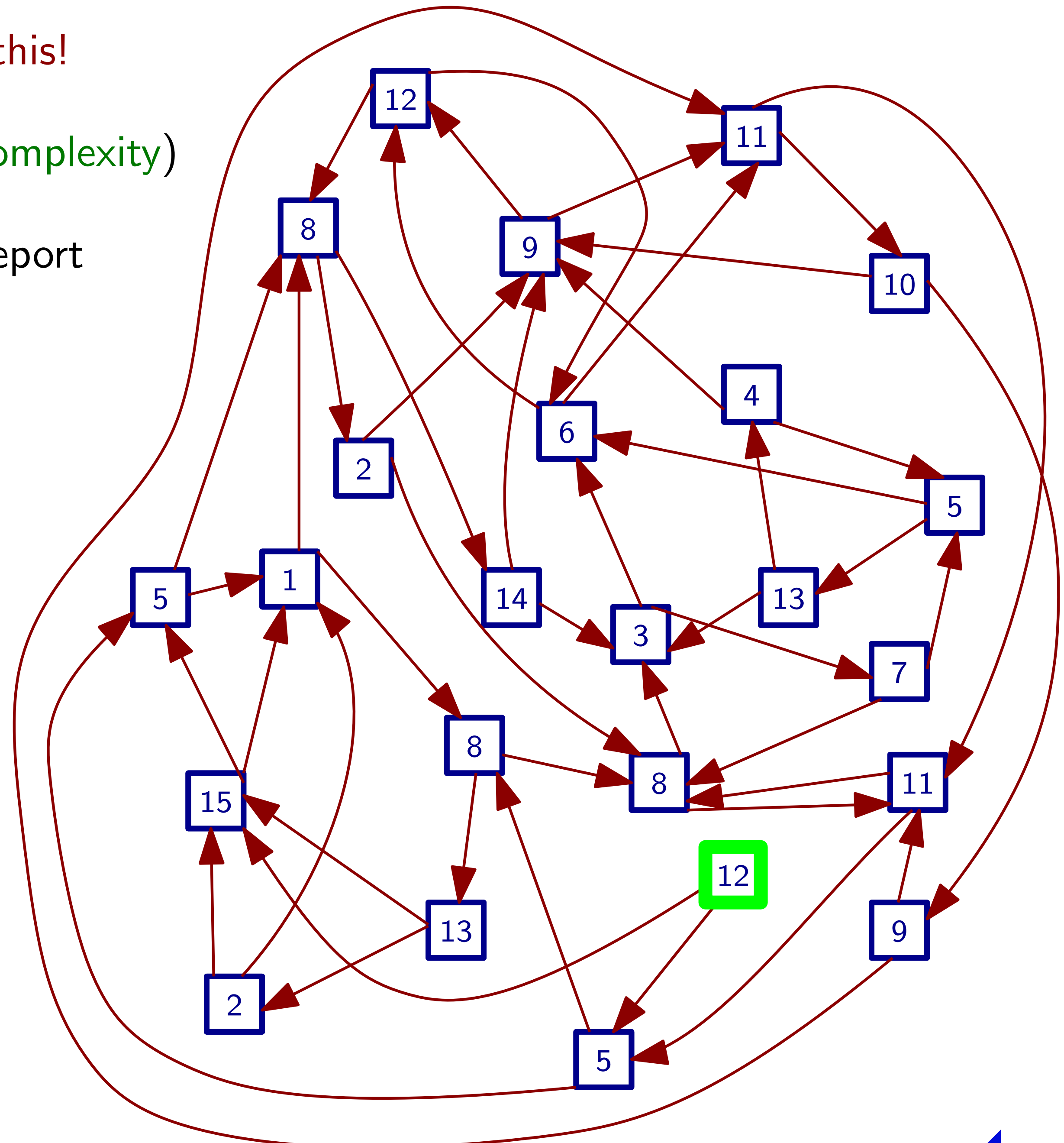


The Model of Computation: A Pointer Machine

Don't care how long it takes to build this!

$\#$ of cells is the space usage (space complexity)

Given a query q, assume we need to report $P_q \subset P$:

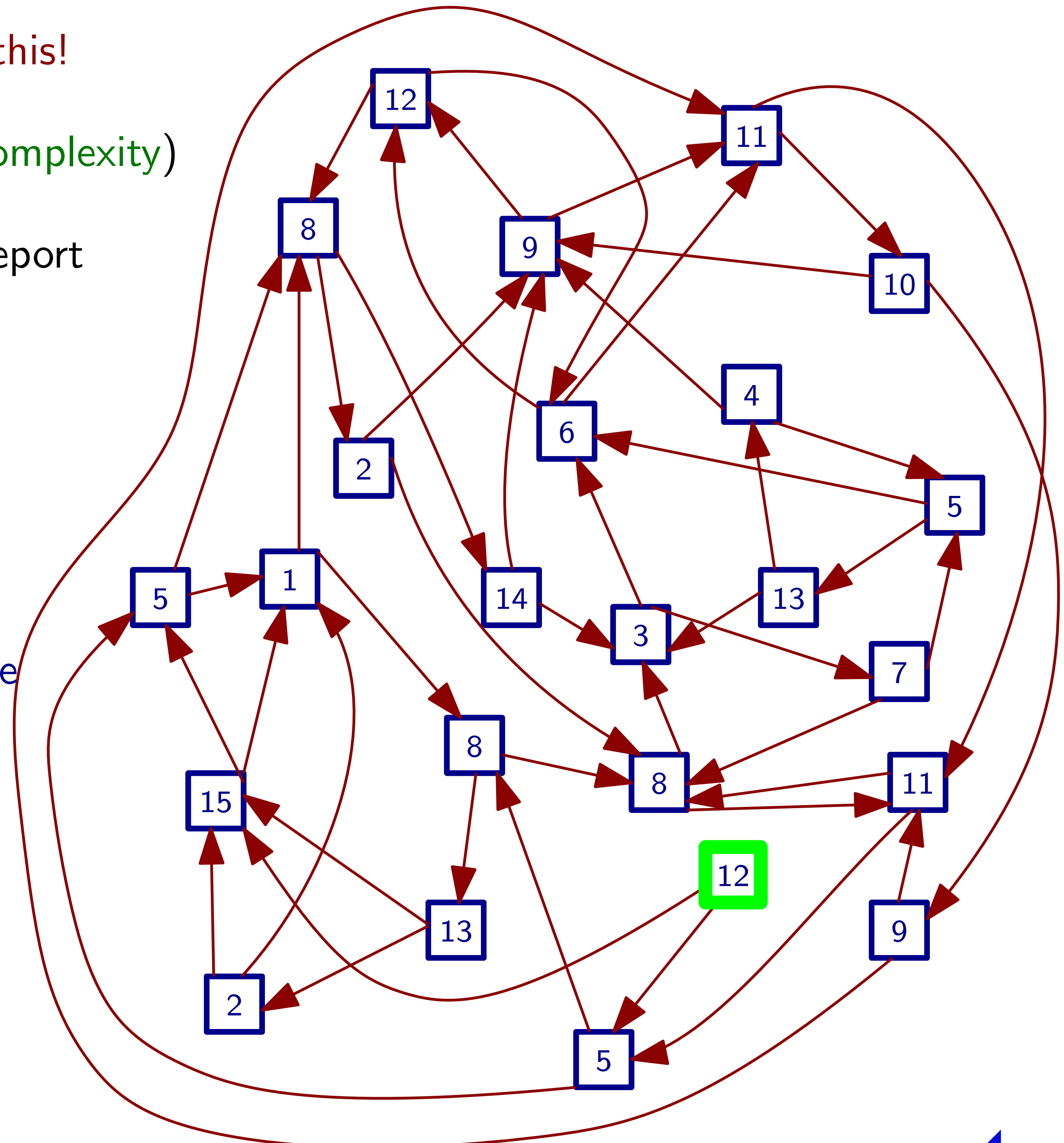


The Model of Computation: A Pointer Machine

Don't care how long it takes to build this!

$\#$ of cells is the space usage (space complexity)

Given a query q, assume we need to report $P_q \subset P$:



- $\forall x \in P_q$: We must visit a cell that stores x
- Only through pointer navigation
- $\#$ of pointer navigations = query time

The Model of Computation: A Pointer Machine

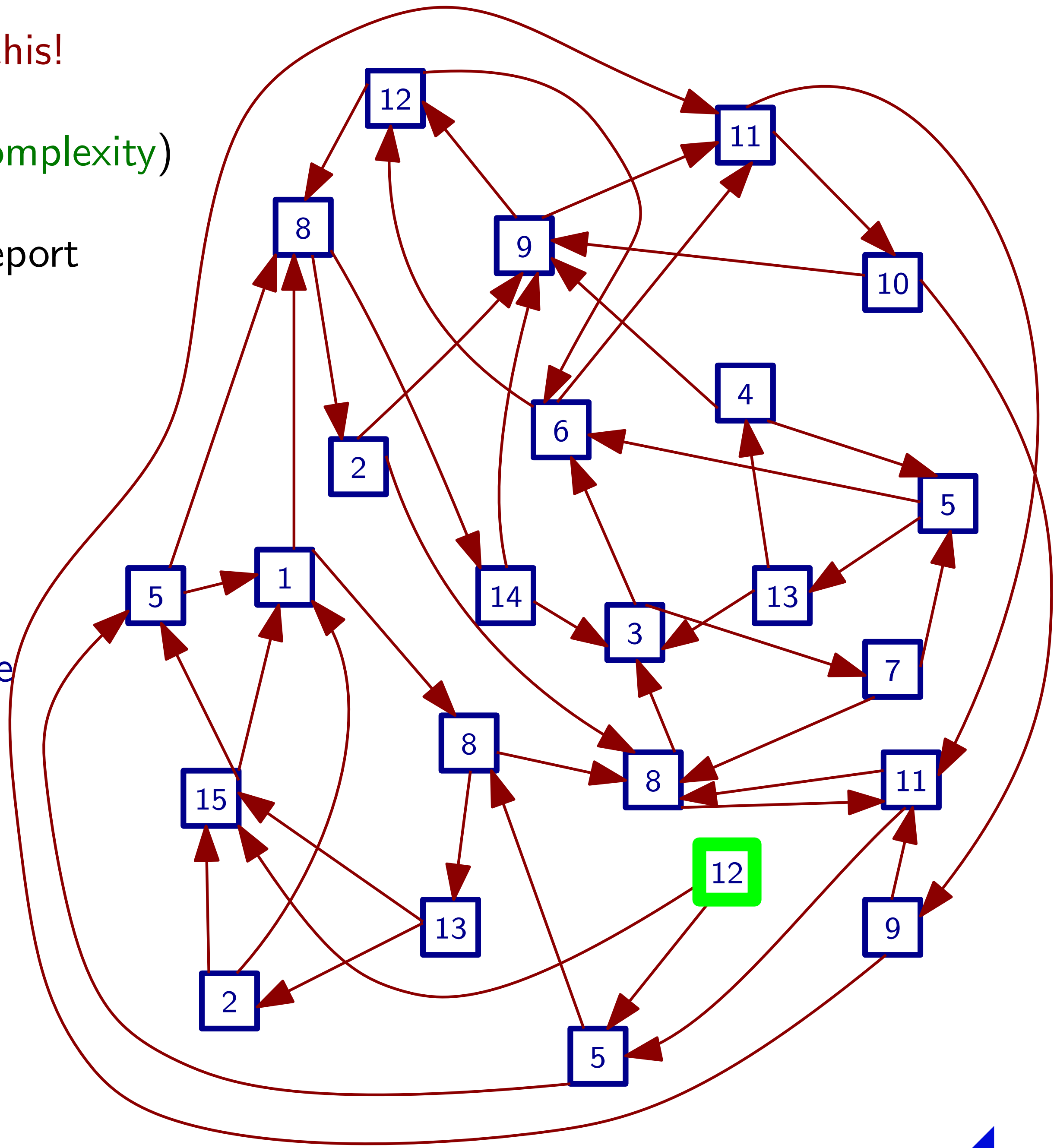
Don't care how long it takes to build this!

# of cells is the space usage (space complexity)

Given a query q, assume we need to report $P_q \subset P$:

- $\forall x \in P_q$: We must visit a cell that stores x
- Only through pointer navigation
- # of pointer navigations = query time

- Computation is free!
- Information is free!



The Model of Computation: A Pointer Machine

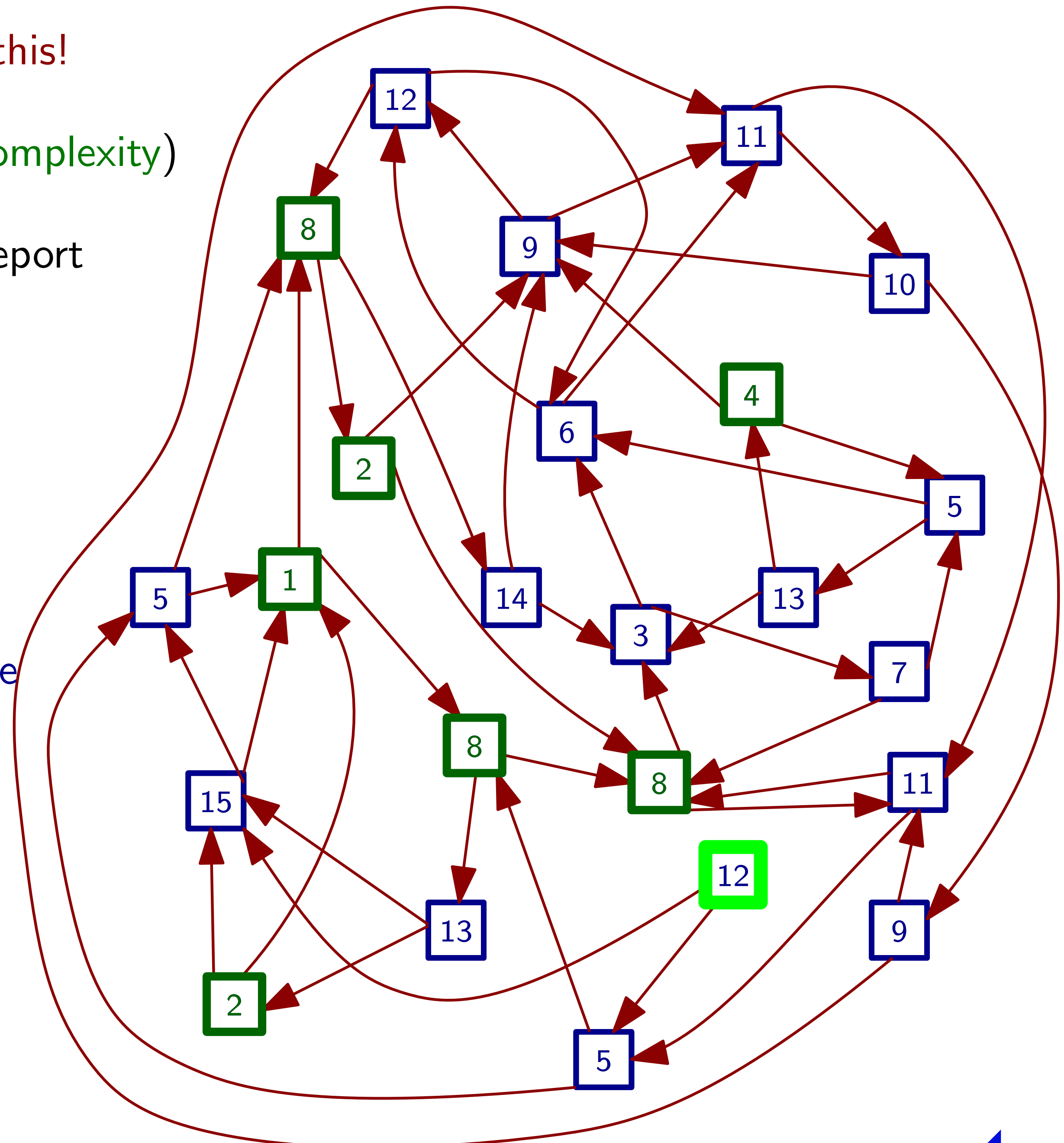
Don't care how long it takes to build this!

$\#$ of cells is the space usage (space complexity)

Given a query q, assume we need to report $P_q \subset P$:

- $\forall x \in P_q$: We must visit a cell that stores x
- Only through pointer navigation
- $\#$ of pointer navigations $=$ query time

We want to report $\{1, 2, 4, 8\}$

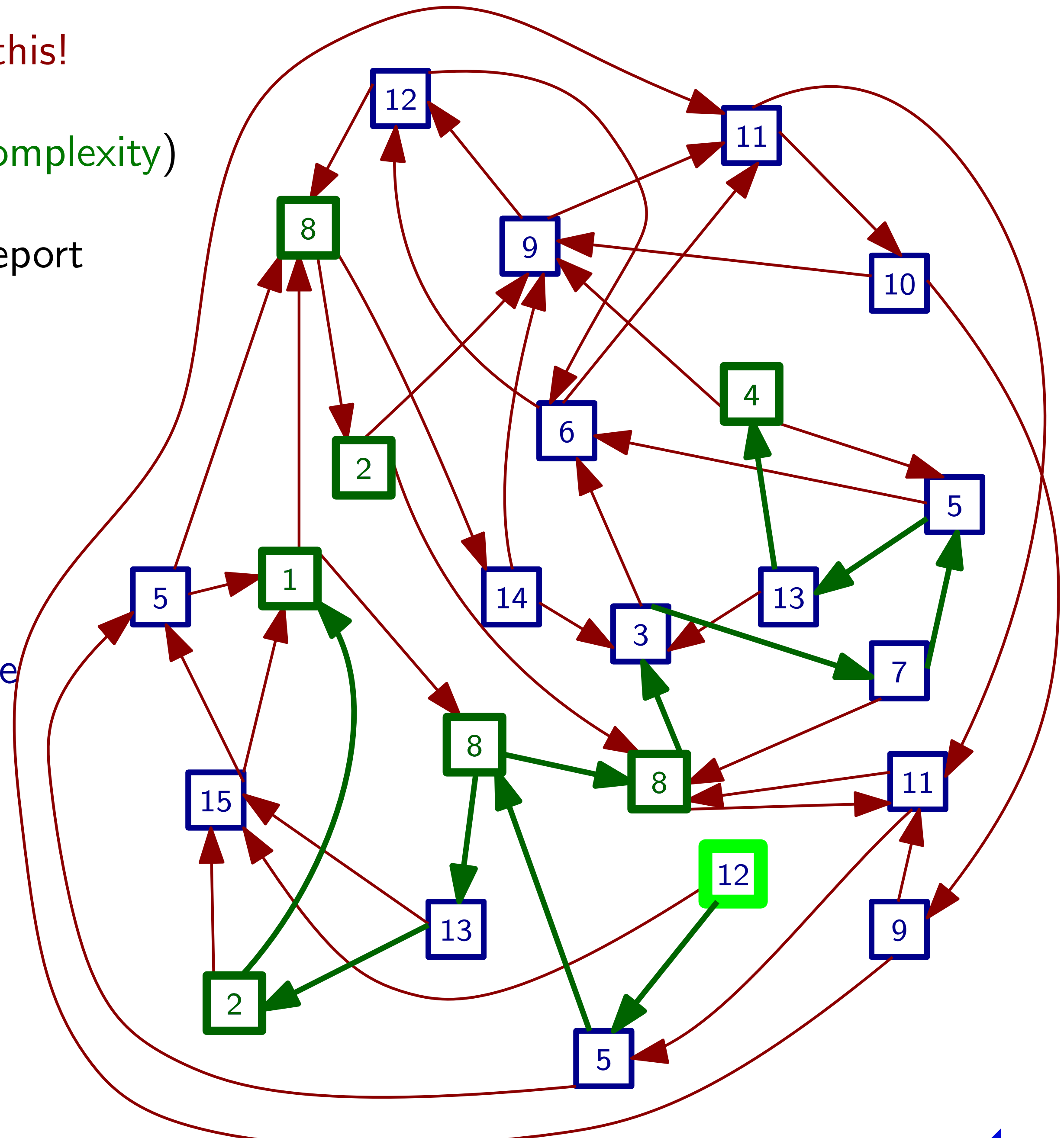


The Model of Computation: A Pointer Machine

Don't care how long it takes to build this!

