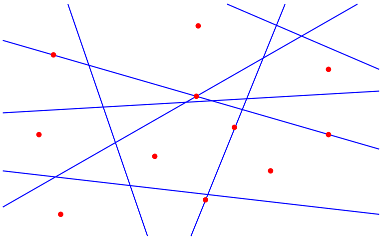
Hopcroft's Problem

2D Fractional Cascading and Decision Trees



Timothy M. Chan and **Da Wei Zheng**

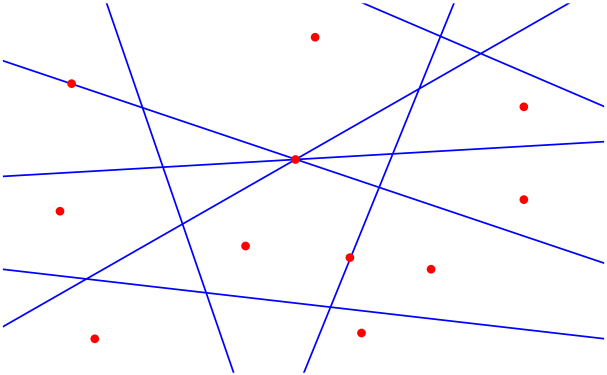
January 9, 2022

University of Illinois Urbana-Champaign

Outline

What is Hopcroft's problem?

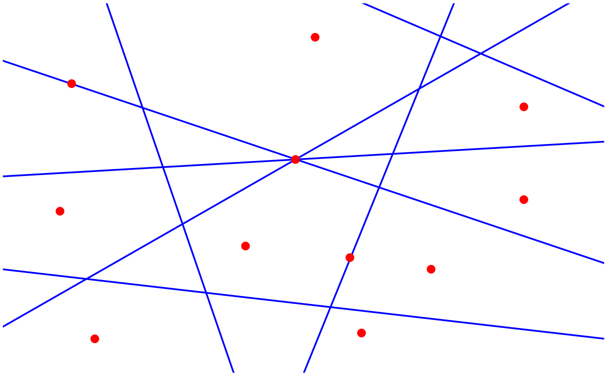
Given *n* points and *n* lines, does any point lie on any line?



What is Hopcroft's problem?

Given *n* points and *n* lines, does any point lie on any line?

Given *n* points and *n* lines, how many **point-line incidences** are there?

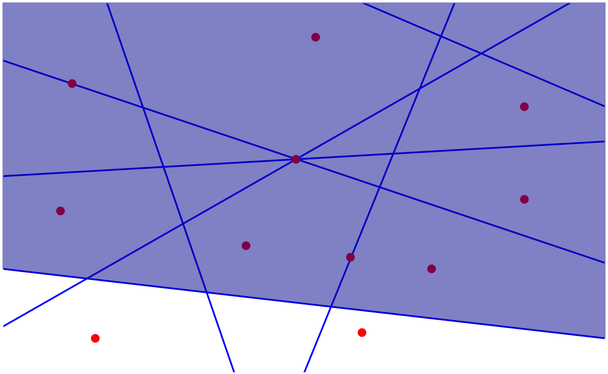


What is Hopcroft's problem?

Given *n* points and *n* lines, does any point lie on any line?

Given *n* points and *n* lines, how many **point-line incidences** are there?

Given *n* points and *n* lines, how many **line-below-point pairs** are there?

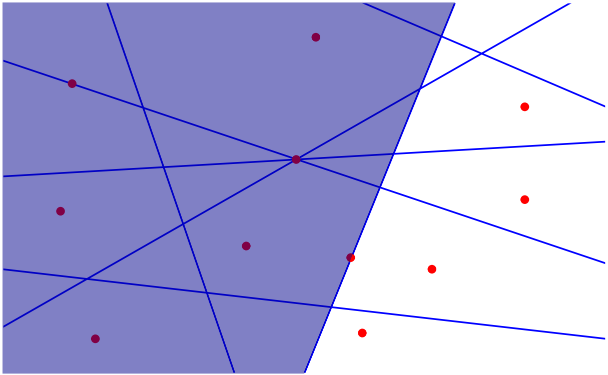


What is Hopcroft's problem?

Given *n* points and *n* lines, does any point lie on any line?

Given *n* points and *n* lines, how many **point-line incidences** are there?

Given *n* points and *n* lines, how many **line-below-point pairs** are there?

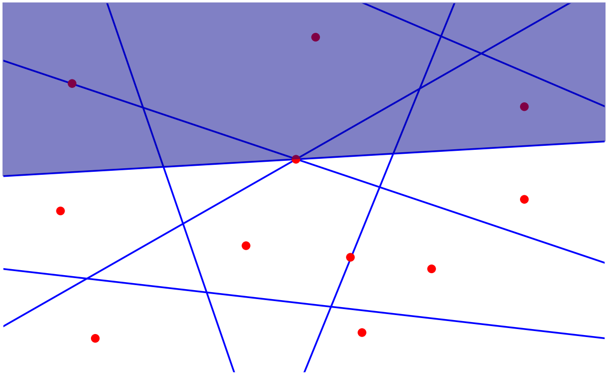


What is Hopcroft's problem?

Given *n* points and *n* lines, does any point lie on any line?

Given *n* points and *n* lines, how many **point-line incidences** are there?

Given *n* points and *n* lines, how many **line-below-point pairs** are there?

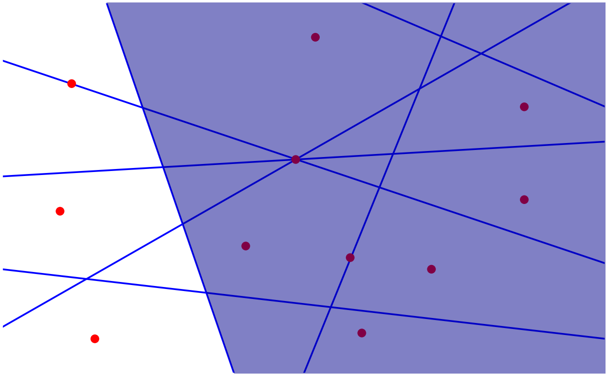


What is Hopcroft's problem?

Given n points and n lines, does any point lie on any line?

Given n points and n lines, how many **point-line incidences** are there?

Given n points and n lines, how many **line-below-point pairs** are there?

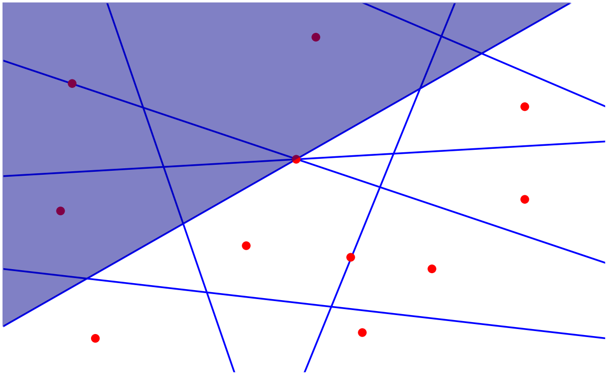


What is Hopcroft's problem?