$\#$ of cells is the space usage (space complexity)

Given a query q, assume we need to report $P_q \subset P$:



- $\forall x \in P_q$: We must visit a cell that stores x
- Only through pointer navigation
- $\#$ of pointer navigations $=$ query time

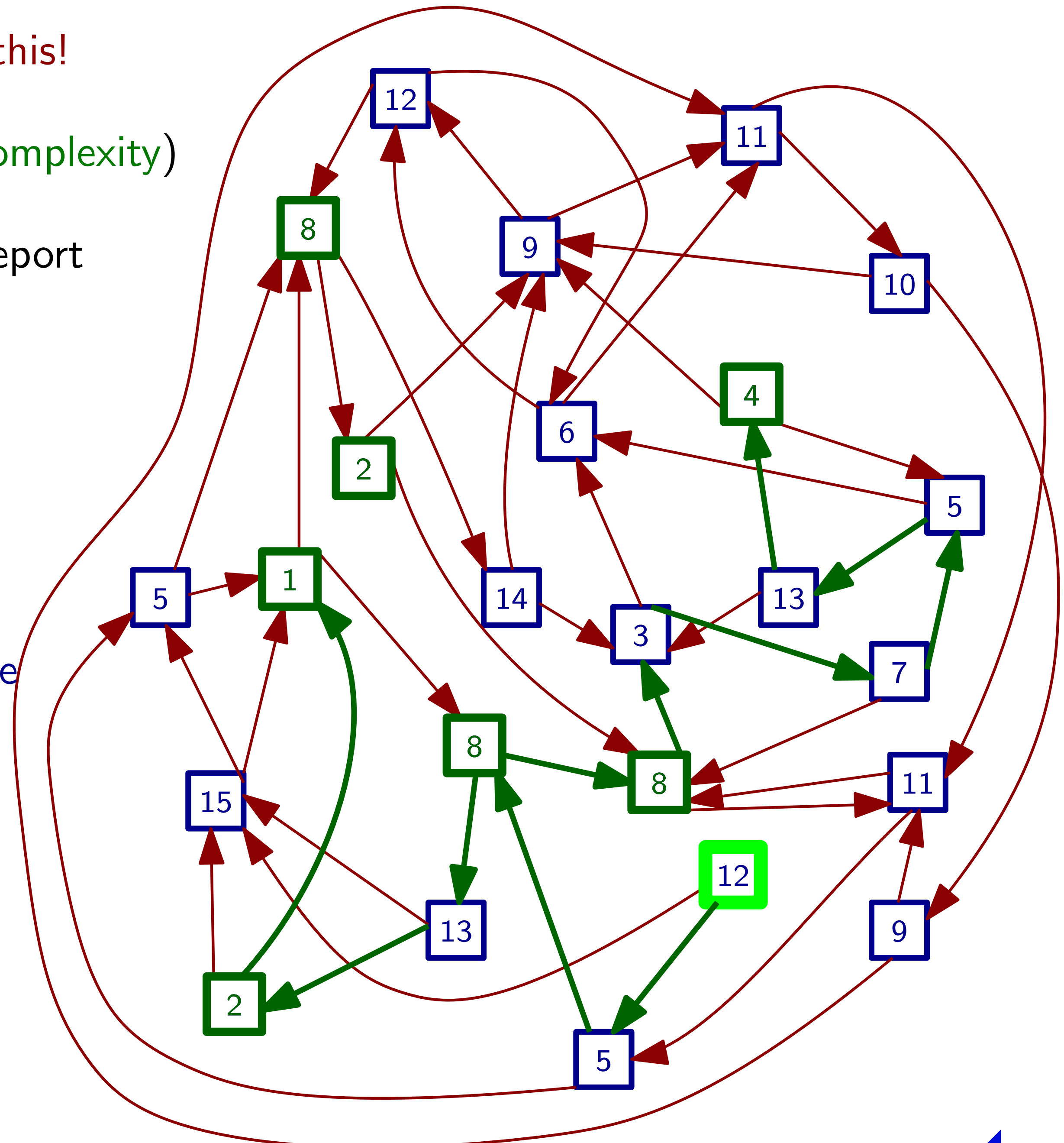
We want to report $\{1, 2, 4, 8\}$

The Model of Computation: A Pointer Machine

Don't care how long it takes to build this!

$\#$ of cells is the space usage (space complexity)

Given a query q, assume we need to report $P_q \subset P$:



- $\forall x \in P_q$: We must visit a cell that stores x
- Only through pointer navigation
- $\#$ of pointer navigations $=$ query time

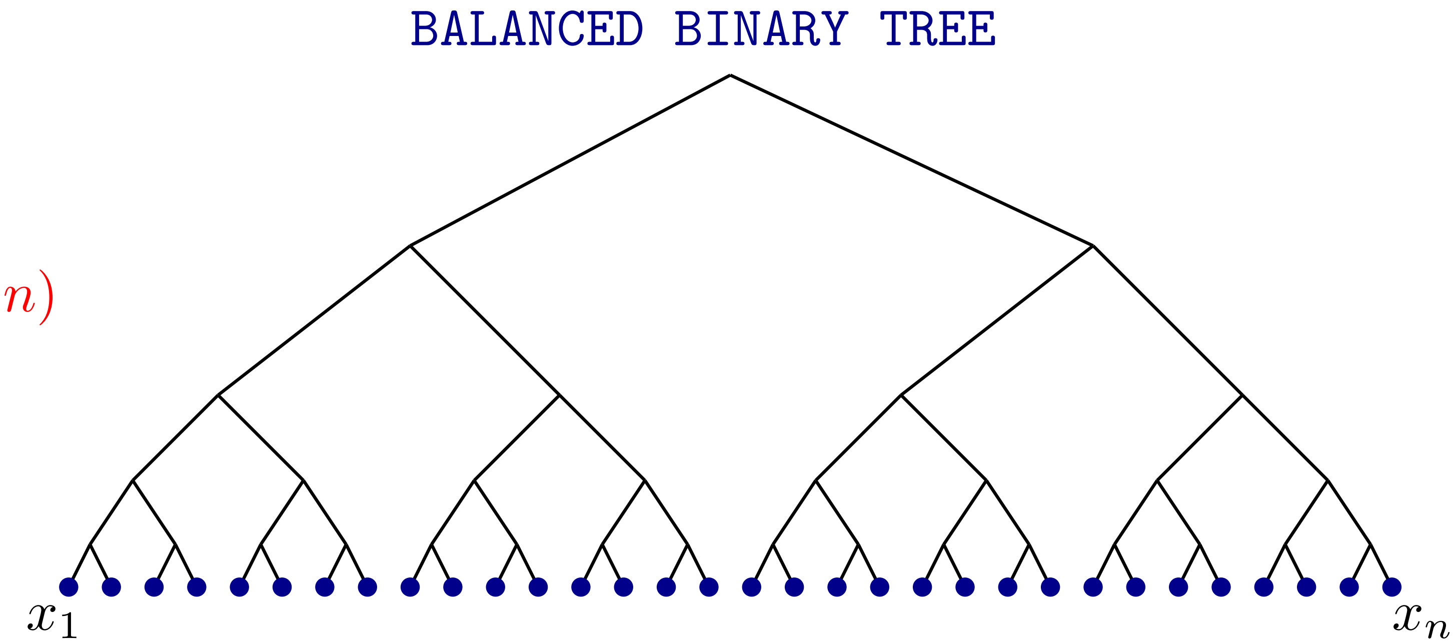
We want to report $\{1, 2, 4, 8\}$

We used 11 pointers $\Rightarrow$ query time at least 11

The Model of Computation: A Pointer Machine

Space: $O(n)$

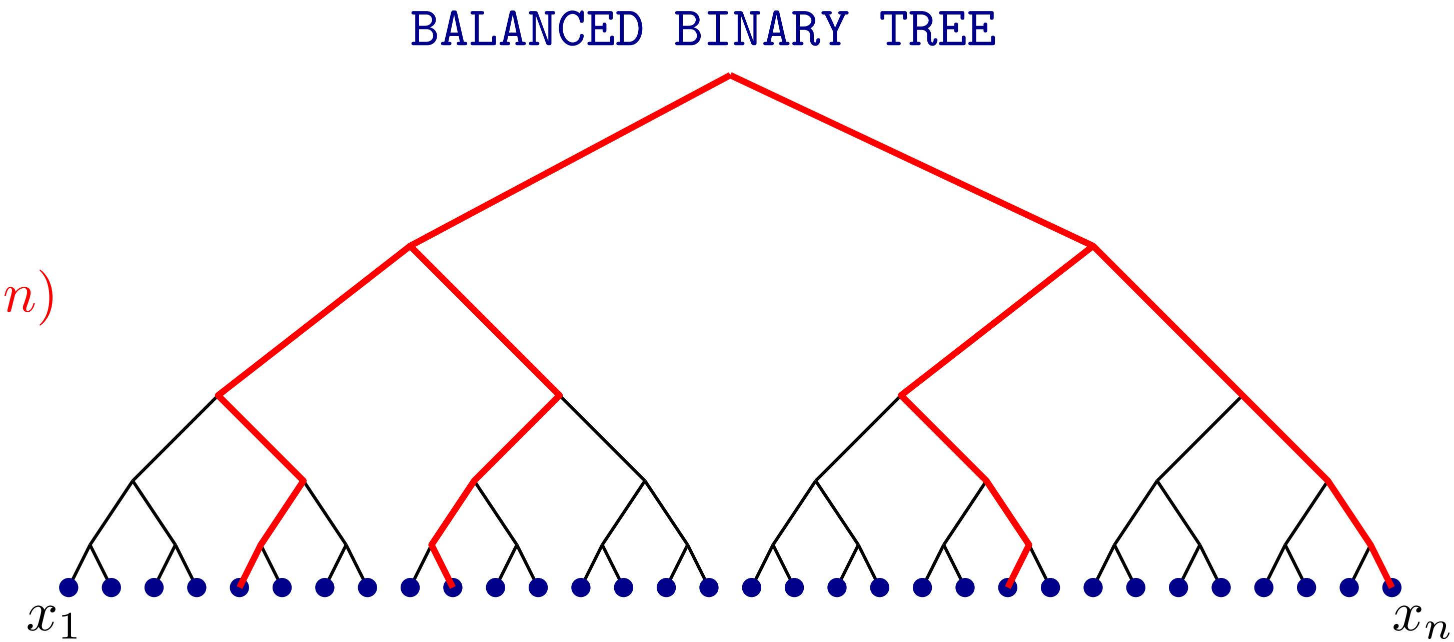
Query: $O(k \log n)$



The Model of Computation: A Pointer Machine

Space: $O(n)$

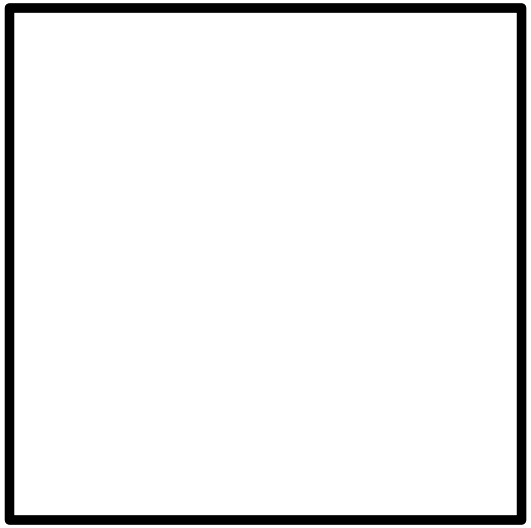
Query: $O(k \log n)$



- Query time must be $Q(n) + O(k)$ (or $Q(n) + o(k \log n)$)
- PM can simulate RAM w/ extra $O(\log n)$ factor
  - LB in PM with $Q(n) + O(k \log n) \Rightarrow Q(n)/\log n + O(k)$ LB in RAM

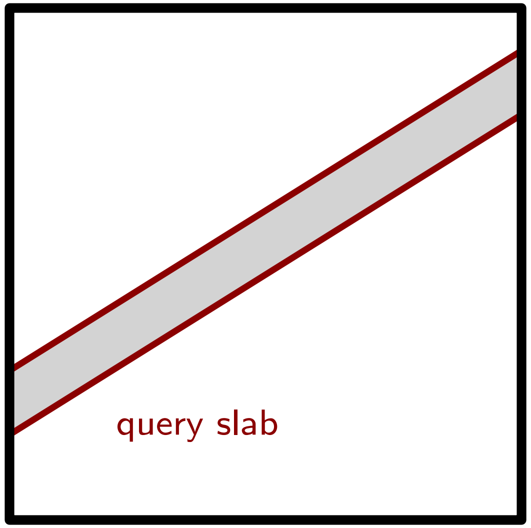
A Framework Theorem

Unit square in 2D



A Framework Theorem

Unit square in 2D

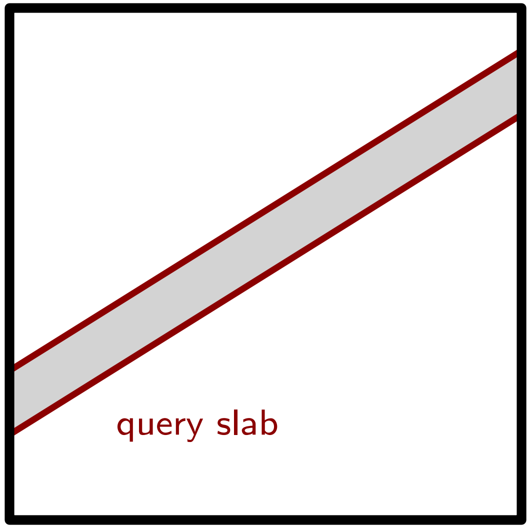


Problem:

- Input: n points
- Goal: A data structure
- Query: A region inside the unit square
- Output: All the points inside the region

A Framework Theorem

Unit square in 2D



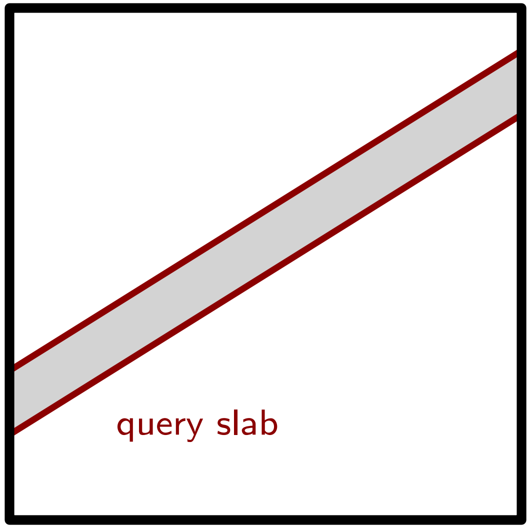
Problem:

- Input: n points
- Goal: A data structure
- Query: A region inside the unit square
- Output: All the points inside the region

Geometric Range Reporting: GRR

A Framework Theorem

Unit square in 2D



Problem:

- Input: n points
- Goal: A data structure
- Query: A region inside the unit square
- Output: All the points inside the region

Geometric Range Reporting: GRR

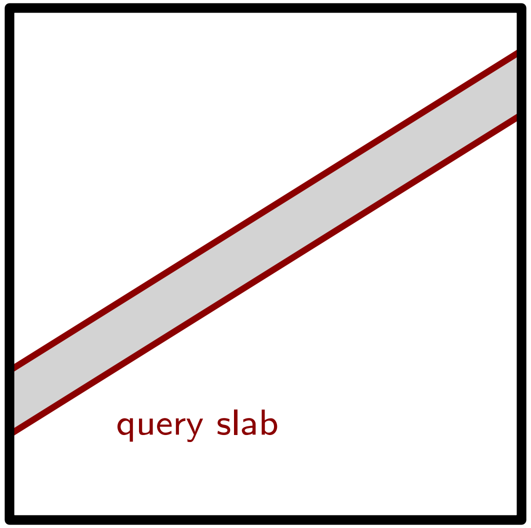
Framework Theorem:

(i) Assume we have a data structure that solves our GRR:

1. Given any input of n points
2. stores them using $S(n)$ space, s.t., it
3. answers any query in $O(Q(n) + k)$ time.

A Framework Theorem

Unit square in 2D



Problem:

- Input: n points
- Goal: A data structure
- Query: A region inside the unit square
- Output: All the points inside the region

Geometric Range Reporting: GRR

Framework Theorem:
(i) Assume we have a data structure that solves our GRR:

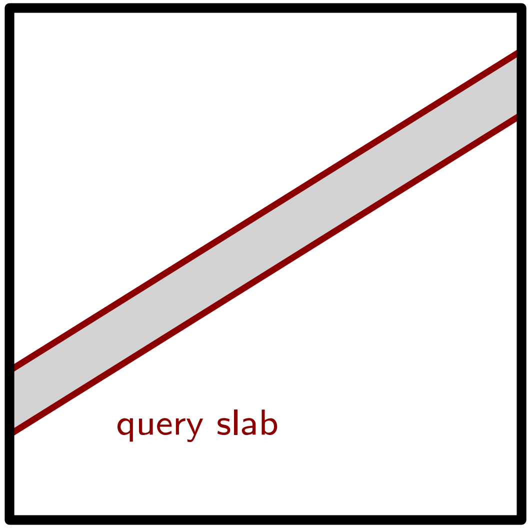
1. Given any input of n points
2. stores them using $S(n)$ space, s.t., it
3. answers any query in $O(Q(n) + k)$ time.