Given n points and n lines, does any point lie on any line?

Given n points and n lines, how many **point-line incidences** are there?

Given n points and n lines, how many **line-below-point pairs** are there?

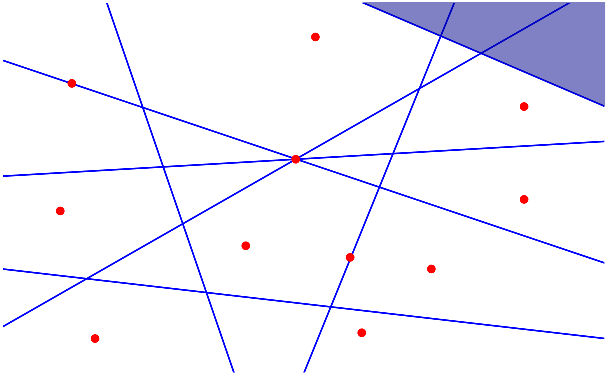


What is Hopcroft's problem?

Given *n* points and *n* lines, does any point lie on any line?

Given *n* points and *n* lines, how many **point-line incidences** are there?

Given *n* points and *n* lines, how many **line-below-point pairs** are there?

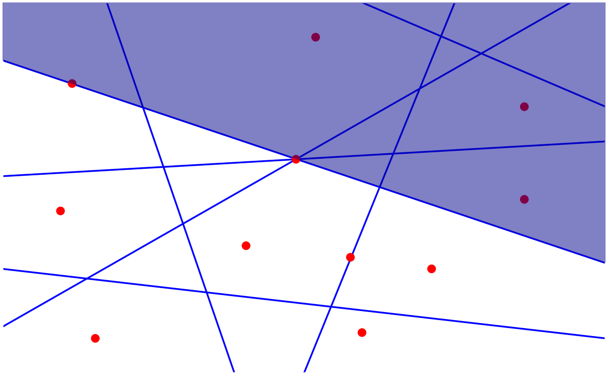


What is Hopcroft's problem?

Given *n* points and *n* lines, does any point lie on any line?

Given *n* points and *n* lines, how many **point-line incidences** are there?

Given *n* points and *n* lines, how many **line-below-point pairs** are there?



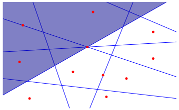
Why do we care about Hopcroft's problem?

It is related to many **offline** problems involving range searching.

Why do we care about Hopcroft's problem?

It is related to many **offline** problems involving range searching.

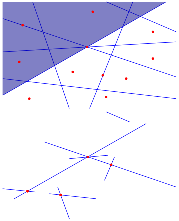
- Offline half-space range query
- Offline simplex range query



Why do we care about Hopcroft's problem?

It is related to many **offline** problems involving range searching.

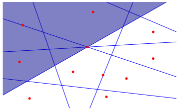
- Offline half-space range query
- Offline simplex range query
- 2D line segment intersection counting
- 2D line segment connected components

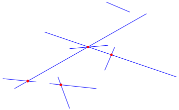


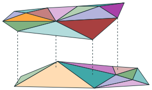
Why do we care about Hopcroft's problem?

It is related to many **offline** problems involving range searching.

- Offline half-space range query
- Offline simplex range query
- 2D line segment intersection counting
- 2D line segment connected components
- 3D line towering problem
- 3D vertical distance between polyhedral terrains



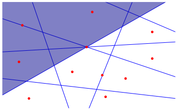
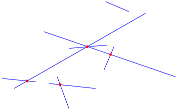
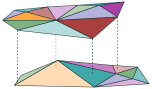
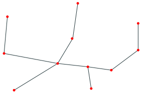




Why do we care about Hopcroft's problem?

It is related to many **offline** problems involving range searching.

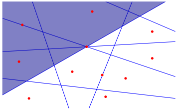
- Offline half-space range query
- Offline simplex range query
- 2D line segment intersection counting
- 2D line segment connected components
- 3D line towering problem
- 3D vertical distance between polyhedral terrains
- 3D Bichromatic closest pair
- 3D Euclidean Minimum Spanning Tree

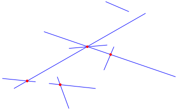





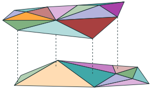
Why do we care about Hopcroft's problem?

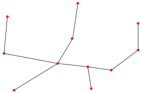
It is related to many **offline** problems involving range searching.

- Offline half-space range query
- Offline simplex range query
- 2D line segment intersection counting
- 2D line segment connected components
- 3D line towering problem
- 3D vertical distance between polyhedral terrains
- 3D Bichromatic closest pair
- 3D Euclidean Minimum Spanning Tree









... and many other problems in computational geometry!

Outline

History

Posed by Hopcroft in the 1980s - n points and n lines, any point on any line?

- $O(n^2)$ Brute force.

History

Posed by Hopcroft in the 1980s - n points and n lines, any point on any line?

- $O(n^2)$ Brute force.
- $O(n^{1.695})$ [Chazelle, 1986] (line segment intersection).

History

Posed by Hopcroft in the 1980s - n points and n lines, any point on any line?

- $O(n^2)$ Brute force.
- $O(n^{1.695})$ [Chazelle, 1986] (line segment intersection).
- $O(n^{3/2} \log^{1/2} n)$ [Hopcroft & Seidel, 1986?] (no paper).

History

Posed by Hopcroft in the 1980s - n points and n lines, any point on any line?

- $O(n^2)$ Brute force.
- $O(n^{1.695})$ [Chazelle, 1986] (line segment intersection).
- $O(n^{3/2} \log^{1/2} n)$ [Hopcroft & Seidel, 1986?] (no paper).
- $O(n^{1.412})$ [Cole, Sharir, Yap, 1987].

History

Posed by Hopcroft in the 1980s - n points and n lines, any point on any line?

- $O(n^2)$ Brute force.
- $O(n^{1.695})$ [Chazelle, 1986] (line segment intersection).
- $O(n^{3/2} \log^{1/2} n)$ [Hopcroft & Seidel, 1986?] (no paper).
- $O(n^{1.412})$ [Cole, Sharir, Yap, 1987].
- $O(n^{4/3+\varepsilon})$ [Edelsbrunner, Guibas, Sharir, 1990].

History

Posed by Hopcroft in the 1980s - n points and n lines, any point on any line?

- $O(n^2)$ Brute force.
- $O(n^{1.695})$ [Chazelle, 1986] (line segment intersection).
- $O(n^{3/2} \log^{1/2} n)$ [Hopcroft & Seidel, 1986?] (no paper).
- $O(n^{1.412})$ [Cole, Sharir, Yap, 1987].
- $O(n^{4/3+\varepsilon})$ [Edelsbrunner, Guibas, Sharir, 1990].
- $O(n^{4/3} \log^4 n)$ [Edelsbrunner, Guibas, Hershberger, Seidel, Sharir, Snoeyink, Welzl 1989].