Assume we can build:

- n points
- m query regions, $r_1, \ldots, r_m$
- (Cond. I) Every r_i contains $\Omega(Q(n))$ points
- (Cond. II) Any α queries contain at most β points

A Framework Theorem

Unit square in 2D



Problem:

- Input: n points
- Goal: A data structure
- Query: A region inside the unit square
- Output: All the points inside the region

Geometric Range Reporting: GRR

Framework Theorem:
(i) Assume we have a data structure that solves our GRR:

1. Given any input of n points
2. stores them using $S(n)$ space, s.t., it
3. answers any query in $O(Q(n) + k)$ time.

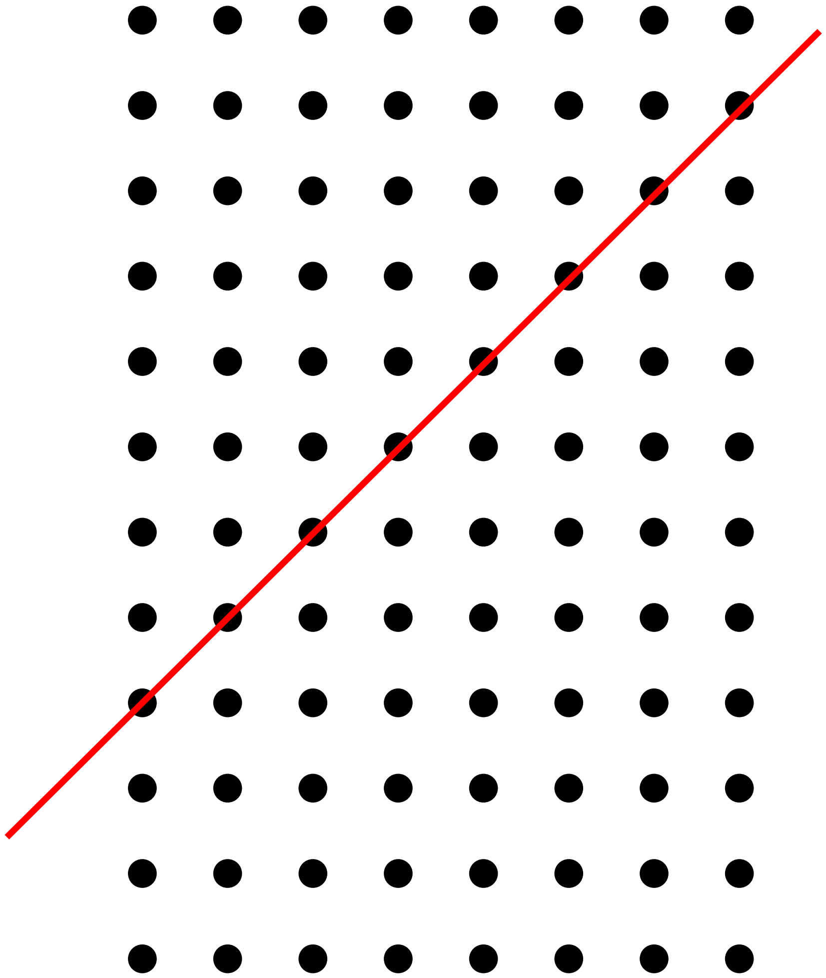
Assume we can build:

- n points
- m query regions, $r_1, \ldots, r_m$
- (Cond. I) Every r_i contains $\Omega(Q(n))$ points
- (Cond. II) Any α queries contain at most β points

$$S(n) = \Omega\left(\frac{\sum |r_i|}{\alpha 2^{O(\beta)}}\right)$$

A Discrete Geometry View

- Input: n points
- Query: lines



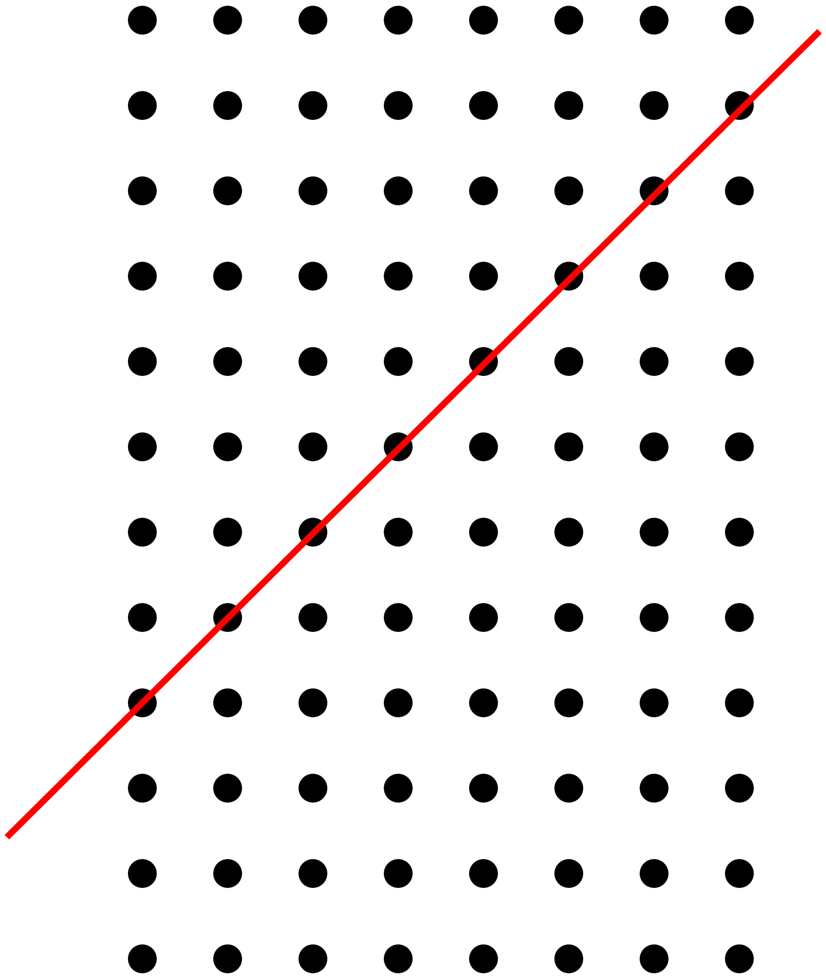
A Discrete Geometry View

- Input: n points
- Query: lines

Build:

- n points
- (a lot of) m query regions, $r_1, \ldots, r_m$
- (Cond. I) Every r_i contains $\Omega(Q(n))$ points
- (Cond. II) Any α queries contain at most β points.

$$S(n) = \Omega\left(\frac{\sum |r_i|}{\alpha 2^{O(\beta)}}\right)$$



A Discrete Geometry View

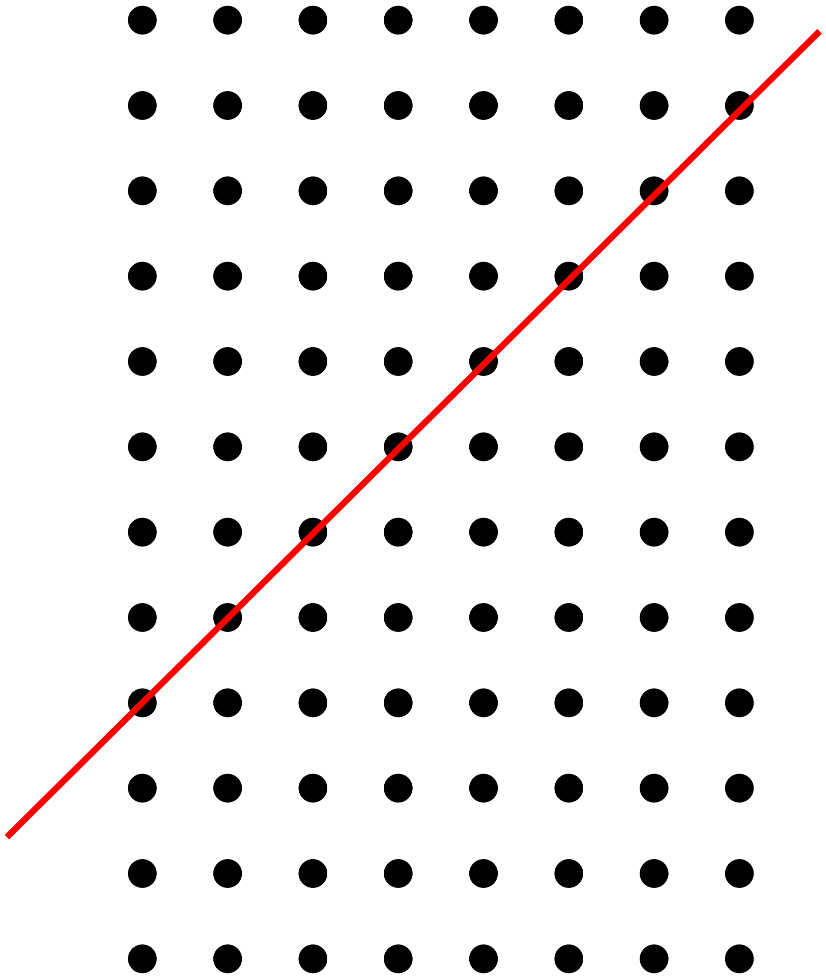
- Input: n points
- Query: lines

Build:

- n points
- (a lot of) m query regions, $r_1, \ldots, r_m$
- (Cond. I) Every r_i contains $\Omega(Q(n))$ points
- (Cond. II) Any α queries contain at most β points.

$$S(n) = \Omega\left(\frac{\sum |r_i|}{\alpha 2^{O(\beta)}}\right)$$

- Every line is $Q(n)$-rich
- No $K_{\alpha,\beta}$ in incidence graph
- Lower bound: $S(n) \gg \frac{\text{\# of incidences}}{\alpha 2^{O(\beta)}}$



A Discrete Geometry View

- Input: n points
- Query: lines

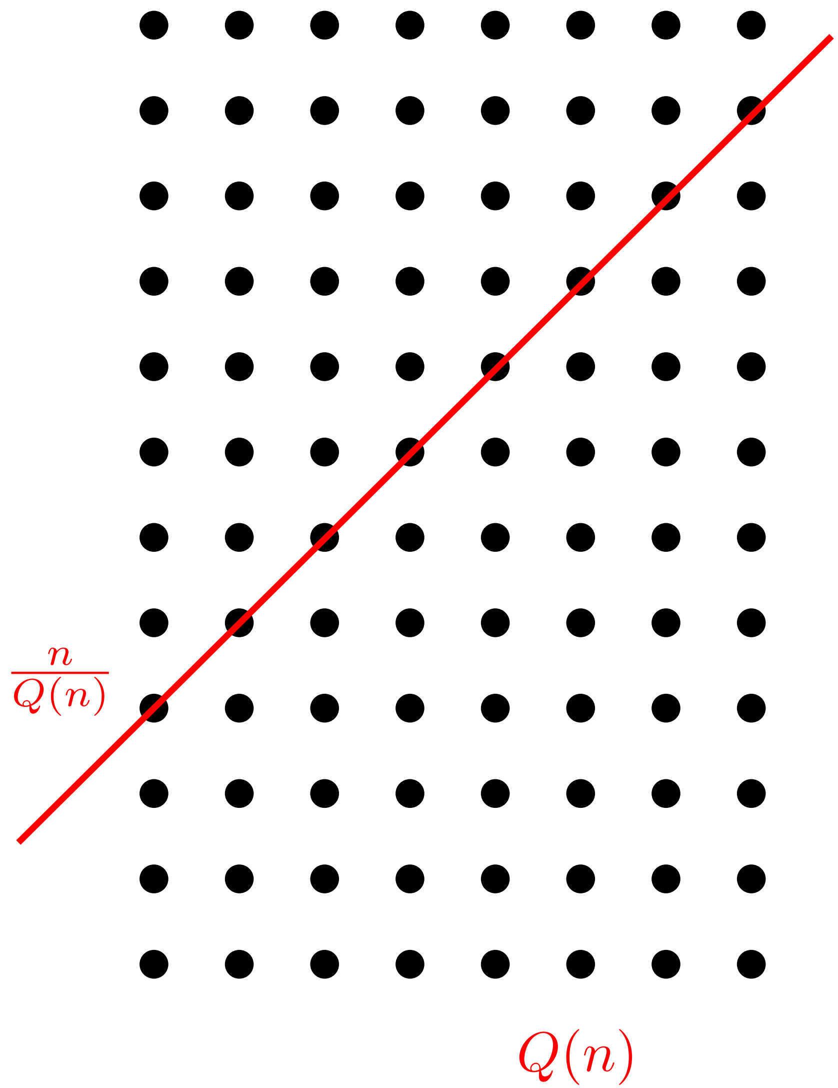
Build:

- n points
- (a lot of) m query regions, $r_1, \ldots, r_m$
- (Cond. I) Every r_i contains $\Omega(Q(n))$ points
- (Cond. II) Any α queries contain at most β points.

$$S(n) = \Omega\left(\frac{\sum |r_i|}{\alpha 2^{O(\beta)}}\right)$$

- Every line is $Q(n)$-rich
- No $K_{\alpha,\beta}$ in incidence graph
- Lower bound: $S(n) \gg \frac{\#\text{ of incidences}}{\alpha 2^{O(\beta)}}$

Well-known construction:



A Discrete Geometry View

- Input: n points
- Query: lines

Build:

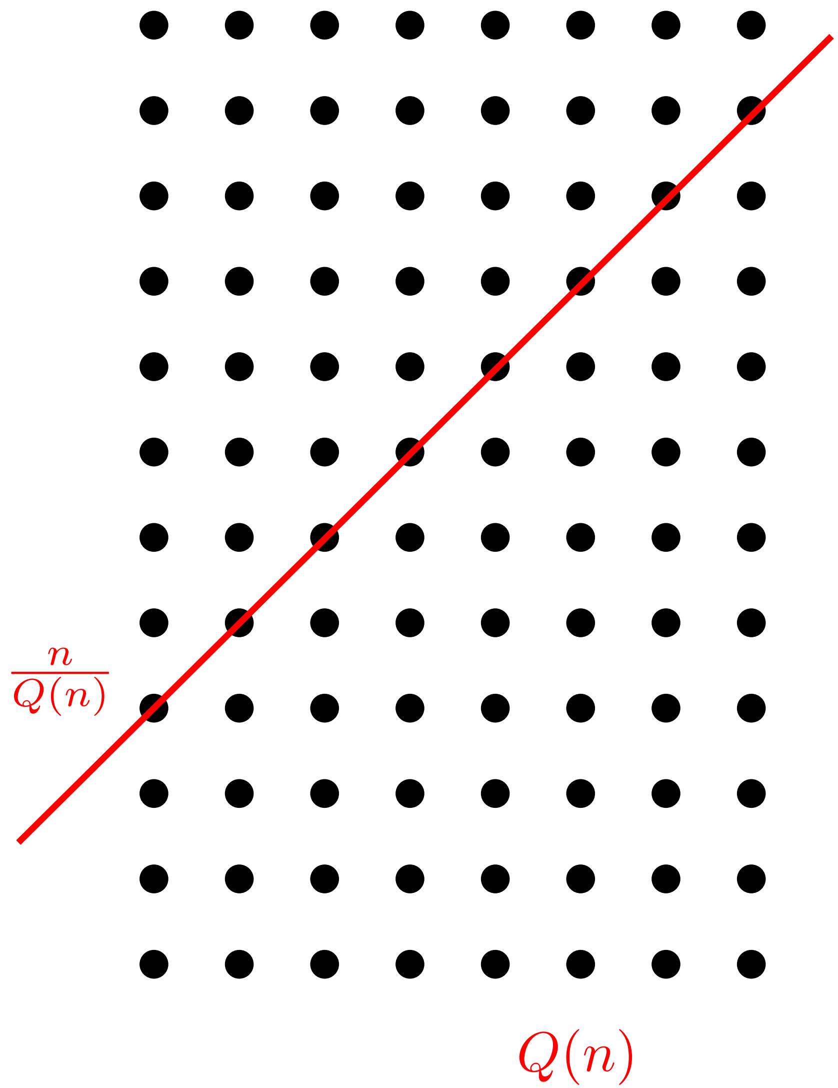
- n points
- (a lot of) m query regions, $r_1, \ldots, r_m$
- (Cond. I) Every r_i contains $\Omega(Q(n))$ points
- (Cond. II) Any α queries contain at most β points.

$$S(n) = \Omega\left(\frac{\sum |r_i|}{\alpha 2^{O(\beta)}}\right)$$

- Every line is $Q(n)$-rich
- No $K_{\alpha,\beta}$ in incidence graph
- Lower bound: $S(n) \gg \frac{\#\text{ of incidences}}{\alpha 2^{O(\beta)}}$

Well-known construction:

Slopes of 1,2,3,. . . , $\frac{n}{Q^2(n)}$



A Discrete Geometry View

- Input: n points
- Query: lines

Build:

- n points
- (a lot of) m query regions, $r_1, \ldots, r_m$
- (Cond. I) Every r_i contains $\Omega(Q(n))$ points
- (Cond. II) Any α queries contain at most β points.

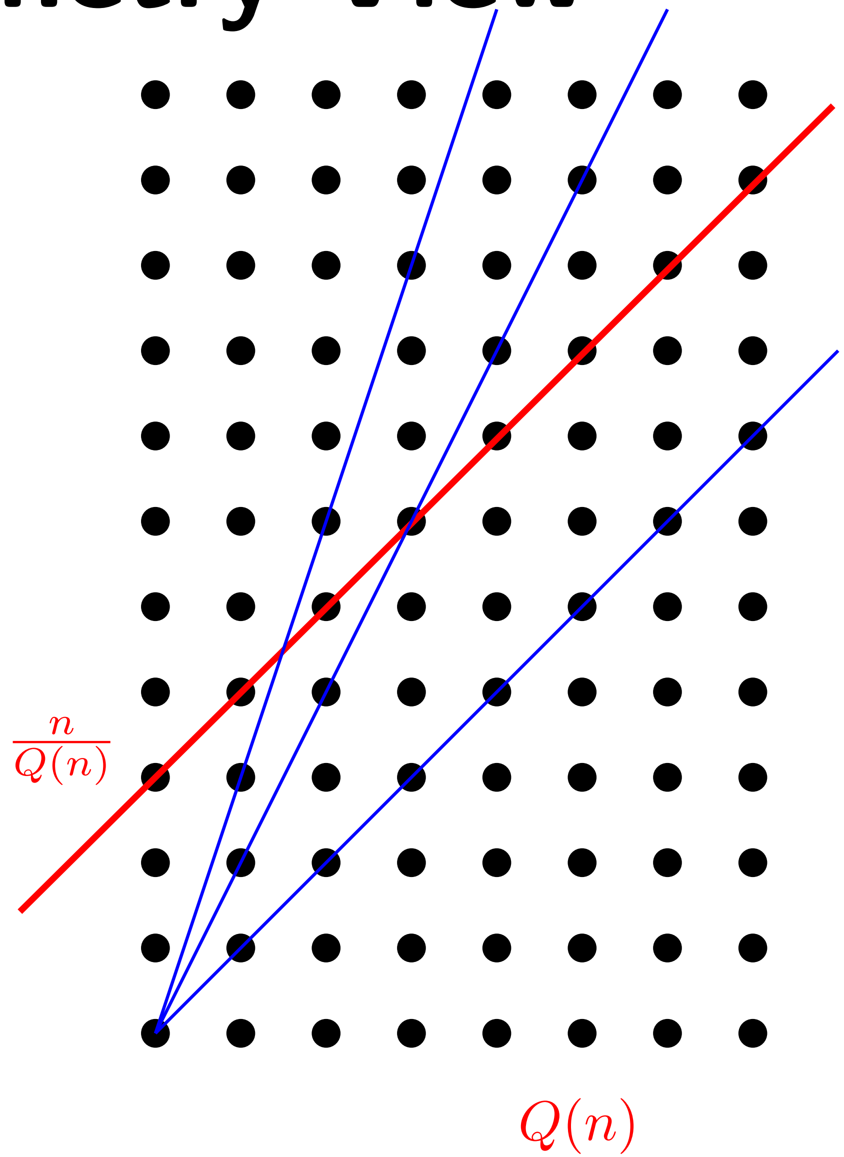
$$S(n) = \Omega\left(\frac{\sum |r_i|}{\alpha 2^{O(\beta)}}\right)$$

- Every line is $Q(n)$-rich
- No $K_{\alpha,\beta}$ in incidence graph
- Lower bound: $S(n) \gg \frac{\#\text{ of incidences}}{\alpha 2^{O(\beta)}}$

Well-known construction:

Slopes of 1,2,3,$\ldots$, $\frac{n}{Q^2(n)}$

$\Omega\left(\frac{n}{Q(n)}\right)$ values for Y-intersepts



A Discrete Geometry View

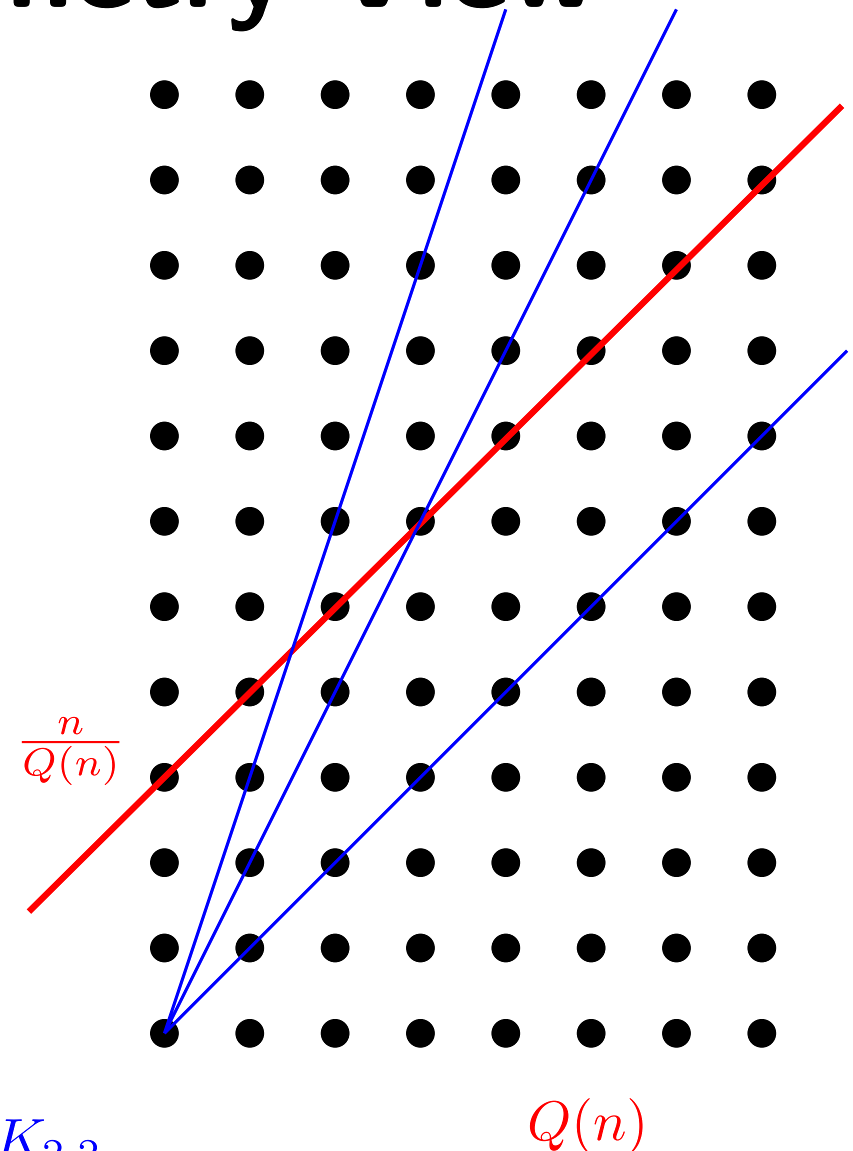
- Input: n points
- Query: lines

Build:

- n points
- (a lot of) m query regions, $r_1, \ldots, r_m$
- (Cond. I) Every r_i contains $\Omega(Q(n))$ points
- (Cond. II) Any α queries contain at most β points.

$$S(n) = \Omega\left(\frac{\sum |r_i|}{\alpha 2^{O(\beta)}}\right)$$

- Every line is $Q(n)$-rich
- No $K_{\alpha,\beta}$ in incidence graph
- Lower bound: $S(n) \gg \frac{\#\text{ of incidences}}{\alpha 2^{O(\beta)}}$



Well-known construction:

Slopes of 1,2,3,$\ldots$, $\frac{n}{Q^2(n)}$

$\Omega\left(\frac{n}{Q(n)}\right)$ values for Y-intersepts

No $K_{2,2}$

$I = \frac{n^2}{Q^2(n)}$ space lower bound

Optimal

A Discrete Geometry View

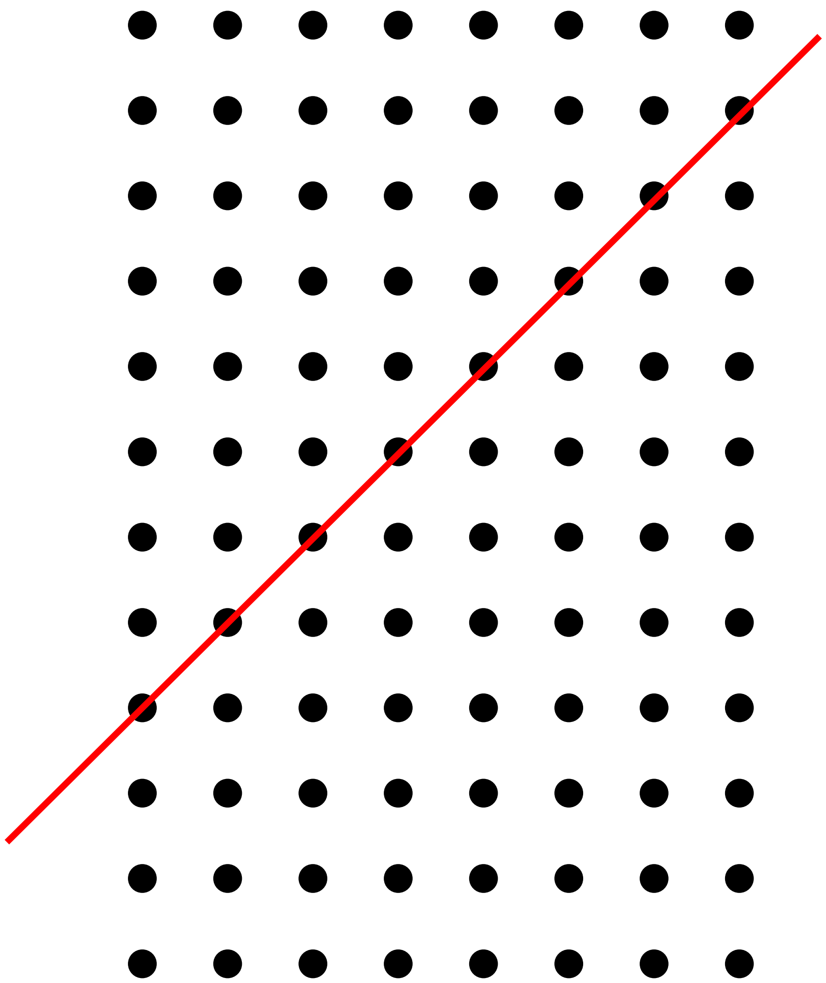
- Input: n points
- Query: lines

Build:

- n points
- (a lot of) m query regions, $r_1, \ldots, r_m$
- (Cond. I) Every r_i contains $\Omega(Q(n))$ points
- (Cond. II) Any α queries contain at most β points.

$$S(n) = \Omega\left(\frac{\sum |r_i|}{\alpha 2^{O(\beta)}}\right)$$

- Every line is $Q(n)$-rich
- No $K_{\alpha,\beta}$ in incidence graph
- Lower bound: $S(n) \gg \frac{\text{\# of incidences}}{\alpha 2^{O(\beta)}}$



Afshani, Cheng, *SOSA'23*:

$Q(n) \gg \left(\frac{n^2}{S(n)}\right)^{\frac{d-1}{d}}$

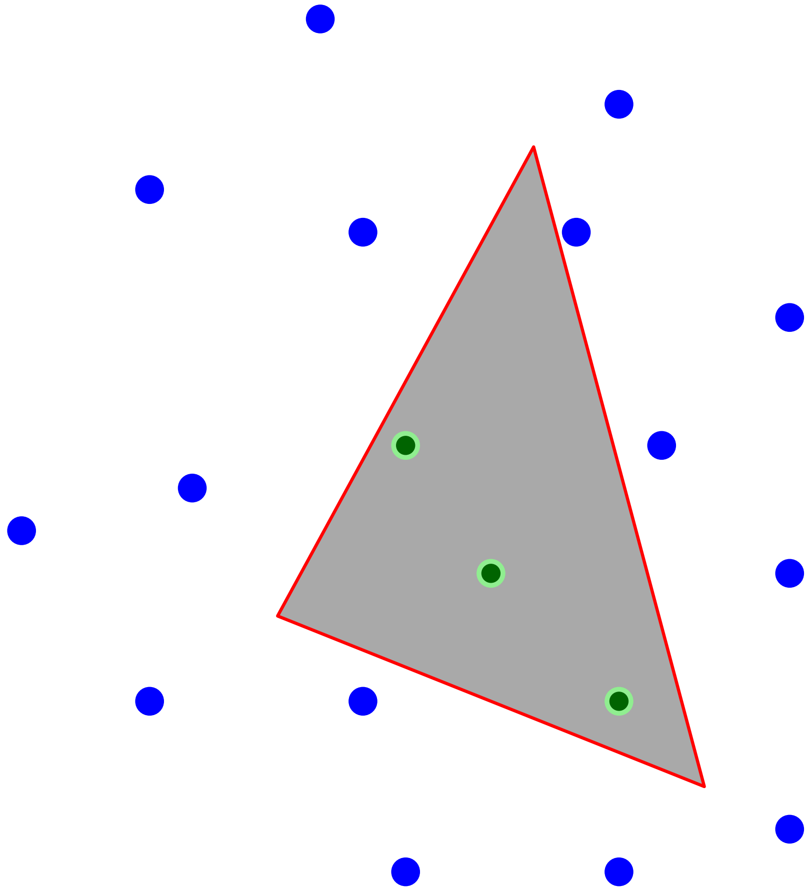
For $S(n) = O(n) \Rightarrow Q(n) = \Omega(n^{1-1/d})$

(only **tight** LB for $d > 2$)

Semialgebraic Range Reporting

Input:

- n points in $\mathbb{R}^d$.
- Store in a DS
- Given a range R
  - list them.



n space, $n^{1-1/d}$ query time (low space)

n^d space, $\log^{d-1} n$ query time (fast query)

Semialgebraic Range Reporting

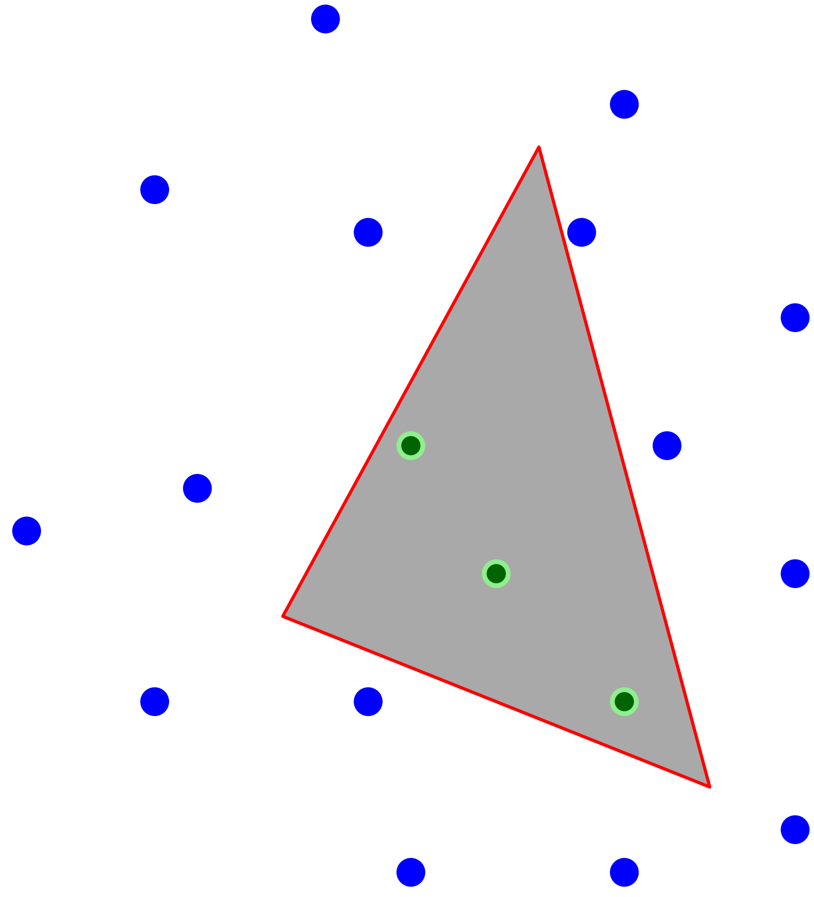
Input:

- n points in $\mathbb{R}^d$.
- Store in a DS
- Given a range R
  - list them.

n space, $n^{1-1/d}$ query time (low space)

n^d space, $\log^{d-1} n$ query time (fast query)

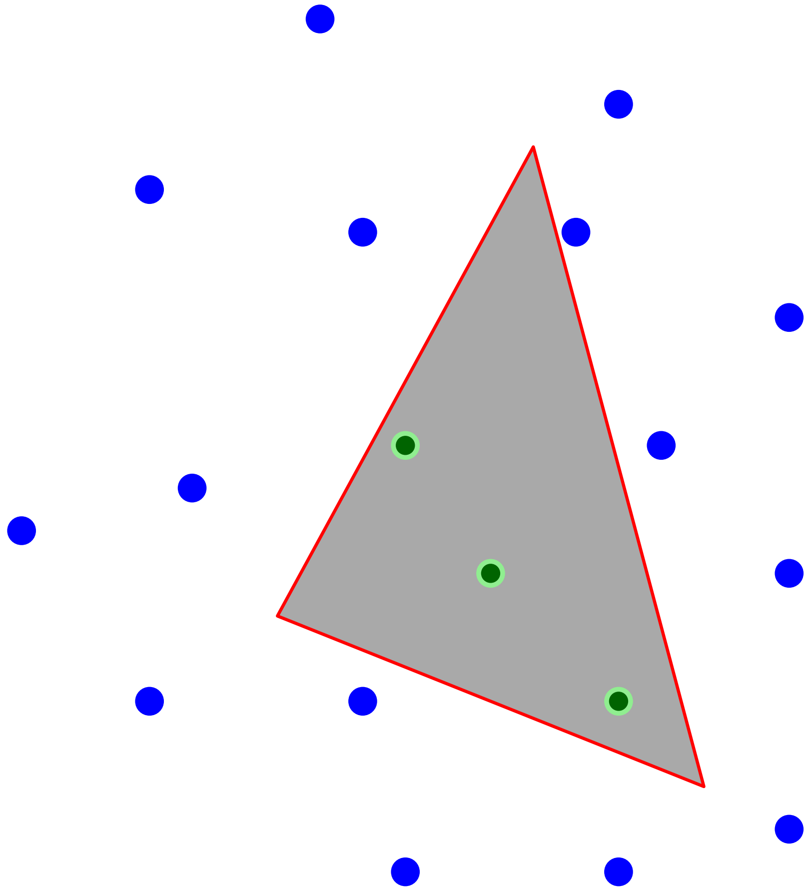
$$S(n) = \frac{n^d}{Q^d(n)}$$



Semialgebraic Range Reporting

Input:

- n points in $\mathbb{R}^d$.
- Store in a DS
- Given a range R
  - list them.



n space, $n^{1-1/d}$ query time (low space)

n^d space, $\log^{d-1} n$ query time (fast query)

$$S(n) = \frac{n^d}{Q^d(n)}$$

Semialgebraic Range Reporting

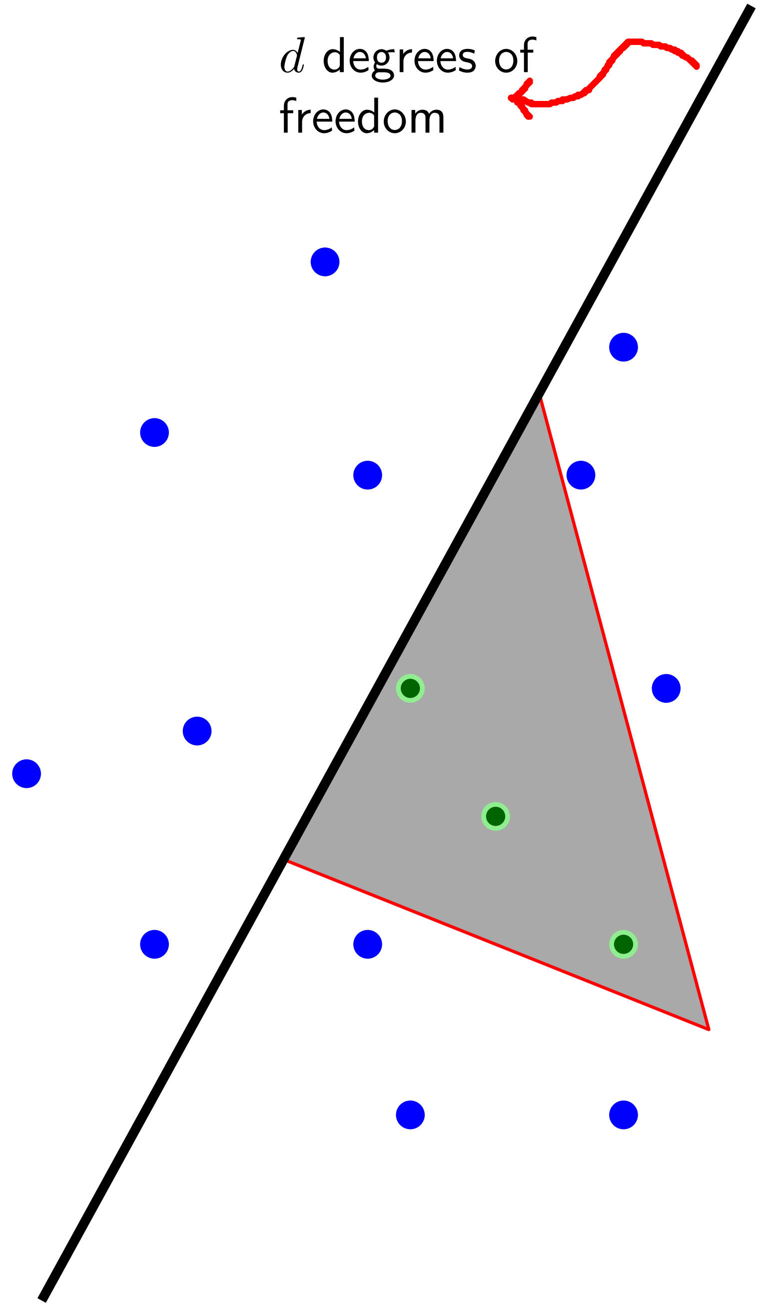
Input:

- n points in $\mathbb{R}^d$.
- Store in a DS
- Given a range R
  - list them.

n space, $n^{1-1/d}$ query time (low space)

n^d space, $\log^{d-1} n$ query time (fast query)

$$S(n) = \frac{n^d}{Q^d(n)}$$

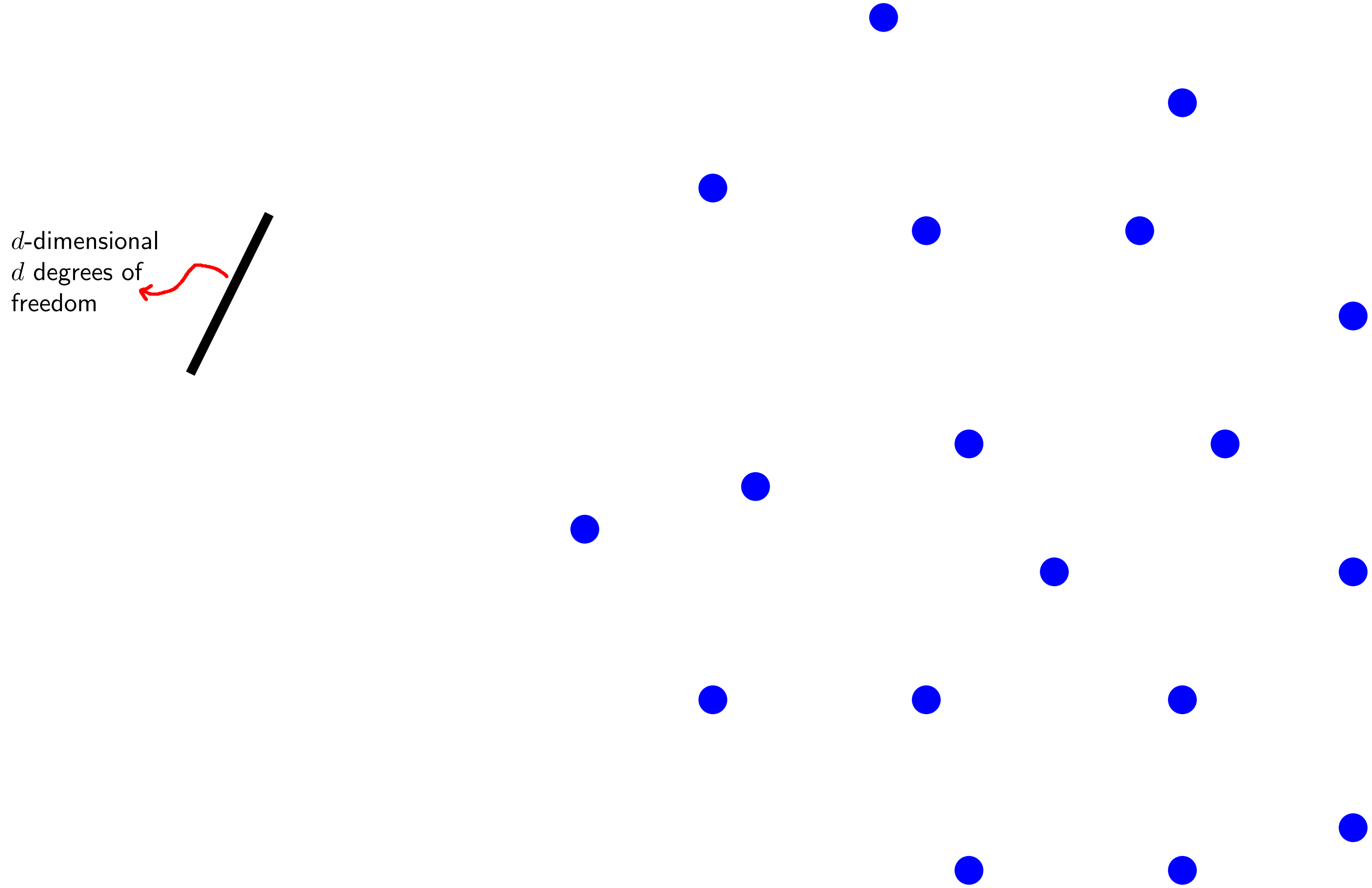


Semialgebraic Range Reporting

Input:

- n points in $\mathbb{R}^d$.
- Store in a DS
- Given a range R
  - list them.

$$S(n) = \frac{n^d}{Q^d(n)}$$



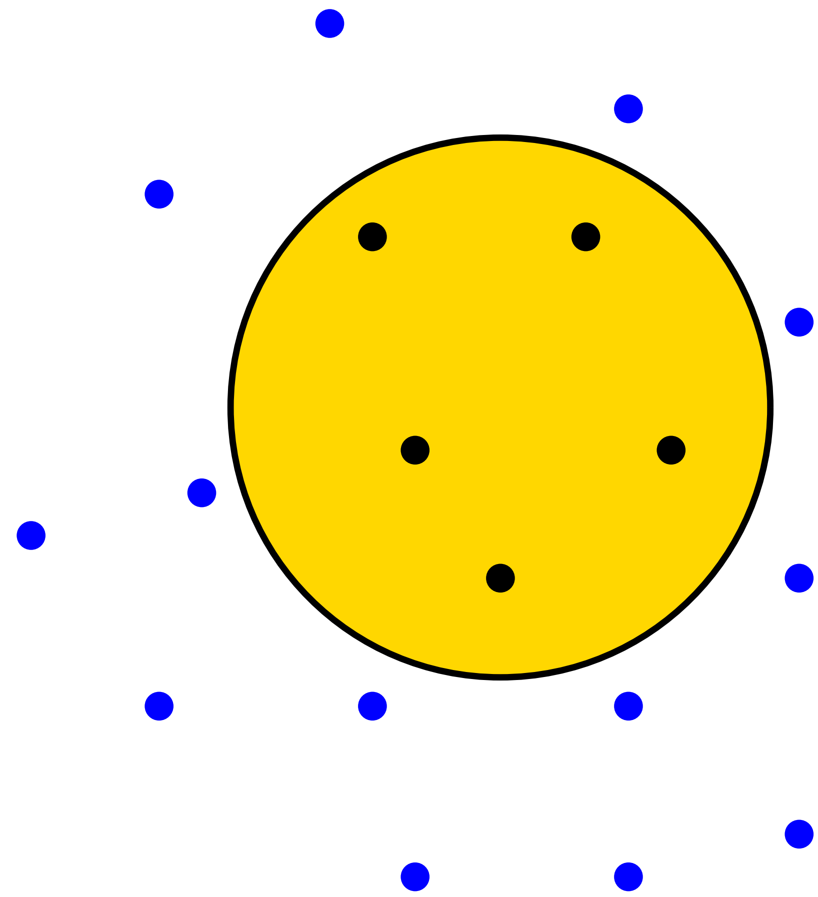
Semialgebraic Range Reporting

Input:

- n points in $\mathbb{R}^d$.
- Store in a DS
- Given a range R
  - list them.

$$S(n) = \frac{n^d}{Q^d(n)}$$

d-dimensional
d degrees of freedom



Semialgebraic Range Reporting

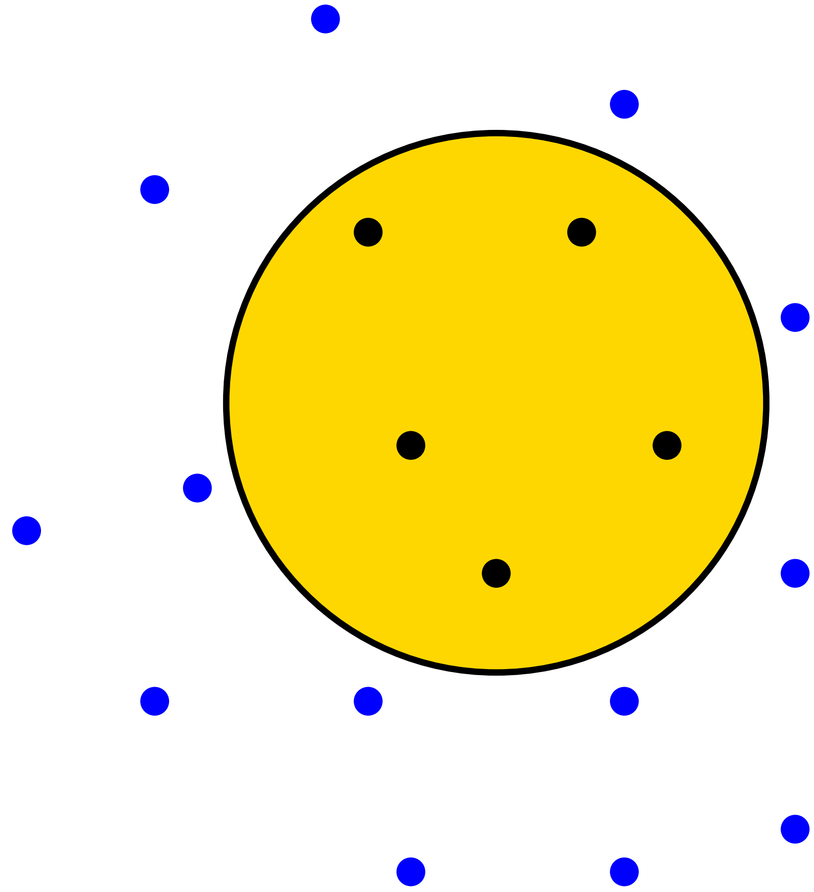
Input:

- n points in $\mathbb{R}^d$.
- Store in a DS
- Given a range R
  - list them.

$$S(n) = \frac{n^d}{Q^d(n)}$$

d-dimensional
d degrees of freedom

Find all (x_i, y_i) s.t.,
$(x_i - a)^2 + (y_i - b)^2 \leq r^2$



Semialgebraic Range Reporting

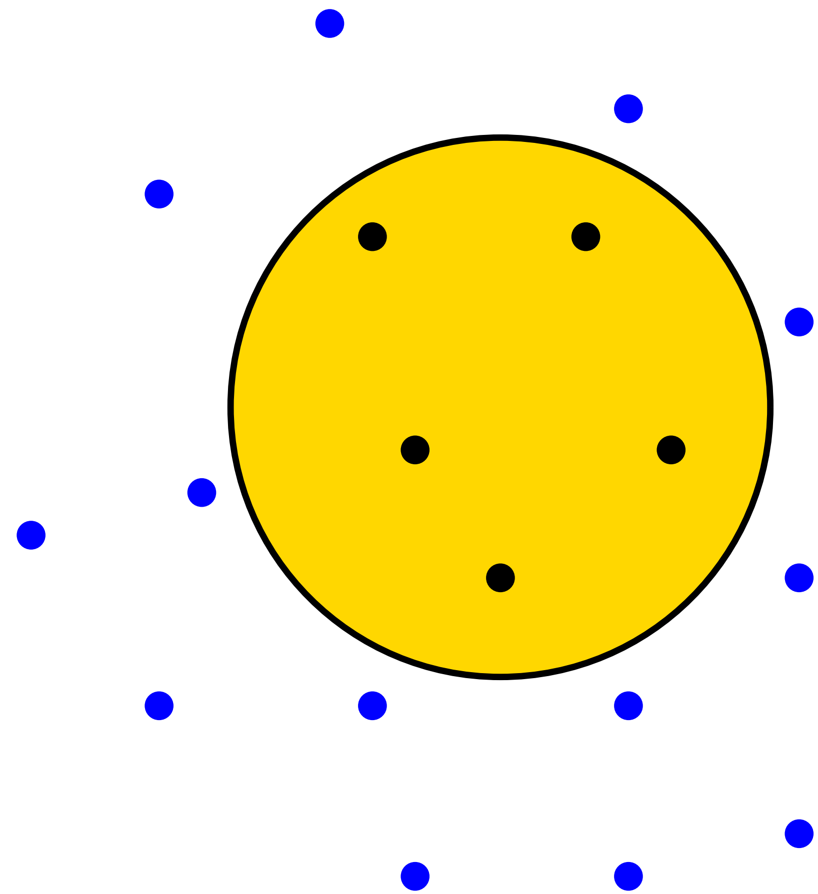
Input:

- n points in $\mathbb{R}^d$.
- Store in a DS
- Given a range R
  - list them.

$$S(n) = \frac{n^d}{Q^d(n)}$$

d-dimensional
d degrees of freedom

Find all (x_i, y_i) s.t.,
$(x_i - a)^2 + (y_i - b)^2 \leq r^2$



Semialgebraic Range Reporting

Input:
- n points in $\mathbb{R}^d$.
- Store in a DS
- Given a range R
  - list them.

$$S(n) = \frac{n^d}{Q^d(n)}$$

d-dimensional
d degrees of
freedom

Find all (x_i, y_i) s.t.,

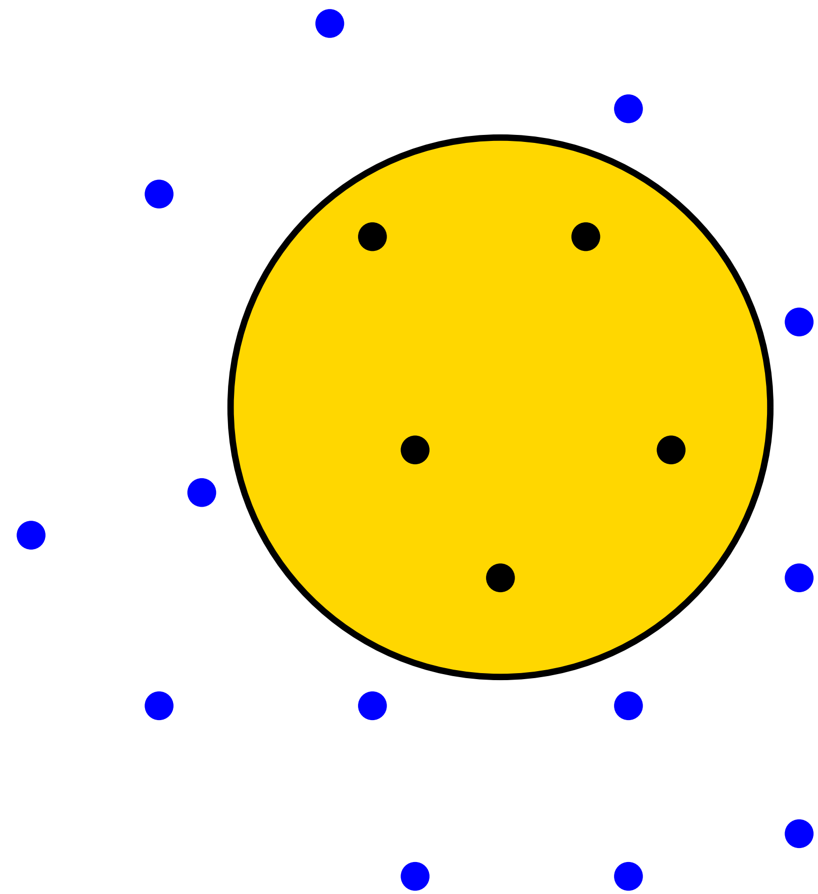
$$(x_i - a)^2 + (y_i - b)^2 \leq r^2$$

$$x_i^2 - 2ax_i + a^2 + y_i^2 - 2by_i + b^2 \leq r^2$$

$$z_i - 2ax_i + a^2 + -2by_i + b^2 \leq r^2$$

$$z_i \leq 2ax_i + 2by_i + r^2 - a^2 - b^2$$

Point $(x_i, y_i, x_i^2 + y_i^2)$ below halfspace
$H(a, b, r) : Z \leq 2aX + 2bY + r^2 - a^2 - b^2$

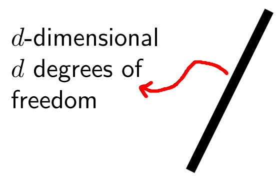


Semialgebraic Range Reporting

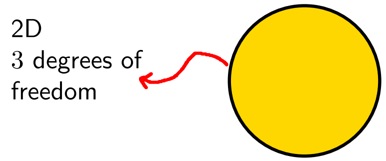
Input:

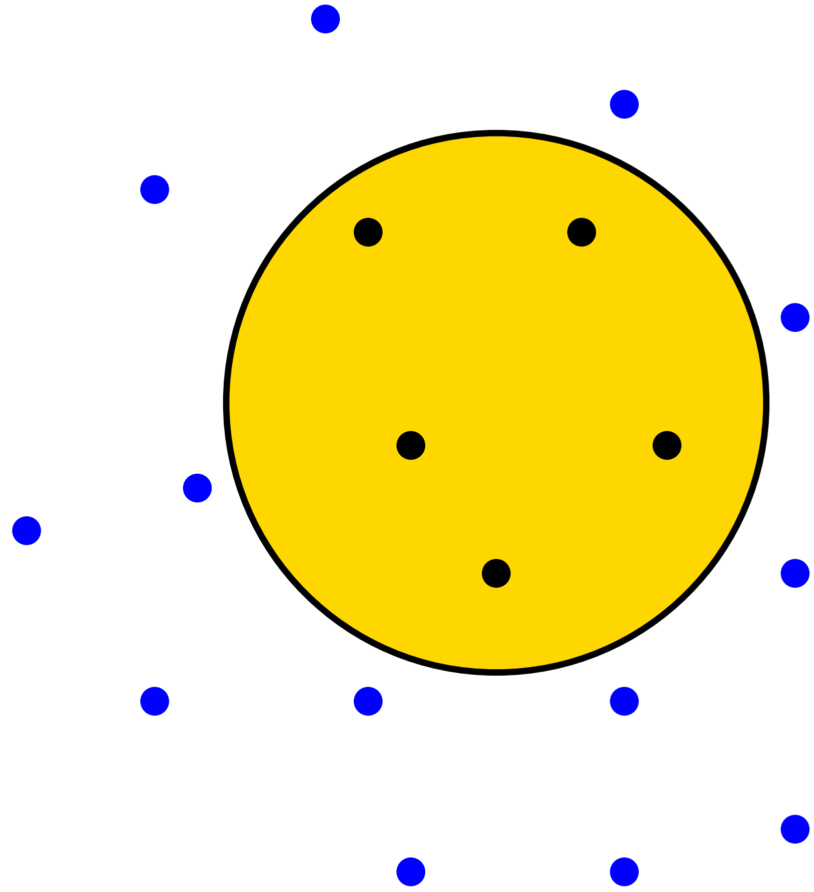
- n points in $\mathbb{R}^d$.
- Store in a DS
- Given a range R
  - list them.

$$S(n) = \frac{n^d}{Q^d(n)}$$



$$S(n) = \frac{n^3}{Q^3(n)}$$



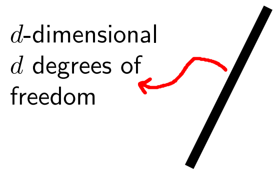


Semialgebraic Range Reporting

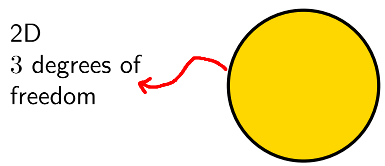
Input:

- n points in $\mathbb{R}^d$.
- Store in a DS
- Given a range R
  - list them.

$$S(n) = \frac{n^d}{Q^d(n)}$$

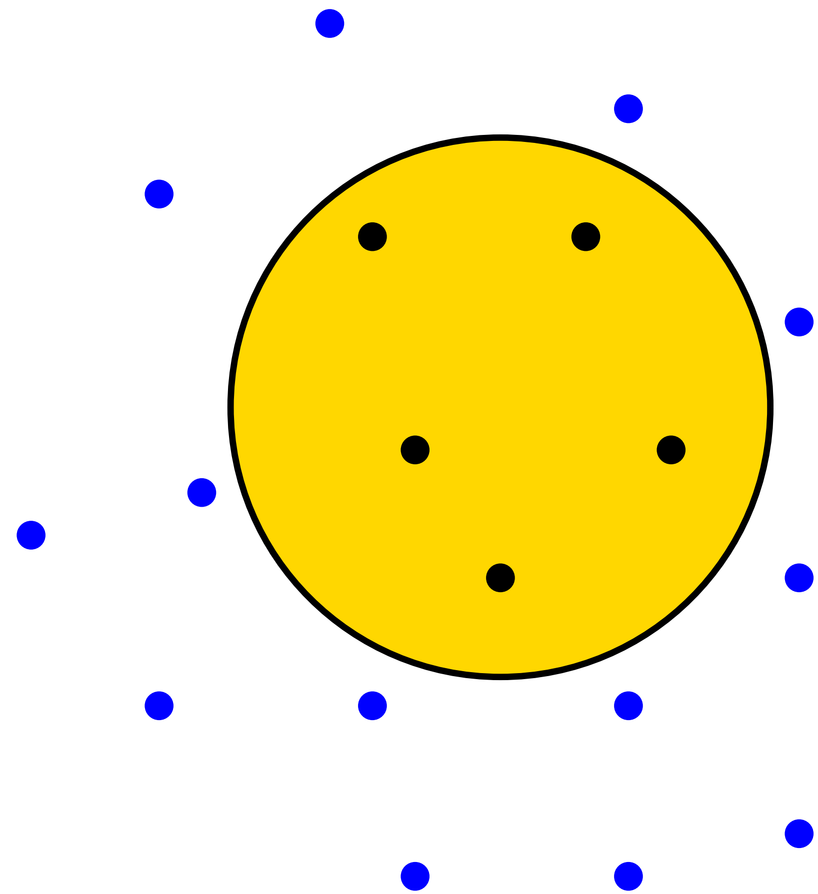


$$S(n) = \frac{n^3}{Q^3(n)}$$



n space, $n^{1-1/3} = n^{2/3}$ query time (low space)

n^3 space, $\log^2 n$ query time (fast query)

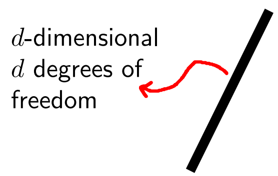


Semialgebraic Range Reporting

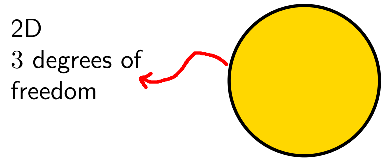
Input:

- n points in $\mathbb{R}^d$.
- Store in a DS
- Given a range R
  - list them.

$$S(n) = \frac{n^d}{Q^d(n)}$$



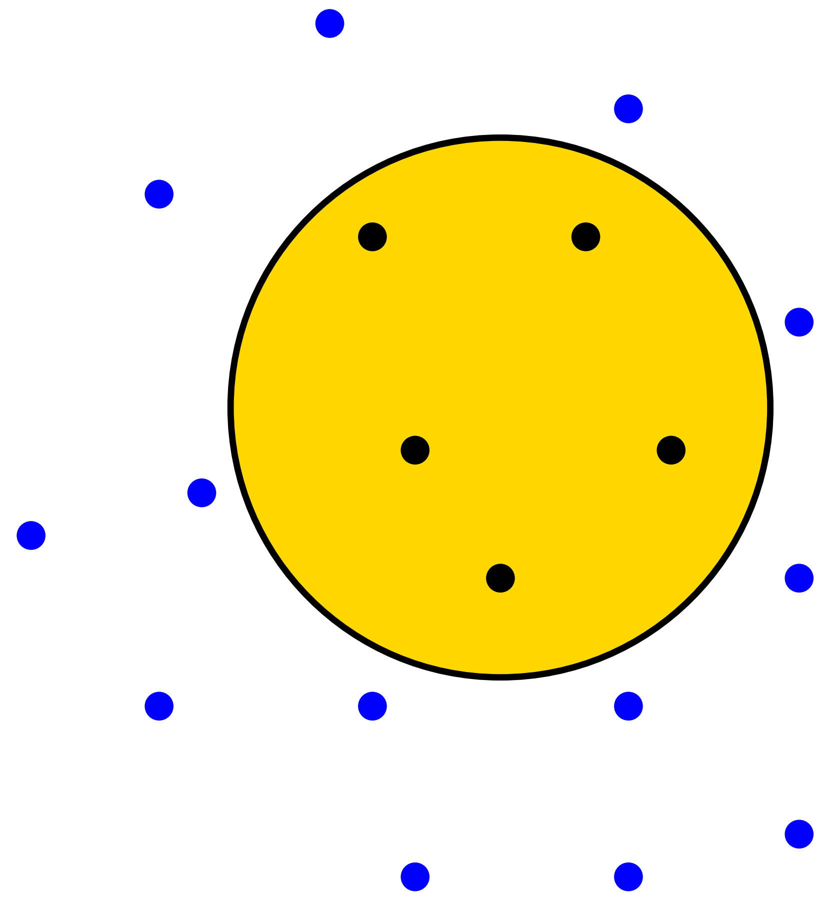
$$S(n) = \frac{n^3}{Q^3(n)}$$



n space, $n^{1-1/2} = n^{1/2}$ query time (low space)

n^3 space, $\log^2 n$ query time (fast query)

The polynomial
Method



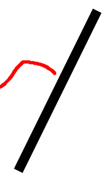
Semialgebraic Range Reporting

Input:

- n points in $\mathbb{R}^d$.
- Store in a DS
- Given a range R
  - list them.

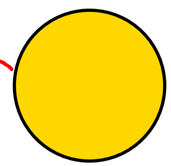
$$S(n) = \frac{n^d}{Q^d(n)}$$

d-dimensional
d degrees of freedom



$$S(n) = \frac{n^3}{Q^3(n)}$$

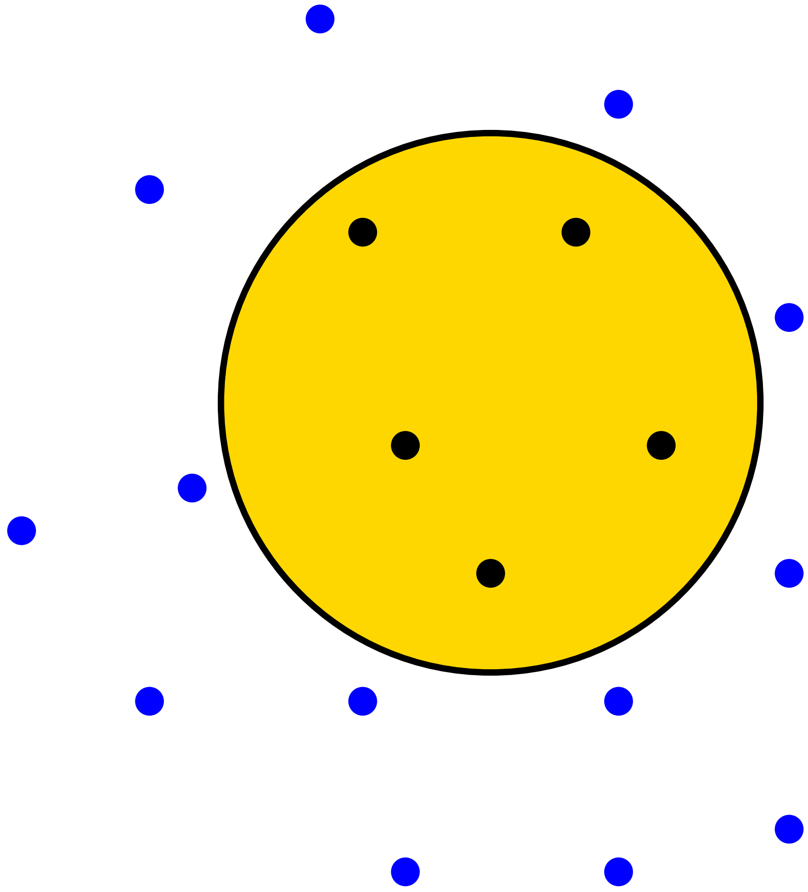
2D
3 degrees of freedom



n space, $n^{1-1/2} = n^{1/2}$ query time (low space)

n^3 space, $\log^2 n$ query time (fast query)

The polynomial Method



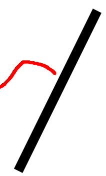
Semialgebraic Range Reporting

Input:

- n points in $\mathbb{R}^d$.
- Store in a DS
- Given a range R
  - list them.

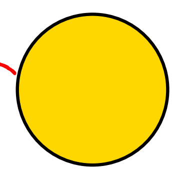
$$S(n) = \frac{n^d}{Q^d(n)}$$

d-dimensional
d degrees of
freedom



$$S(n) = \frac{n^3}{Q^3(n)}$$

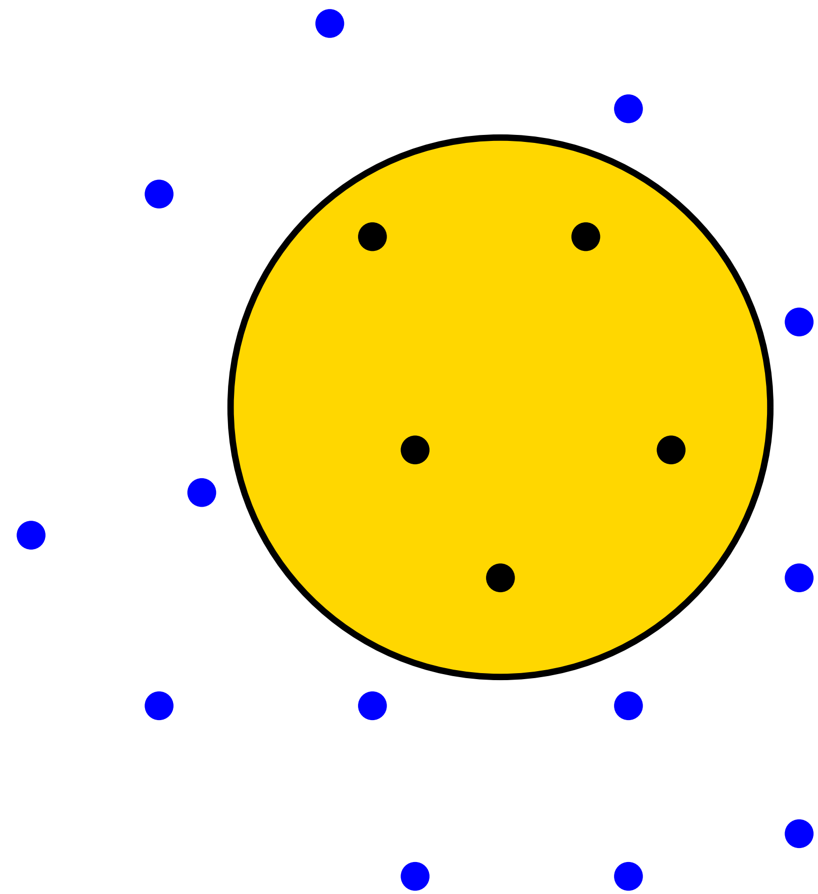
2D
3 degrees of
freedom



n space, $n^{1-1/2} = n^{1/2}$ query time (low space)

n^3 space, $\log^2 n$ query time (fast query)

The polynomial
Method



Current knowledge: $\frac{n^3}{Q^5(n)} \leq S(n) \leq \frac{n^3}{Q^4(n)}$

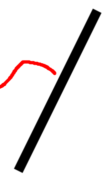
Semialgebraic Range Reporting

Input:

- n points in $\mathbb{R}^d$.
- Store in a DS
- Given a range R
  - list them.

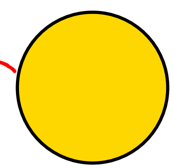
$$S(n) = \frac{n^d}{Q^d(n)}$$

d-dimensional
d degrees of freedom



$$S(n) = \frac{n^3}{Q^3(n)}$$

2D
3 degrees of freedom

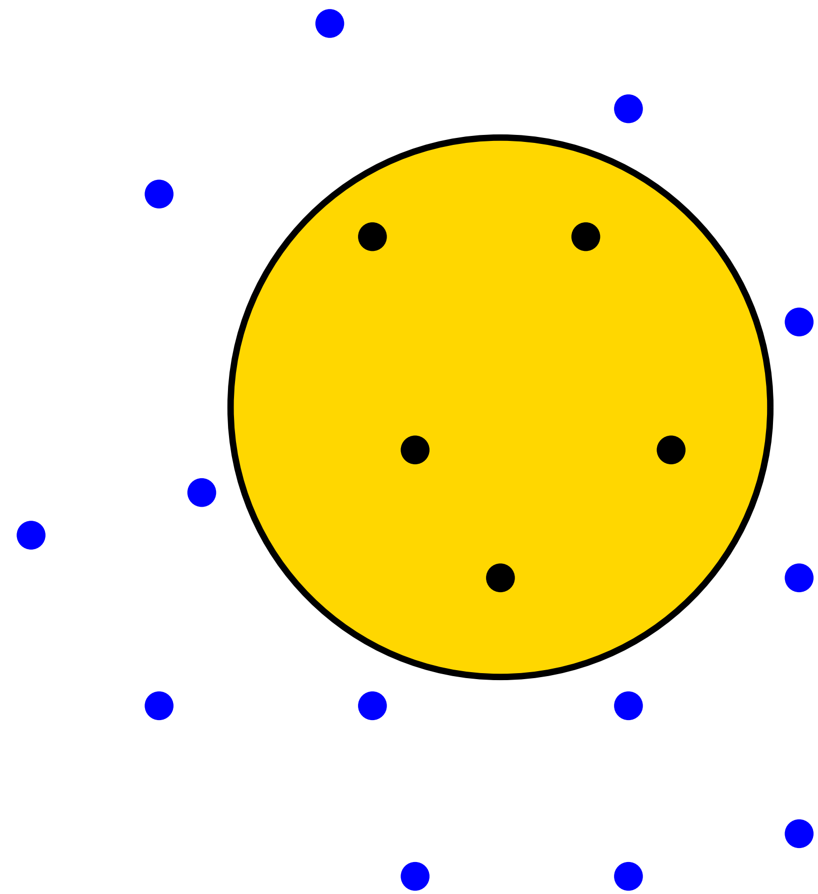


n space, $n^{1-1/2} = n^{1/2}$ query time (low space)

n^3 space, $\log^2 n$ query time (fast query)

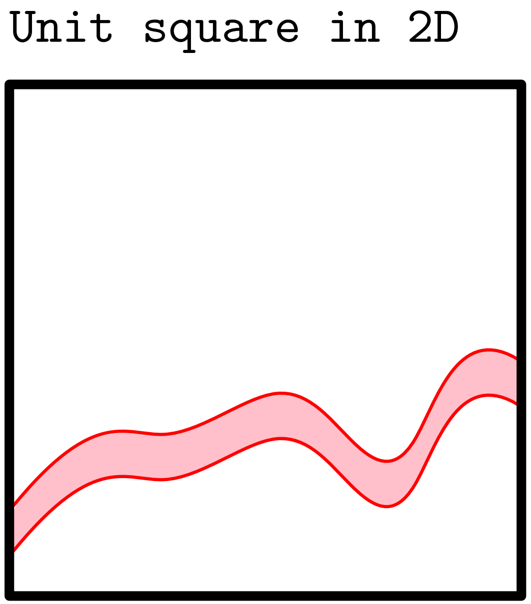
tight

The polynomial Method



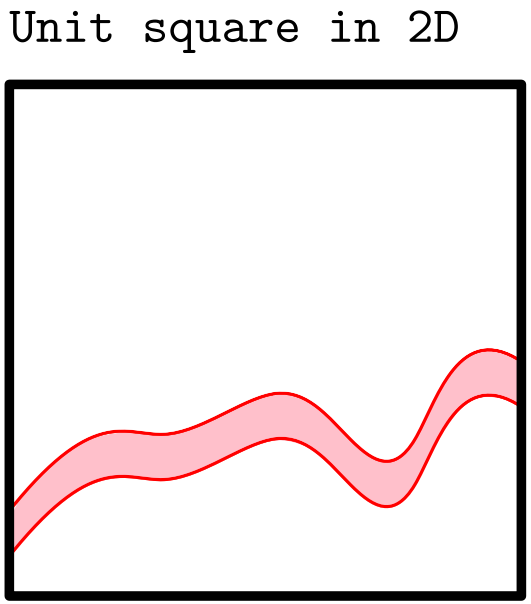
Current knowledge: $\frac{n^3}{Q^5(n)} \leq S(n) \leq \frac{n^3}{Q^4(n)}$

Fast Query Lower Bound: The General Approach



- Input: n uniformly random points
- Query: $-w \leq P(x,y) \leq w$
- List the points in the query
- Goal: Lower bound for polylog $Q(n)$; $Q(n) = \tilde{O}(1)$
- Space Lower Bound: roughly n^{β}
- β: Degrees of freedom

Fast Query Lower Bound: The General Approach

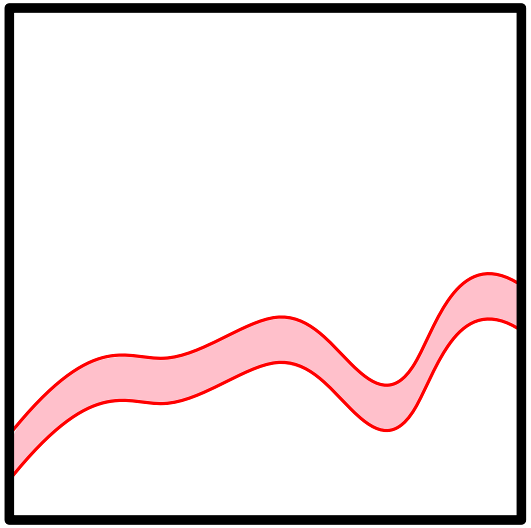


- Input: n uniformly random points
- Query: $-w \leq P(x, y) \leq w$
- List the points in the query
- Goal: Lower bound for polylog $Q(n)$; $Q(n) = \tilde{O}(1)$
- Space Lower Bound: roughly n^{β}
- β: Degrees of freedom

How to:

- Create n^{β} polynomials $P_i(x, y)$
- Area of $-w \leq P(x, y) \leq w$ is $\Theta(w)$
- $w \approx \frac{Q(n)}{n} = \tilde{O}(1)$: Each region is "$Q(n)$-rich"

Fast Query Lower Bound: The General Approach



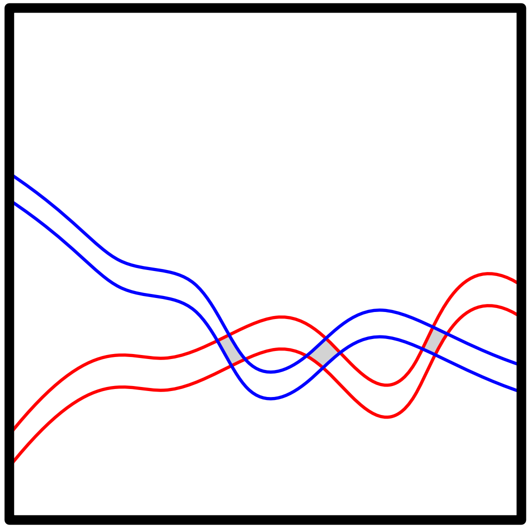
- Input: n uniformly random points
- Query: $-w \leq P(x, y) \leq w$
- List the points in the query
- Goal: Lower bound for polylog $Q(n)$; $Q(n) = \tilde{O}(1)$
- Space Lower Bound: roughly n^β
- β: Degrees of freedom

How to:

- Create n^β polynomials $P_i(x, y)$
- Area of $-w \leq P(x, y) \leq w$ is $\Theta(w)$
- $w \approx \frac{Q(n)}{n} = \tilde{O}(1)$: Each region is "$Q(n)$-rich"
- **(main challenge)** Intersection of two regions: $\ll \frac{1}{n}$

Fast Query Lower Bound: The General Approach

`Unit square in 2D`



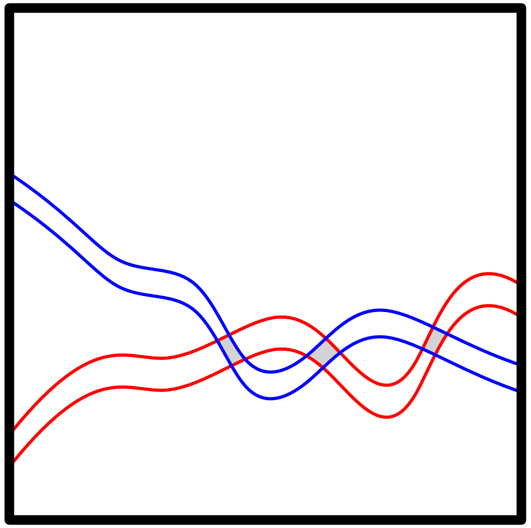
- Input: n uniformly random points
- Query: $-w \leq P(x, y) \leq w$
- List the points in the query
- Goal: Lower bound for polylog $Q(n)$; $Q(n) = \tilde{O}(1)$
- Space Lower Bound: roughly n^β
- β: Degrees of freedom

How to:

- Create n^β polynomials $P_i(x, y)$
- Area of $-w \leq P(x, y) \leq w$ is $\Theta(w)$
- $w \approx \frac{Q(n)}{n} = \tilde{O}(1)$: Each region is "$Q(n)$-rich"
- **(main challenge)** Intersection of two regions: $\ll \frac{1}{n}$

Fast Query Lower Bound: The General Approach

Unit square in 2D



- Input: n uniformly random points
- Query: $-w \leq P(x,y) \leq w$
- List the points in the query
- Goal: Lower bound for polylog $Q(n)$; $Q(n) = \tilde{O}(1)$
- Space Lower Bound: roughly n^β
- β: Degrees of freedom

How to:

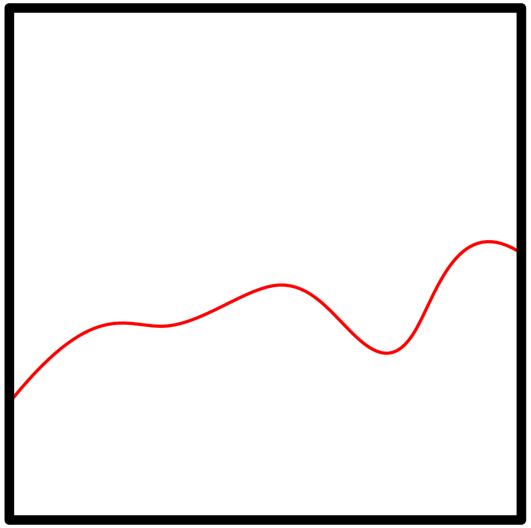
- Create n^β polynomials $P_i(x,y)$
- Area of $-w \leq P(x,y) \leq w$ is $\Theta(w)$
- $w \approx \frac{Q(n)}{n} = \tilde{O}(1)$: Each region is "$Q(n)$-rich"
- **(main challenge)** Intersection of two regions: $\ll \frac{1}{n}$

So far only one approach:

Create: $P_1(x,y), P_2(x,y), \ldots, P_M(x,y)$
Min. distance between coefficients is **large**
Prove it implies **(main challenge)**

The First Technique

Unit square in 2D



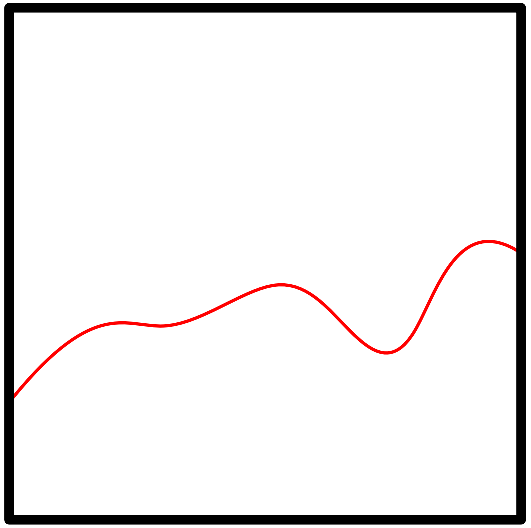
$P(x, y): \quad Y = \square X^{\Delta} + \square X^{\Delta-1} + \ldots + \square X + \square$

How to:

- Create $n^{\Delta+1}$ polynomials $P_i(x, y)$
- $-\frac{Q(n)}{n} \leq P(x, y) \leq \frac{Q(n)}{n}$
- Each region is "$Q(n)$-rich"
- **(main challenge)** Intersection of two regions: $\ll \frac{1}{n}$

The First Technique

Unit square in 2D



$P_j(x,y): \quad Y = \Box X^{\Delta} + \Box X^{\Delta-1} + \ldots + \Box X + \Box$

Distance $\frac{Q^{\Delta}(n)}{n}$ is enough to imply **(main challenge)**

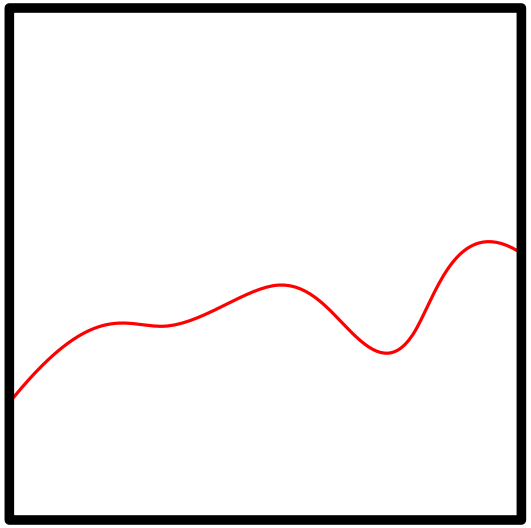
$P_i(x,y): \quad Y = \Box X^{\Delta} + \Box X^{\Delta-1} + \ldots + \Box X + \Box$

How to:

- Create $n^{\Delta+1}$ polynomials $P_i(x,y)$
- $-\frac{Q(n)}{n} \le P(x,y) \le \frac{Q(n)}{n}$
- Each region is "$Q(n)$-rich"
- **(main challenge)** Intersection of two regions: $\ll \frac{1}{n}$

The Second Technique

Unit square in 2D



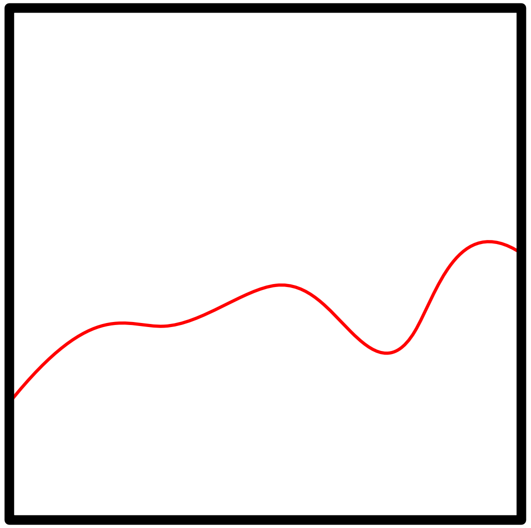
$P(x,y): \quad Y = X^{\Delta} + \square X^{\Delta-1}Y + \ldots + \square X^i Y^j + \ldots + \square Y + \square X + \square$

How to:

- Create $n^{\binom{\Delta+d}{d}}$ polynomials $P_i(x,y)$
- $-\frac{Q(n)}{n} \leq P(x,y) \leq \frac{Q(n)}{n}$
- Each region is "$Q(n)$-rich"
- **(main challenge)** Intersection of two regions: $\ll \frac{1}{n}$

The Second Technique

Unit square in 2D



$P_i(x,y):\quad Y = X^{\Delta} + \Box X^{\Delta-1}Y + \ldots + \Box X^i Y^j + \ldots + \Box Y + \Box X + \Box$

Distance $\frac{Q^{\Delta^2}(n)}{n}$ and small magnitude is enough to imply **(main challenge)**

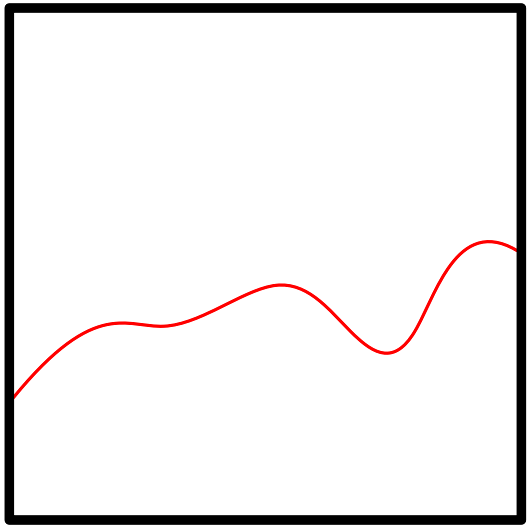
$P_j(x,y):\quad Y = X^{\Delta} + \Box X^{\Delta-1}Y + \ldots + \Box X^i Y^j + \ldots + \Box Y + \Box X + \Box$

How to:

- Create $n^{\binom{\Delta+d}{d}}$ polynomials $P_i(x,y)$
- $-\frac{Q(n)}{n} \leq P(x,y) \leq \frac{Q(n)}{n}$
- Each region is "$Q(n)$-rich"
- **(main challenge)** Intersection of two regions: $\ll \frac{1}{n}$

The Main Open Question

Unit square in 2D



$P(x,y): \quad 0 = \square X^{\Delta} + \square X^{\Delta-1} Y + \ldots + \square X^i Y^j + \ldots + \square Y + \square X + \square$

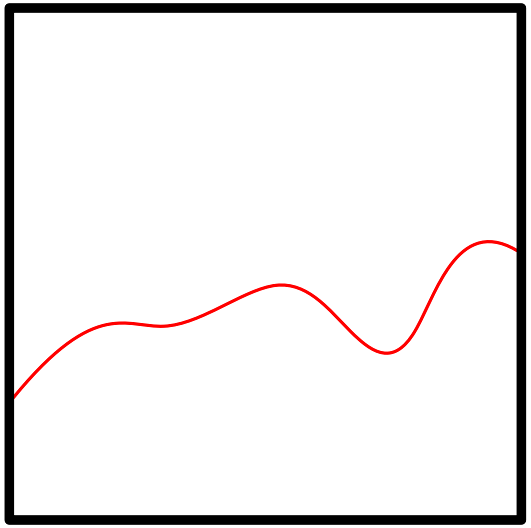
- In many problems, $\square$'s **CANNOT** be independent.
- $\square$ is a polynomial of $a_1, \ldots, a_\beta$
- Some of them have to zero.
- Some of them have to constants
- Some of them depend on other coefficients

How to:

- Create n^β polynomials $P_i(x,y)$
- $-\frac{Q(n)}{n} \leq P(x,y) \leq \frac{Q(n)}{n}$
- Each region is "$Q(n)$-rich"
- **(main challenge)** Intersection of two regions: $\ll \frac{1}{n}$

The Main Open Question

Unit square in 2D



$P(x,y): \quad 0 = \square X^{\Delta} + \square X^{\Delta-1}Y + \ldots + \square X^i Y^j + \ldots + \square Y + \square X + \square$

- In many problems, $\square$'s **CANNOT** be independent.
- $\square$ is a polynomial of $a_1, \ldots, a_\beta$
- Some of them have to zero.
- Some of them have to constants
- Some of them depend on other coefficients

How to:

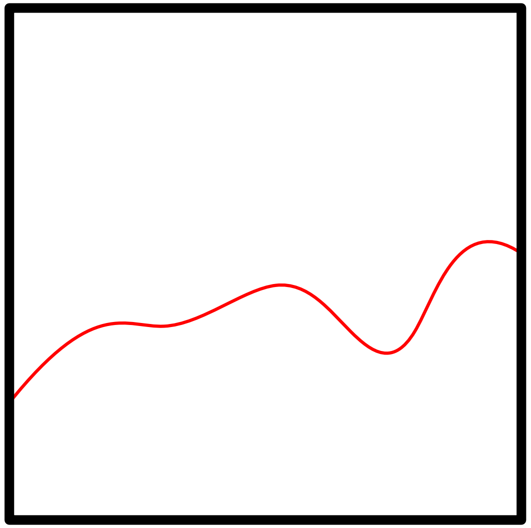
- Create n^β polynomials $P_i(x,y)$
- $-\frac{Q(n)}{n} \leq P(x,y) \leq \frac{Q(n)}{n}$
- Each region is "$Q(n)$-rich"
- **(main challenge)** Intersection of two regions: $\ll \frac{1}{n}$

Hurdle:

- $P_1(x,y)H(x,y) = 0$
- $P_2(x,y)H(x,y) = 0$
- Have arbitrary large coefficient distance
- Infinitely many zeroes in common

The Third Technique

Unit square in 2D



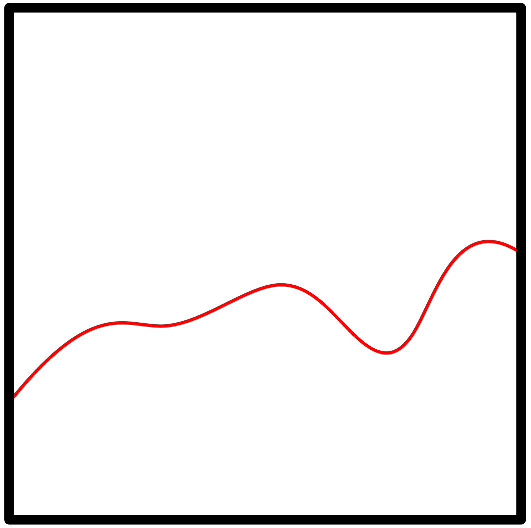
$P(x,y): YG(X) = F(X)$
G and F "far from" sharing a root
$YG(X) - F(X)$ is irreducible

How to:

- Create n^β polynomials $P_i(x,y)$
- $-\frac{Q(n)}{n} \leq P(x,y) \leq \frac{Q(n)}{n}$
- Each region is "$Q(n)$-rich"
- **(main challenge)** Intersection of two regions: $\ll \frac{1}{n}$

The Third Technique

Unit square in 2D



$P(x, y) : YG(X) = F(X)$
G and F "far from" sharing a root
$YG(X) - F(X)$ is irreducible

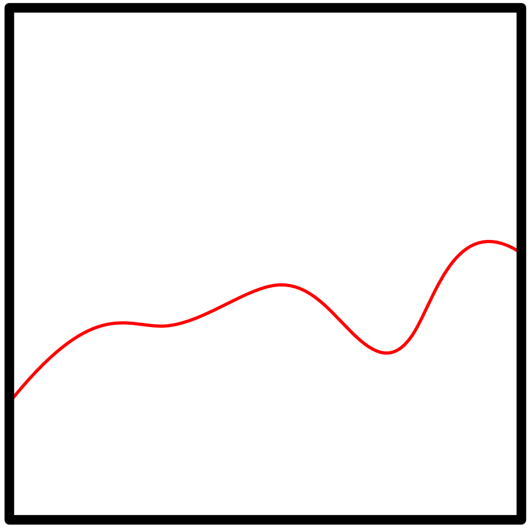
Distance $\frac{Q^{\text{poly}}\Delta(n)}{n}$ and small magnitude is enough to imply **(main challenge)**

How to:

- Create n^β polynomials $P_i(x, y)$
- $-\frac{Q(n)}{n} \leq P(x, y) \leq \frac{Q(n)}{n}$
- Each region is "$Q(n)$-rich"
- **(main challenge)** Intersection of two regions: $\ll \frac{1}{n}$

The End?

How to Main Challenge

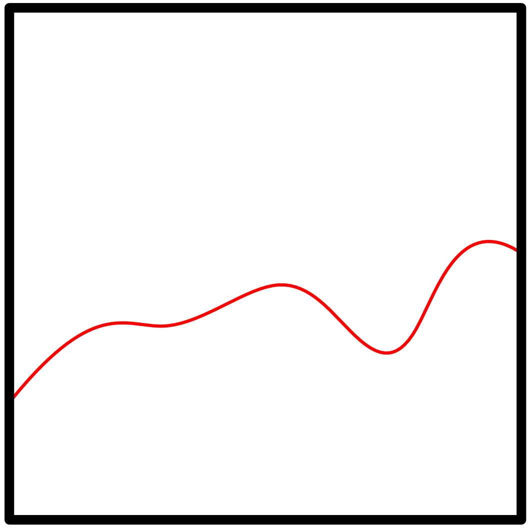


Setup:

$P(x, y) : YG(X) = F(X)$

$t = \text{Resultant}(F, G) > 0$

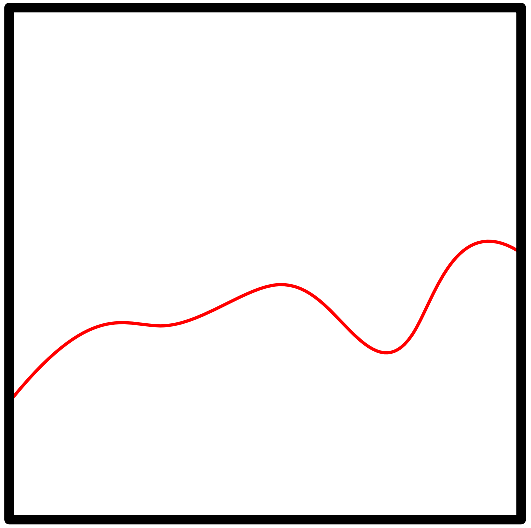
How to Main Challenge



Setup:
$P(x, y) : YG(X) = F(X)$
$t = \text{Resultant}(F, G) > 0$

$$\exists H(X), L(X) : GH + FL = 1$$

How to Main Challenge



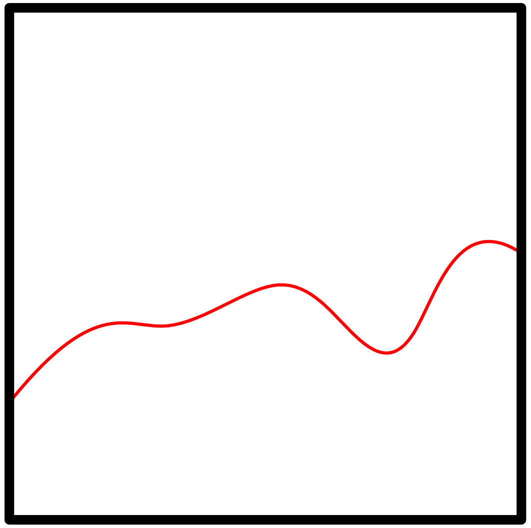
Setup:
$P(x,y) : YG(X) = F(X)$
$t = \text{Resultant}(F,G) > 0$

$$\exists H(X), L(X) : GH + FL = 1$$

Create lots of poly:

- A "grid" of side-length δ around P

How to Main Challenge



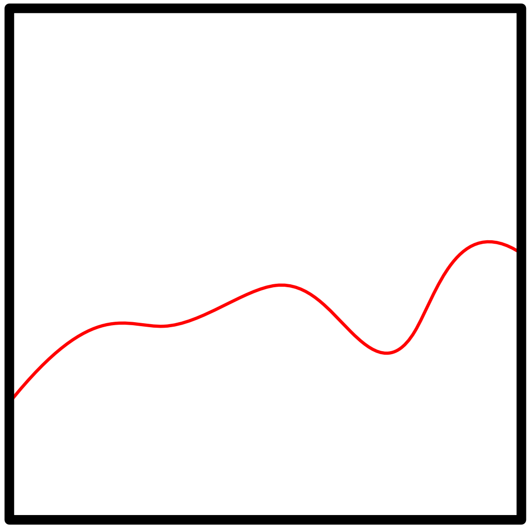
Setup:
$P(x, y) : YG(X) = F(X)$
$t = \text{Resultant}(F, G) > 0$

$\exists H(X), L(X) : GH + FL = 1$

Create lots of poly:

- A "grid" of side-length δ around P
- For each coeff. a of P:
  - For each $i = 0, \ldots, \frac{n}{Q^C(n)}$:
    - Add δi to a

How to Main Challenge



Setup:
$P(x, y) : YG(X) = F(X)$
$t = \text{Resultant}(F, G) > 0$

$$\exists H(X), L(X) : GH + FL = 1$$

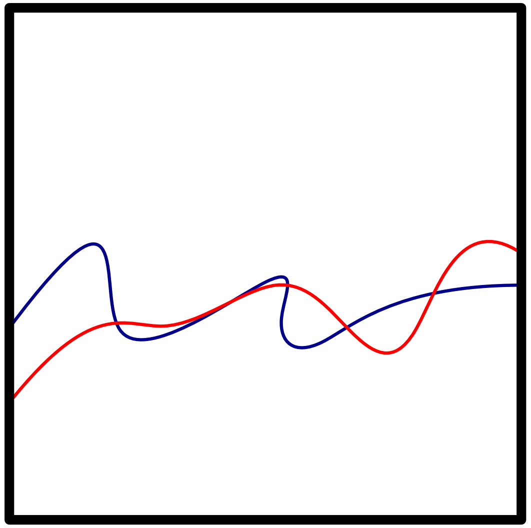
Create lots of poly:

- A "grid" of side-length δ around P
- For each coeff. a of P:
  - For each $i = 0, \ldots, \frac{n}{Q^C(n)}$:
    - Add δi to a

Get:

- $M = n^\beta$ polys, $P_1, ..., P_M$ (ignoring poly $Q(n)$ factors)
- Every two differ at by at least δ in one coeff.
- Every P_i in a small neighborhood of P (within radius $n\delta$)
- δ sufficiently small constant
- Region: $0 \leq P_i(x, y) \leq \frac{Q(n)}{n} = w$