History

Posed by Hopcroft in the 1980s - n points and n lines, any point on any line?

- $O(n^2)$ Brute force.
- $O(n^{1.695})$ [Chazelle, 1986] (line segment intersection).
- $O(n^{3/2} \log^{1/2} n)$ [Hopcroft & Seidel, 1986?] (no paper).
- $O(n^{1.412})$ [Cole, Sharir, Yap, 1987].
- $O(n^{4/3+\varepsilon})$ [Edelsbrunner, Guibas, Sharir, 1990].
- $O(n^{4/3} \log^4 n)$ [Edelsbrunner, Guibas, Hershberger, Seidel, Sharir, Snoeyink, Welzl 1989].
- $O(n^{4/3} \log^{1.78} n)$ [Agarwal, 1990].

History

Posed by Hopcroft in the 1980s - n points and n lines, any point on any line?

- $O(n^2)$ Brute force.
- $O(n^{1.695})$ [Chazelle, 1986] (line segment intersection).
- $O(n^{3/2} \log^{1/2} n)$ [Hopcroft & Seidel, 1986?] (no paper).
- $O(n^{1.412})$ [Cole, Sharir, Yap, 1987].
- $O(n^{4/3+\varepsilon})$ [Edelsbrunner, Guibas, Sharir, 1990].
- $O(n^{4/3} \log^4 n)$ [Edelsbrunner, Guibas, Hershberger, Seidel, Sharir, Snoeyink, Welzl 1989].
- $O(n^{4/3} \log^{1.78} n)$ [Agarwal, 1990].
- $O(n^{4/3} \log^{1/3} n)$ [Chazelle, 1993].

History

Posed by Hopcroft in the 1980s - n points and n lines, any point on any line?

- $O(n^2)$ Brute force.
- $O(n^{1.695})$ [Chazelle, 1986] (line segment intersection).
- $O(n^{3/2} \log^{1/2} n)$ [Hopcroft & Seidel, 1986?] (no paper).
- $O(n^{1.412})$ [Cole, Sharir, Yap, 1987].
- $O(n^{4/3+\varepsilon})$ [Edelsbrunner, Guibas, Sharir, 1990].
- $O(n^{4/3} \log^4 n)$ [Edelsbrunner, Guibas, Hershberger, Seidel, Sharir, Snoeyink, Welzl 1989].
- $O(n^{4/3} \log^{1.78} n)$ [Agarwal, 1990].
- $O(n^{4/3} \log^{1/3} n)$ [Chazelle, 1993].
- $O(n^{4/3} 2^{O(\log^* n)})$ [Matoušek 1993].

History

Posed by Hopcroft in the 1980s - n points and n lines, any point on any line?

- $O(n^2)$ Brute force.
- $O(n^{1.695})$ [Chazelle, 1986] (line segment intersection).
- $O(n^{3/2} \log^{1/2} n)$ [Hopcroft & Seidel, 1986?] (no paper).
- $O(n^{1.412})$ [Cole, Sharir, Yap, 1987].
- $O(n^{4/3+\varepsilon})$ [Edelsbrunner, Guibas, Sharir, 1990].
- $O(n^{4/3} \log^4 n)$ [Edelsbrunner, Guibas, Hershberger, Seidel, Sharir, Snoeyink, Welzl 1989].
- $O(n^{4/3} \log^{1.78} n)$ [Agarwal, 1990].
- $O(n^{4/3} \log^{1/3} n)$ [Chazelle, 1993].
- $O(n^{4/3} 2^{O(\log^* n)})$ [Matoušek 1993].
- $\Omega(n^{4/3})$ [Erickson 1996] (For so-called partitioning algorithm).

History

Posed by Hopcroft in the 1980s - n points and n lines, any point on any line?

- $O(n^2)$ Brute force.
- $O(n^{1.695})$ [Chazelle, 1986] (line segment intersection).
- $O(n^{3/2} \log^{1/2} n)$ [Hopcroft & Seidel, 1986?] (no paper).
- $O(n^{1.412})$ [Cole, Sharir, Yap, 1987].
- $O(n^{4/3+\varepsilon})$ [Edelsbrunner, Guibas, Sharir, 1990].
- $O(n^{4/3} \log^4 n)$ [Edelsbrunner, Guibas, Hershberger, Seidel, Sharir, Snoeyink, Welzl 1989].
- $O(n^{4/3} \log^{1.78} n)$ [Agarwal, 1990].
- $O(n^{4/3} \log^{1/3} n)$ [Chazelle, 1993].
- $O(n^{4/3} 2^{O(\log^* n)})$ [Matoušek 1993].
- $\Omega(n^{4/3})$ [Erickson 1996] (For so-called partitioning algorithm).
- and now ...

And now...

$O(n^{4/3})$ algorithm for Hopcroft's problem **NEW!**

Extensions

It also improves the runtime of many related problems!

Extensions

It also improves the runtime of many related problems!

- Offline half-space range query [Matoušek, '93] $O(n^{2d/(d+1)}2^{O(\log^* n)})$
- Offline simplex range query [Matoušek, '93] $O(n^{2d/(d+1)}2^{O(\log^* n)})$
- 2D line segment intersection counting [Chazelle, '83] $O(n^{4/3}\log^{1/3} n)$
- 2D line segment connected components [Lopez, Thurimella, '85] $O(n^{4/3}\log^{3} n)$
- 3D line towering problem. [Chazelle, Edelsbrunner, Guibas, Sharir, '94] $O(n^{4/3+\varepsilon})$
- 3D vertical distance between polyhedral terrains [↑↑↑↑↑, '94] $O(n^{4/3+\varepsilon})$
- 3D Bichromatic closest pair [Agarwal, Edelsbrunner, Schwarzkopf, Welzl, '93] $O(n^{4/3}\log^{4/3} n)$
- 3D Euclidean Minimum Spanning Tree [↑↑↑↑↑, '93] $O(n^{4/3}\log^{4/3} n)$

Extensions

It also improves the runtime of many related problems!

- Offline half-space range query $O(n^{2d/(d+1)}2^{O(\log^* n)})$
- Offline simplex range query $O(n^{2d/(d+1)}2^{O(\log^* n)})$
- 2D line segment intersection counting $O(n^{4/3}\log^{1/3} n)$
- 2D line segment connected components $O(n^{4/3}\log^{3} n)$
- 3D line towering problem $O(n^{4/3+\varepsilon})$
- 3D vertical distance between polyhedral terrains $O(n^{4/3+\varepsilon})$
- 3D Bichromatic closest pair $O(n^{4/3}\log^{4/3} n)$
- 3D Euclidean Minimum Spanning Tree $O(n^{4/3}\log^{4/3} n)$

Extensions

It also improves the runtime of many related problems!

- Offline half-space range query $O(n^{2d/(d+1)})$
- Offline simplex range query $O(n^{2d/(d+1)})$
- 2D line segment intersection counting $O(n^{4/3})$
- 2D line segment connected components $O(n^{4/3})$
- 3D line towering problem $O(n^{4/3})$
- 3D vertical distance between polyhedral terrains $O(n^{4/3})$
- 3D Bichromatic closest pair $O(n^{4/3})$
- 3D Euclidean Minimum Spanning Tree $O(n^{4/3})$

Outline

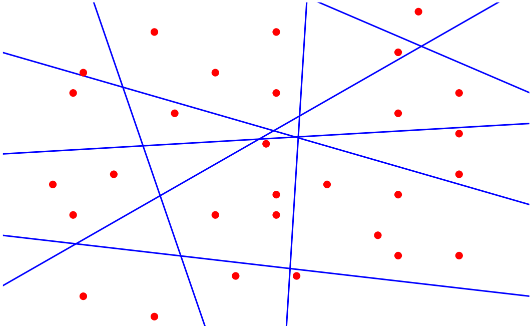
Asymmetric Hopcroft's Problem

Let $T(m, n)$ be the time to solve Hopcroft's problem with m points and n lines.

Asymmetric Hopcroft's Problem

Let $T(m, n)$ be the time to solve Hopcroft's problem with m points and n lines.

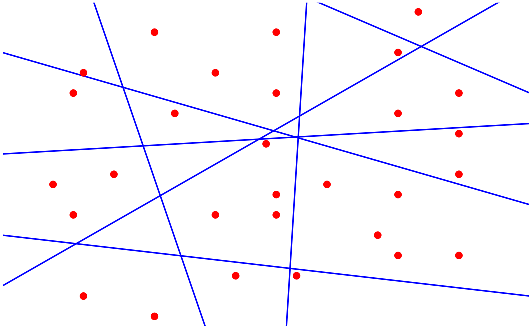
What if we have a lot more points than lines, say $m > n^2$?



Asymmetric Hopcroft's Problem

Let $T(m, n)$ be the time to solve Hopcroft's problem with m points and n lines.

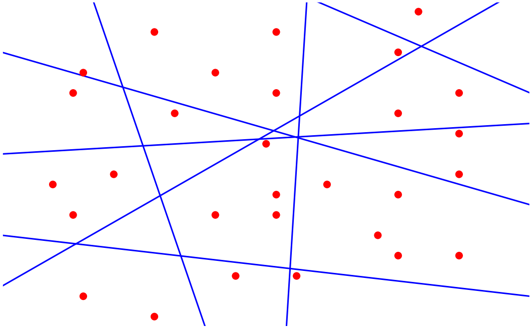
What if we have a lot more points than lines, say $m > n^2$?



Point Location Data structure - There exists an $O(n^2)$ data structure that allows for point location queries in $O(\log n)$ time

Let $T(m, n)$ be the time to solve Hopcroft's problem with m points and n lines.

What if we have a lot more points than lines, say $m > n^2$?

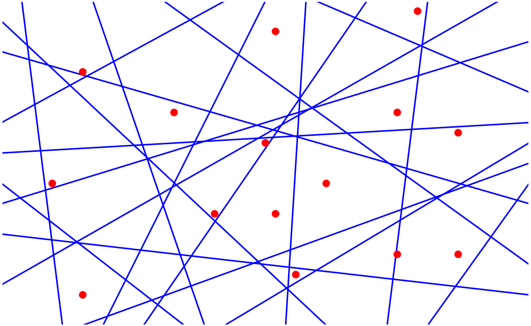


Point Location Data structure - There exists an $O(n^2)$ data structure that allows for point location queries in $O(\log n)$ time , so $T(m, n) = O(n^2 + m \log n)$.

More lines than points

Let $T(m, n)$ be the time to solve Hopcroft's problem with m points and n lines.

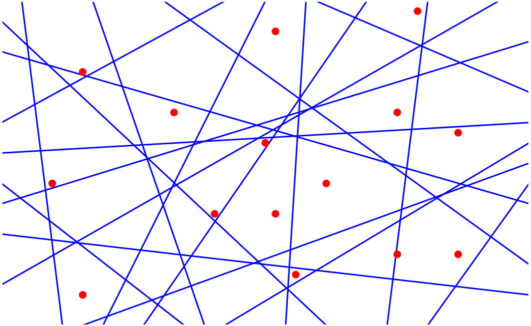
What if we have a lot more lines than points, say $n > m^2$?



More lines than points

Let $T(m, n)$ be the time to solve Hopcroft's problem with m points and n lines.

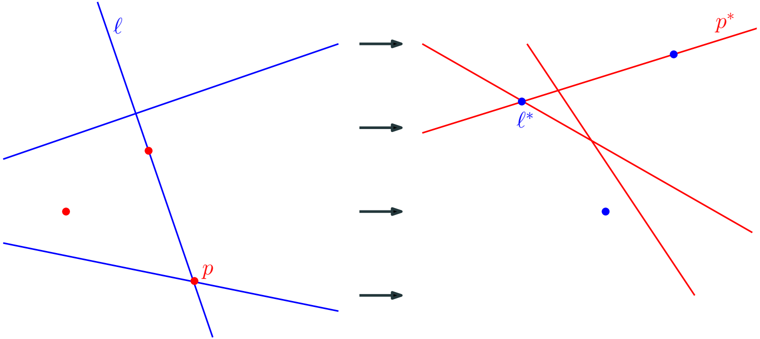
What if we have a lot more lines than points, say $n > m^2$?



It would be nice if we can exchange our lines with our points.

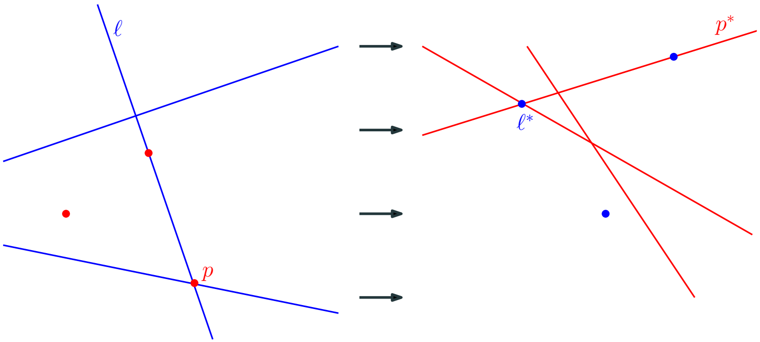
Point-Line Duality

Point-Line Duality - There exists a transform that takes points to lines and lines to points that preserves incidences and above-below relationships.



Point-Line Duality

Point-Line Duality - There exists a transform that takes points to lines and lines to points that preserves incidences and above-below relationships.