How to Main Challenge

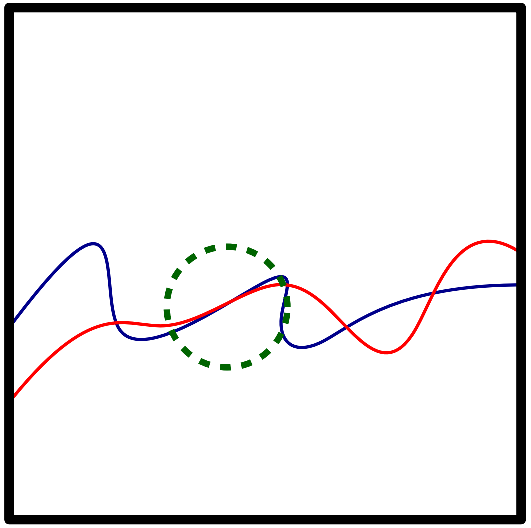


Setup:
$P(x, y) : YG(X) = F(X)$
$t = \text{Resultant}(F, G) > 0$

$\exists H(X), L(X) : GH + FL = 1$

Consider P_1 and P_2:

How to Main Challenge



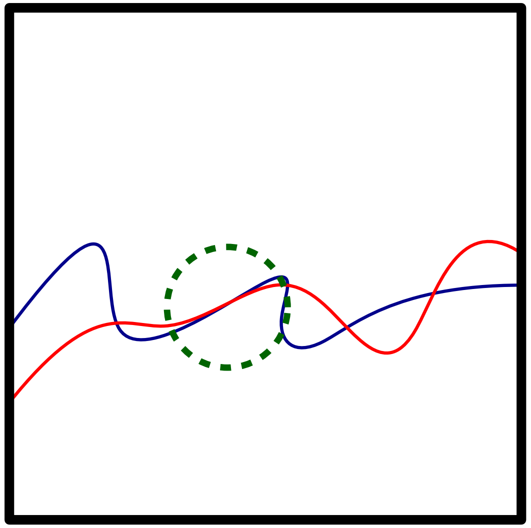
Setup:
$P(x, y) : YG(X) = F(X)$
$t = \text{Resultant}(F, G) > 0$

$\exists H(X), L(X) : GH + FL = 1$

Consider P_1 and P_2:

Imagine big overlap

How to Main Challenge

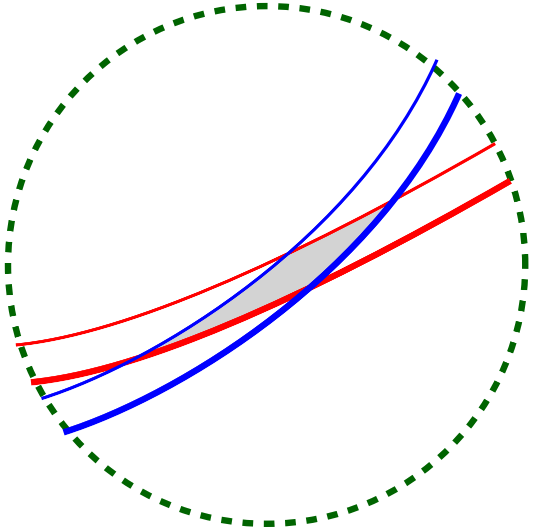


Setup:
$P(x, y) : YG(X) = F(X)$
$t = \text{Resultant}(F, G) > 0$

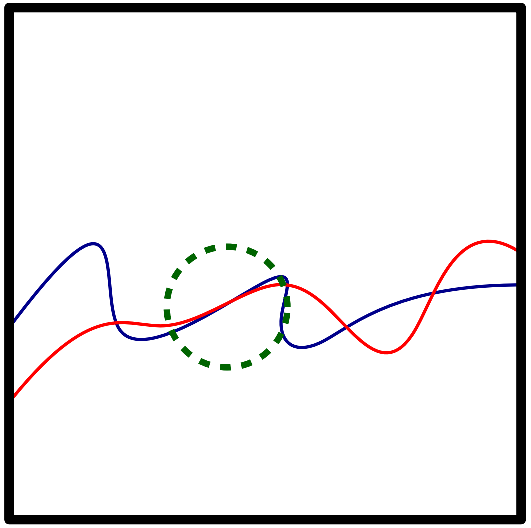
$\exists H(X), L(X) : GH + FL = 1$

Consider P_1 and P_2:

Imagine big overlap



How to Main Challenge

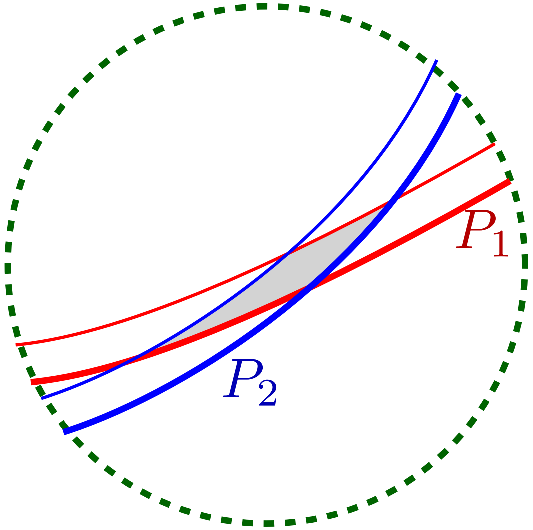


Setup:
$P(x,y): YG(X) = F(X)$
$t = \text{Resultant}(F,G) > 0$

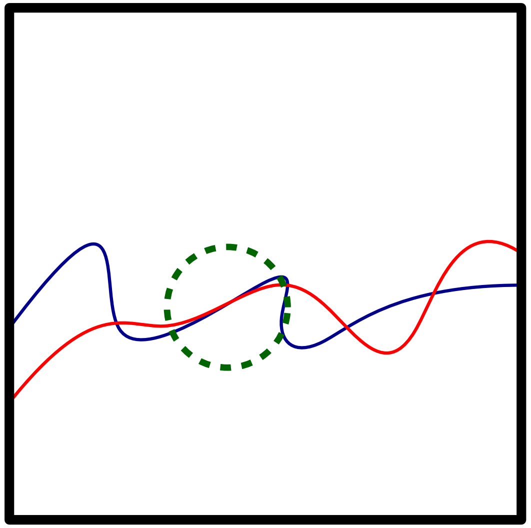
$\exists H(X), L(X) : GH + FL = 1$

Consider P_1 and P_2:

Imagine big overlap



How to Main Challenge



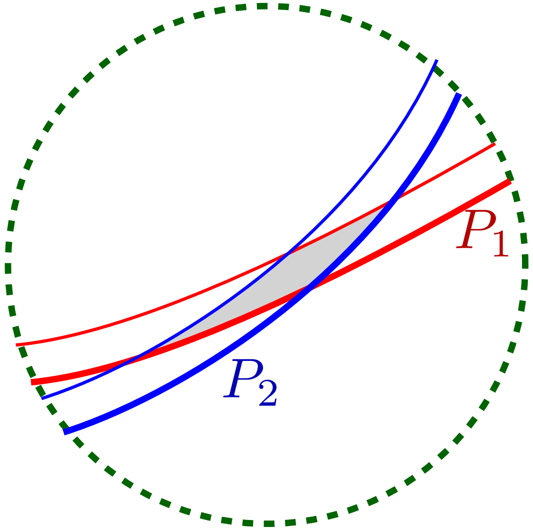
Setup:
$P(x, y) : YG(X) = F(X)$
$t = \text{Resultant}(F, G) > 0$

$\exists H(X), L(X) : GH + FL = 1$

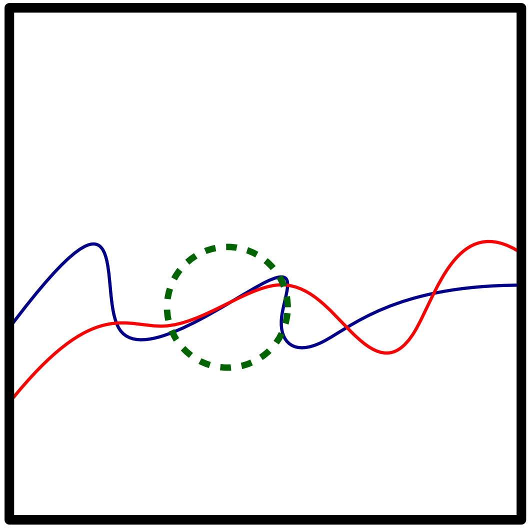
Consider P_1 and P_2:

Imagine big overlap

P_1 and P_2 evaluate within $[0, w]$ in a big interval I of length at least $\frac{1}{Q(n)}$



How to Main Challenge

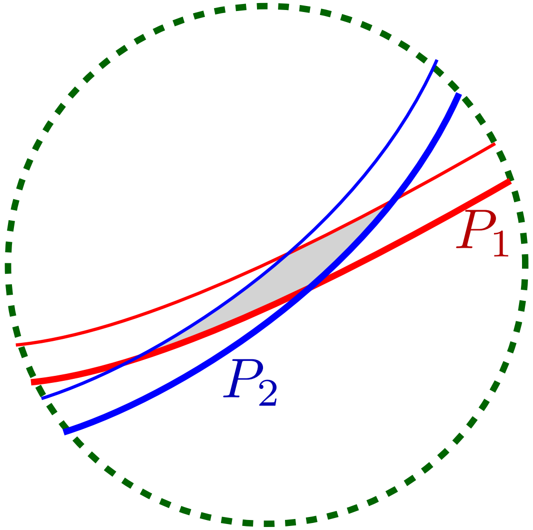


Setup:
$P(x, y) : YG(X) = F(X)$
$t =$ Resultant$(F, G) > 0$

$\exists H(X), L(X) : GH + FL = 1$

Consider P_1 and P_2:

Imagine big overlap

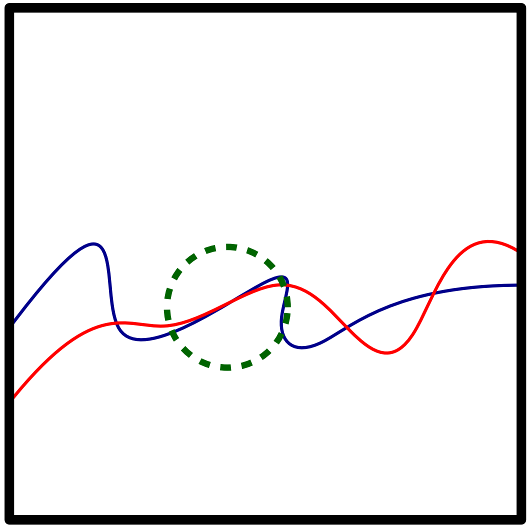


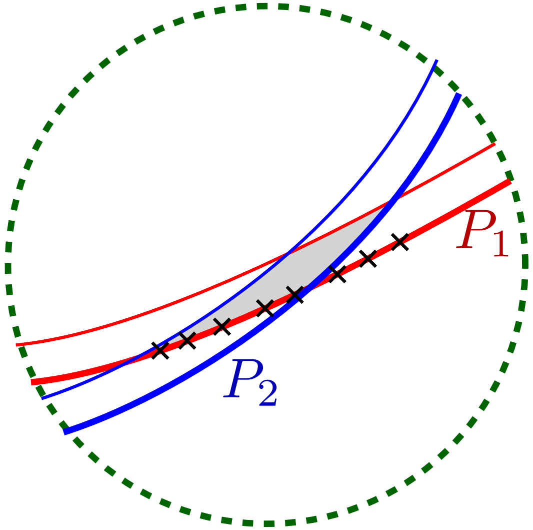
P_1 and P_2 evaluate within $[0, w]$ in a big interval I of length at least $\frac{1}{Q(n)}$

Approach:

- Pick ℓ points in I on P_1

How to Main Challenge





Setup:
$P(x, y) : YG(X) = F(X)$
$t =$ Resultant$(F, G) > 0$

$\exists H(X), L(X) : GH + FL = 1$

Consider P_1 and P_2:

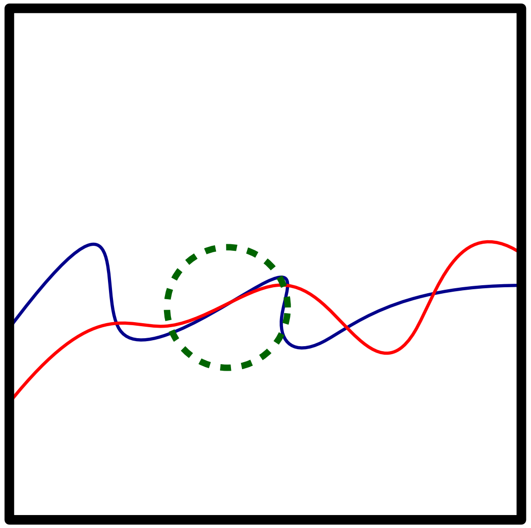
Imagine big overlap

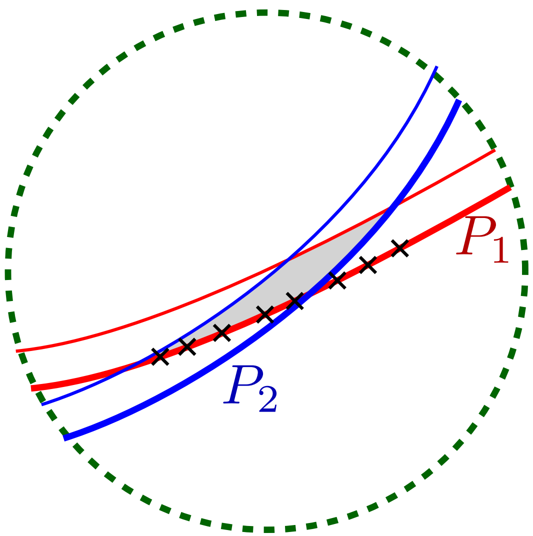
P_1 and P_2 evaluate within $[0, w]$ in a big interval I of length at least $\frac{1}{Q(n)}$

Approach:

- Pick ℓ points in I on P_1

How to Main Challenge





Setup:
$P(x, y) : YG(X) = F(X)$
$t = \text{Resultant}(F, G) > 0$

$\exists H(X), L(X) : GH + FL = 1$

Consider P_1 and P_2:

Imagine big overlap

P_1 and P_2 evaluate within $[0, w]$ in a big interval I of length at least $\frac{1}{Q(n)}$

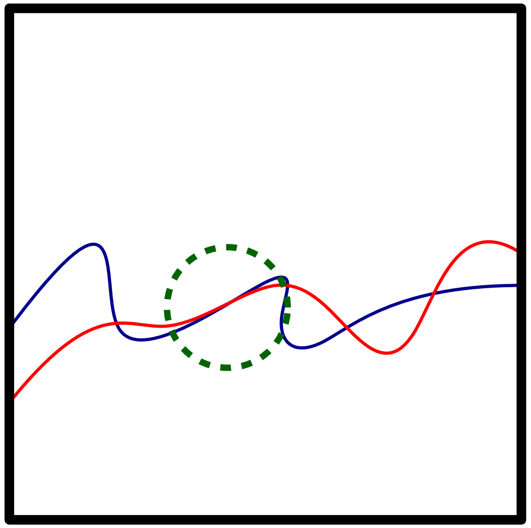
Approach:

- Pick ℓ points in I on P_1

V: Vector of monomials:

- all monomials except yX^{Δ_G}.
- X^i for $i = 1, ..., k$ so we get ℓ mono. in total
- Build an $\ell \times \ell$ matrix A:
  - Row i is the evaluation of V on the i-th point

How to Main Challenge



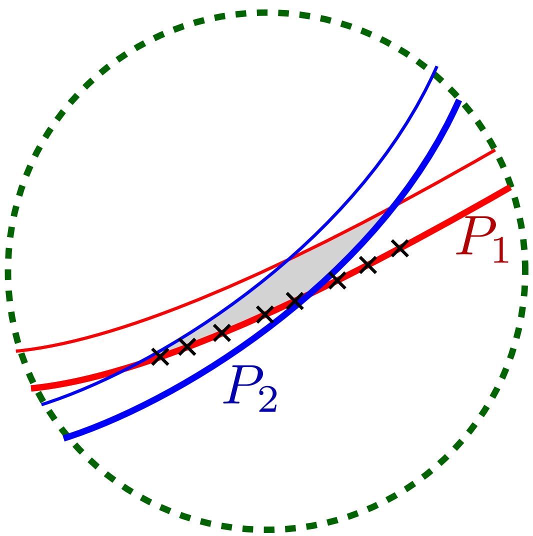
Setup:
$P(x, y) : YG(X) = F(X)$
$t = \text{Resultant}(F, G) > 0$

$\exists H(X), L(X) : GH + FL = 1$

Consider P_1 and P_2:

Imagine big overlap

P_1 and P_2 evaluate within $[0, w]$ in a big interval I of length at least $\frac{1}{Q(n)}$



Approach:

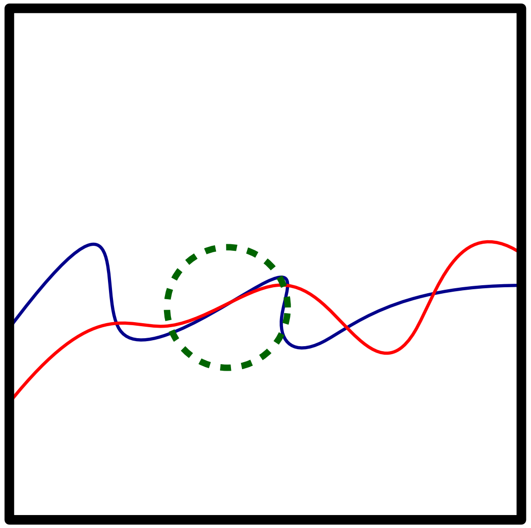
- Pick ℓ points in I on P_1

V: Vector of monomials:

- all monomials except yX^{Δ_G}.
- X^i for $i = 1, ..., k$ so we get ℓ mono. in total
- Build an $\ell \times \ell$ matrix A:
  - Row i is the evaluation of V on the i-th point

Claim: $|\det(A)| \geq \text{Resultant}(F, G)|I|^{\ell^2} - O(w)$

How to Main Challenge



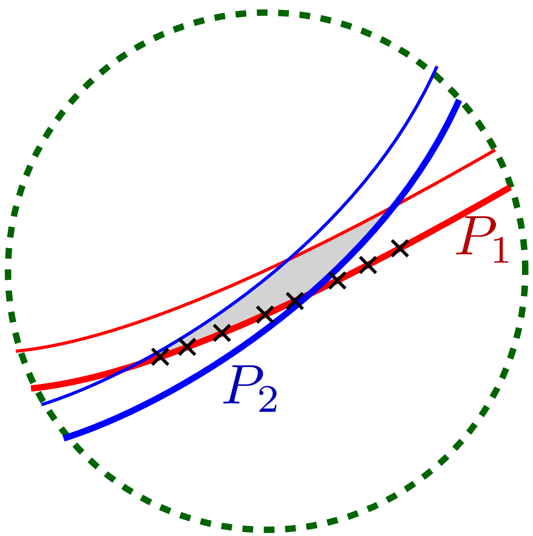
Setup:
$P(x, y) : YG(X) = F(X)$
$t =$ Resultant$(F, G) > 0$

$\exists H(X), L(X) : GH + FL = 1$

Consider P_1 and P_2:

Imagine big overlap

P_1 and P_2 evaluate within $[0, w]$ in a big interval I of length at least $\frac{1}{Q(n)}$



Tweak coeff of P_2 by smaller than δ to pass through the ℓ points $\Rightarrow$ contradiction

Approach:

- Pick ℓ points in I on P_1

V: Vector of monomials:

- all monomials except yX^{Δ_G}.
- X^i for $i = 1, ..., k$ so we get ℓ mono. in total
- Build an $\ell \times \ell$ matrix A:
  - Row i is the evaluation of V on the i-th point

Claim: $|\det(A)| \geq$ Resultant$(F, G)|I|^{\ell^2} - O(w)$

Thank you!