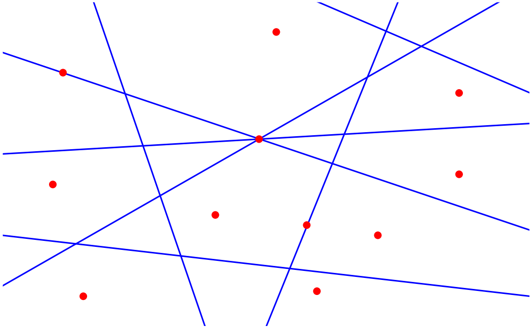


$$T(m, n) = T(n, m)$$

Nearly Equal Case

Let $T(m, n)$ be the time to solve Hopcroft's problem with m points and n lines.

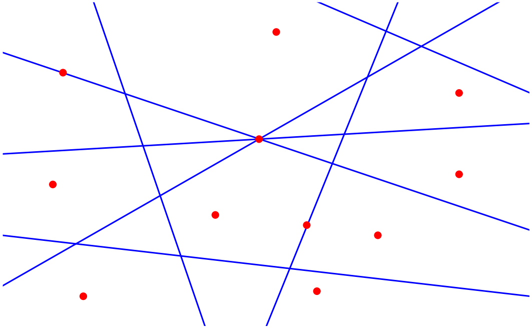
What if we have roughly equal number of lines and points, say $\sqrt{m} < n < m^2$?



Nearly Equal Case

Let $T(m, n)$ be the time to solve Hopcroft's problem with m points and n lines.

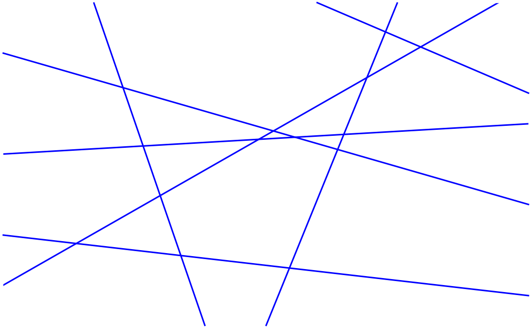
What if we have roughly equal number of lines and points, say $\sqrt{m} < n < m^2$?



Divide and conquer?

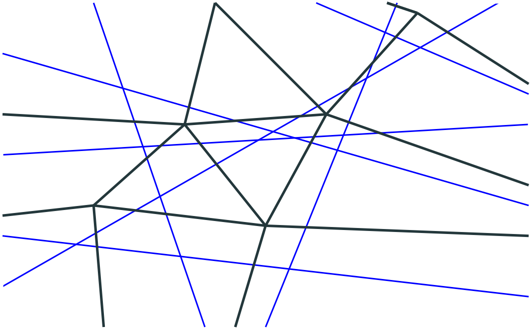
2D Divide and Conquer

Cuttings - Given n lines and $r < n$, there exists a decomposition of $\mathbb{R}^2$ into $O(r^2)$ cells each with at most $\frac{n}{r}$ lines crossing each cell



2D Divide and Conquer

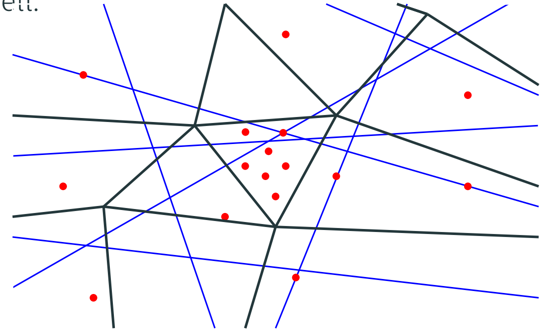
Cuttings - Given n lines and $r < n$, there exists a decomposition of $\mathbb{R}^2$ into $O(r^2)$ cells each with at most $\frac{n}{r}$ lines crossing each cell



We can find these $1/r$-cuttings in time $O(nr)$.

2D Divide and Conquer

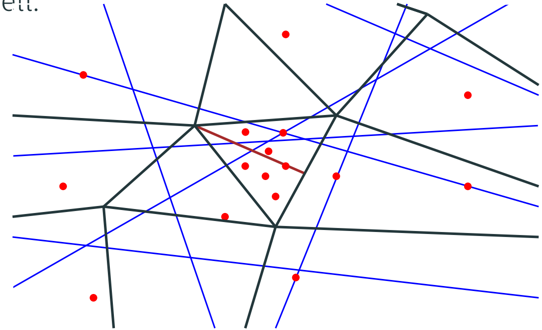
Cuttings - Given m points and n lines and $r < n$, there exists a decomposition of $\mathbb{R}^2$ into $O(r^2)$ cells each with at most $\frac{n}{r}$ lines crossing each cell, and at most $\frac{m}{r^2}$ points are in each cell.



We can find these $1/r$-cuttings in time $O(nr + m \log r)$.

2D Divide and Conquer

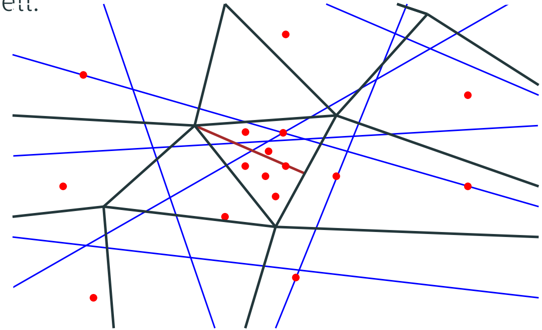
Cuttings - Given m points and n lines and $r < n$, there exists a decomposition of $\mathbb{R}^2$ into $O(r^2)$ cells each with at most $\frac{n}{r}$ lines crossing each cell, and at most $\frac{m}{r^2}$ points are in each cell.



We can find these $1/r$-cuttings in time $O(nr + m \log r)$.

2D Divide and Conquer

Cuttings - Given m points and n lines and $r < n$, there exists a decomposition of $\mathbb{R}^2$ into $O(r^2)$ cells each with at most $\frac{n}{r}$ lines crossing each cell, and at most $\frac{m}{r^2}$ points are in each cell.



We can find these $1/r$-cuttings in time $O(nr + m \log r)$.

Now we can decompose the problem: $T(m, n) = O(r^2)T\left(\frac{m}{r^2}, \frac{n}{r}\right) + O(nr + m \log r)$.

Applying cuttings to Hopcroft's problem [Chazelle, 1993]

Let $T(m, n)$ be the time to solve Hopcroft's problem with m points and n lines.

Applying cuttings to Hopcroft's problem [Chazelle, 1993]

Let $T(m, n)$ be the time to solve Hopcroft's problem with m points and n lines.

$$T(n, n) = O(r^2)T\left(\frac{n}{r^2}, \frac{n}{r}\right) + O(nr + n \log r)$$

Applying cuttings to Hopcroft's problem [Chazelle, 1993]

Let $T(m, n)$ be the time to solve Hopcroft's problem with m points and n lines.

$$T(n, n) = O(r^2)T\left(\frac{n}{r^2}, \frac{n}{r}\right) + O(nr + n\log r)$$

Choose $r = n^{1/3}$.

Applying cuttings to Hopcroft's problem [Chazelle, 1993]

Let $T(m, n)$ be the time to solve Hopcroft's problem with m points and n lines.

$$T(n, n) = O(r^2)T\left(\frac{n}{r^2}, \frac{n}{r}\right) + O(nr + n\log r)$$

Choose $r = n^{1/3}$.

$$T(n, n) = O(n^{2/3})T(n^{1/3}, n^{2/3}) + O(n^{4/3})$$

Applying cuttings to Hopcroft's problem [Chazelle, 1993]

Let $T(m, n)$ be the time to solve Hopcroft's problem with m points and n lines.

$$T(n, n) = O(r^2)T\left(\frac{n}{r^2}, \frac{n}{r}\right) + O(nr + n \log r)$$

Choose $r = n^{1/3}$.

$$T(n, n) = O(n^{2/3})T(n^{1/3}, n^{2/3}) + O(n^{4/3})$$

Use duality + point location: $T(n^{1/3}, n^{2/3}) = O(n^{2/3} + n^{2/3} \log n)$.

Applying cuttings to Hopcroft's problem [Chazelle, 1993]

Let $T(m, n)$ be the time to solve Hopcroft's problem with m points and n lines.

$$T(n, n) = O(r^2)T\left(\frac{n}{r^2}, \frac{n}{r}\right) + O(nr + n \log r)$$

Choose $r = n^{1/3}$.

$$T(n, n) = O(n^{2/3})T(n^{1/3}, n^{2/3}) + O(n^{4/3})$$

Use duality + point location: $T(n^{1/3}, n^{2/3}) = O(n^{2/3} + n^{2/3} \log n)$.

$$T(n, n) = O(n^{4/3} \log n)$$

Applying cuttings to Hopcroft's problem [Chazelle, 1993]

Let $T(m, n)$ be the time to solve Hopcroft's problem with m points and n lines.

$$T(n, n) = O(r^2)T\left(\frac{n}{r^2}, \frac{n}{r}\right) + O(nr + n \log r)$$

Choose $r = n^{1/3}$.

$$T(n, n) = O(n^{2/3})T(n^{1/3}, n^{2/3}) + O(n^{4/3})$$

Use duality + point location: $T(n^{1/3}, n^{2/3}) = O(n^{2/3} + n^{2/3} \log n)$.

$$T(n, n) = O(n^{4/3} \log n)$$

Slightly better with $r = n^{1/3} \log^{1/3} n$ to get $O(n^{4/3} \log^{1/3} n)$ [Chazelle, 1993]

A (rederivation of) Matoušek's $n^{4/3}2^{O(\log^* n)}$ time algorithm

$$T(m,n) = O(r^2)T\left(\frac{m}{r^2}, \frac{n}{r}\right) + O(nr + m\log r) \qquad \text{and} \qquad T(m,n) = T(n,m)$$

A (rederivation of) Matoušek's $n^{4/3}2^{O(\log^* n)}$ time algorithm

$$T(m,n) = O(r^2)T\left(\frac{m}{r^2}, \frac{n}{r}\right) + O(nr + m\log r) \qquad \text{and} \qquad T(m,n) = T(n,m)$$

Apply our this recursion twice (with duality)!

$$T(n,n) = O(r^4)T\left(\frac{n}{r^3}, \frac{n}{r^3}\right) + O(nr\log r)$$

A (rederivation of) Matoušek's $n^{4/3}2^{O(\log^* n)}$ time algorithm

$$T(m,n) = O(r^2)T\left(\frac{m}{r^2}, \frac{n}{r}\right) + O(nr + m\log r) \qquad \text{and} \qquad T(m,n) = T(n,m)$$

Apply our this recursion twice (with duality)!

$$T(n,n) = O(r^4)T\left(\frac{n}{r^3}, \frac{n}{r^3}\right) + O(nr\log r)$$

Choose $r = \frac{n^{1/3}}{\log n}$, to make the non-recursive term $O(n^{4/3})$:

$$T(n,n) = O\left(\frac{n^{4/3}}{\log^4 n}\right) T\left(\log^3 n, \log^3 n\right) + O(n^{4/3})$$

A (rederivation of) Matoušek's $n^{4/3}2^{O(\log^* n)}$ time algorithm

$$T(m,n) = O(r^2)T\left(\frac{m}{r^2}, \frac{n}{r}\right) + O(nr + m\log r) \qquad \text{and} \qquad T(m,n) = T(n,m)$$

Apply our this recursion twice (with duality)!

$$T(n,n) = O(r^4)T\left(\frac{n}{r^3}, \frac{n}{r^3}\right) + O(nr\log r)$$

Choose $r = \frac{n^{1/3}}{\log n}$, to make the non-recursive term $O(n^{4/3})$:

$$T(n,n) = O\left(\frac{n^{4/3}}{\log^4 n}\right) T\left(\log^3 n, \log^3 n\right) + O(n^{4/3})$$

Solving this will give:

$$T(n,n) = O(n^{4/3}2^{O(\log^* n)})$$

Getting rid of extra factors?

Chazelle's approach:

$$T(n,n) = O(n^{2/3})T\left(n^{1/3}, n^{2/3}\right) + O(n^{4/3})$$

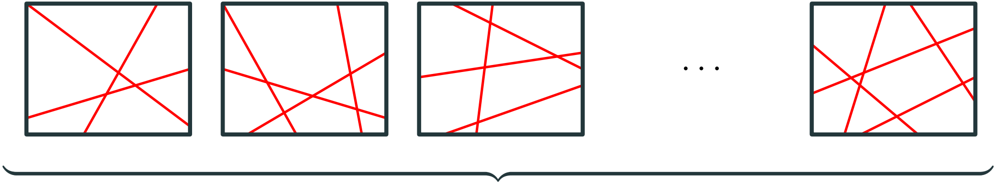
(Duality + point location) $\quad T\left(n^{1/3}, n^{2/3}\right) = O(n^{2/3} + n^{2/3}\log n)$

Getting rid of extra factors?

Chazelle's approach:

$$T(n,n) = O(n^{2/3})T\left(n^{1/3}, n^{2/3}\right) + O(n^{4/3})$$

(Duality + point location) $T\left(n^{1/3}, n^{2/3}\right) = O(n^{2/3} + n^{2/3}\log n)$



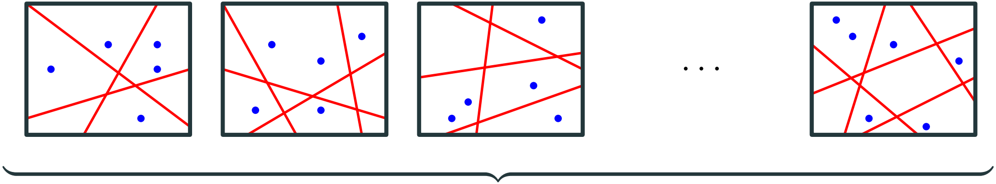
$O(n^{2/3})$ arrangements of $O(n^{1/3})$ lines

Getting rid of extra factors?

Chazelle's approach:

$$T(n,n) = O(n^{2/3})T\left(n^{1/3}, n^{2/3}\right) + O(n^{4/3})$$

(Duality + point location) $\quad T\left(n^{1/3}, n^{2/3}\right) = O(n^{2/3} + n^{2/3}\log n)$



$O(n^{2/3})$ arrangements of $O(n^{1/3})$ lines and $O(n^{2/3})$ points.

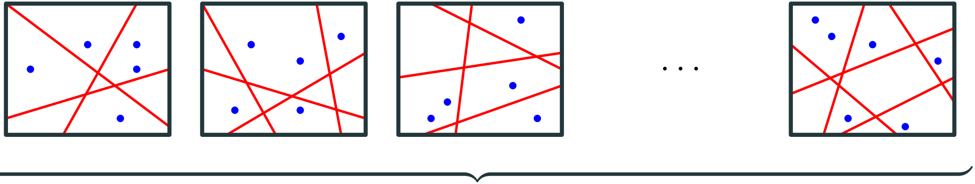
$O(n^{4/3})$ point locations queries total! $\Omega(\log n)$ lower bound for doing a single point query.

Getting rid of extra factors?

Chazelle's approach:

$$T(n,n) = O(n^{2/3})T\left(n^{1/3}, n^{2/3}\right) + O(n^{4/3})$$

(Duality + point location) $\quad T\left(n^{1/3}, n^{2/3}\right) = O(n^{2/3} + n^{2/3}\log n)$



$O(n^{2/3})$ arrangements of $O(n^{1/3})$ lines and $O(n^{2/3})$ points.

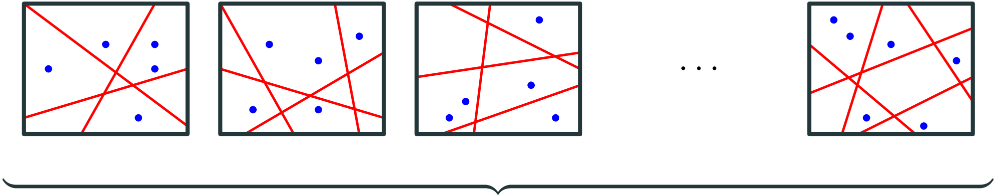
$O(n^{4/3})$ point locations queries total! $\Omega(\log n)$ lower bound for doing a single point query. Can we do this faster than $O(n^{4/3}\log n)$?

Answer

Yes, we can!

Answer

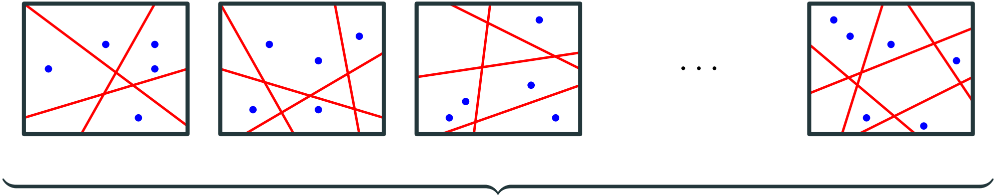
Yes, we can!



$O(n^{2/3})$ arrangements of $O(n^{1/3})$ lines and $O(n^{2/3})$ points.

Answer

Yes, we can!



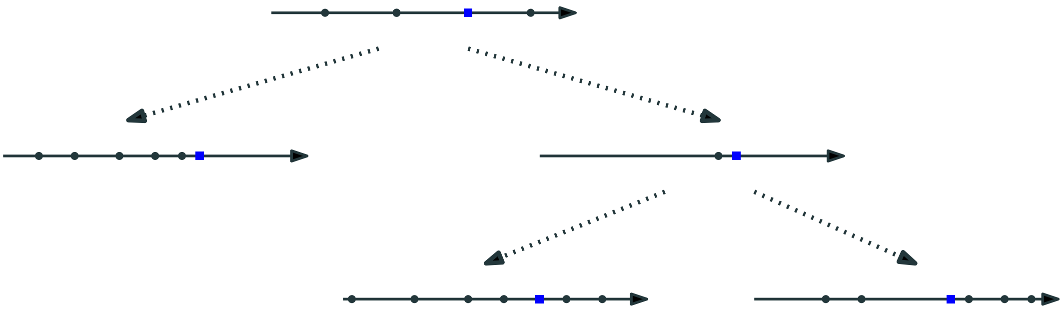
$O(n^{2/3})$ arrangements of $O(n^{1/3})$ lines and $O(n^{2/3})$ points.

Point location of n (dual) points in (average of) $O(n^{1/3})$ (dual) arrangements.

Outline

Fractional cascading in 1d lists [Chazelle, Guibas, 1986]

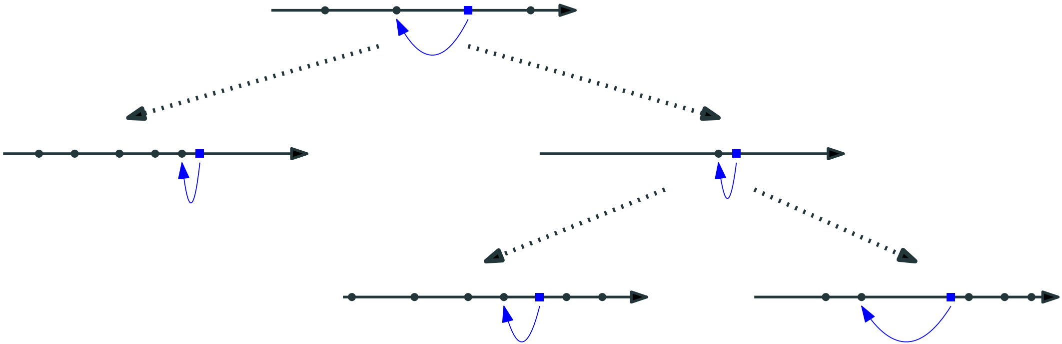
Suppose we're given a constant degree tree T of lists of size z and a query point p.



Fractional cascading in 1d lists [Chazelle, Guibas, 1986]

Suppose we're given a constant degree tree T of lists of size z and a query point p.

We can find all predecessors of p in time $O(|T| \log z)$ with $O(|T|)$ binary searches.

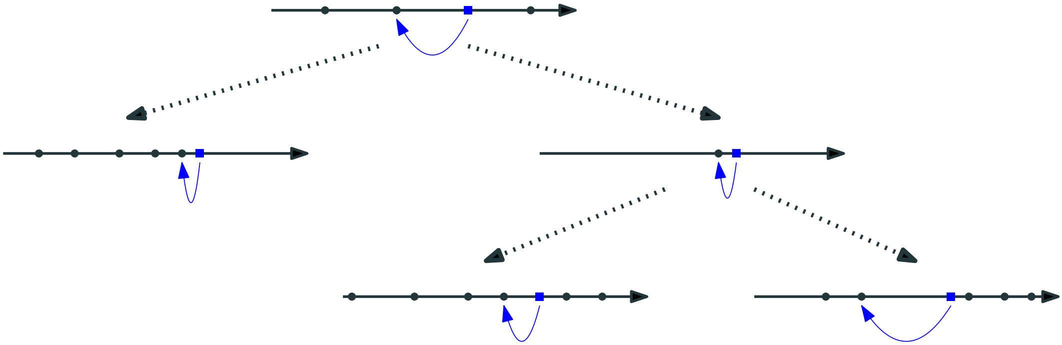


Fractional cascading in 1d lists [Chazelle, Guibas, 1986]

Suppose we're given a constant degree tree T of lists of size z and a query point p.

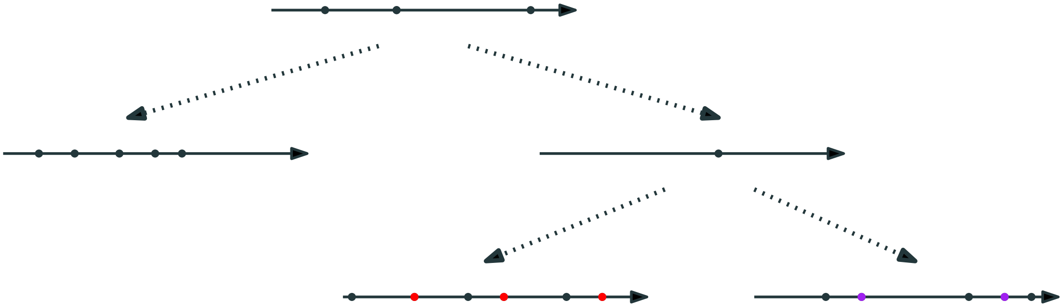
We can find all predecessors of p in time $O(|T| \log z)$ with $O(|T|)$ binary searches.

Fractional cascading finds all predecessors of p in time $O(|T| + \log z)$, this is amortized $O(1)$ per list.



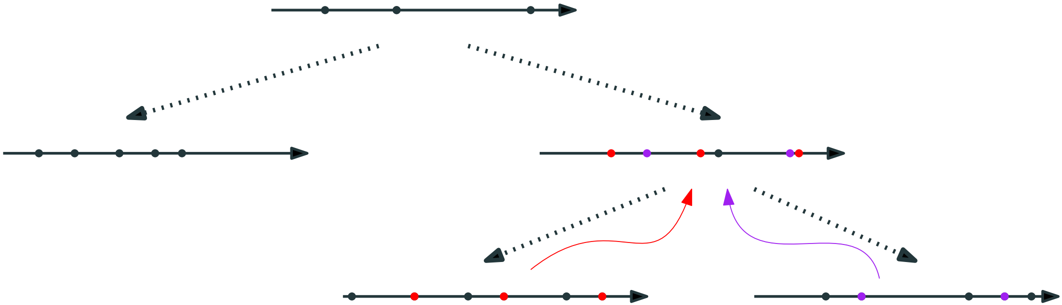
Fractional cascading in 1d lists

Idea: Pass fraction $1/c$ of elements from child lists to parent lists.



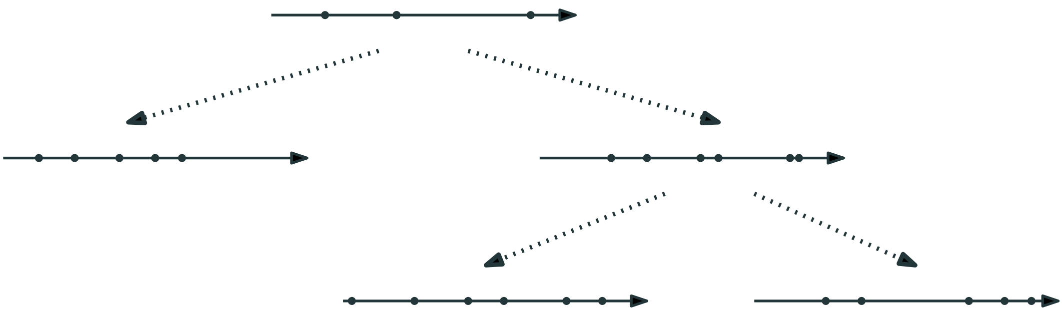
Fractional cascading in 1d lists

Idea: Pass fraction $1/c$ of elements from child lists to parent lists.



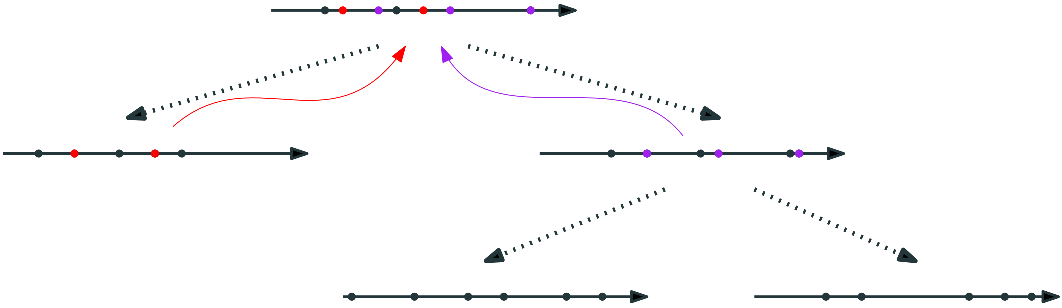
Fractional cascading in 1d lists

Idea: Pass fraction $1/c$ of elements from child lists to parent lists.



Fractional cascading in 1d lists

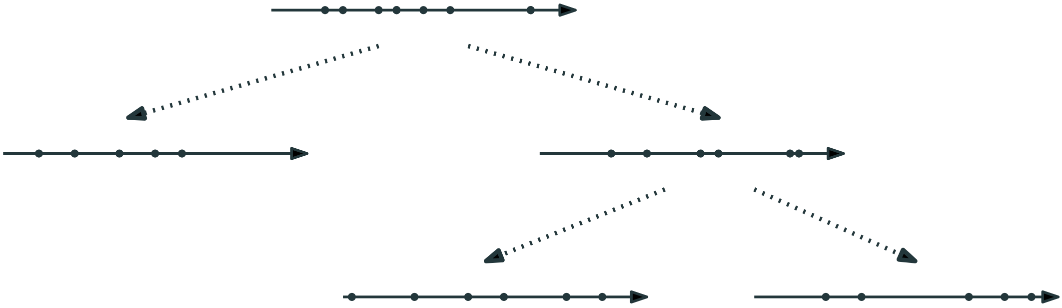
Idea: Pass fraction $1/c$ of elements from child lists to parent lists.



Fractional cascading in 1d lists

Idea: Pass fraction $1/c$ of elements from child lists to parent lists.

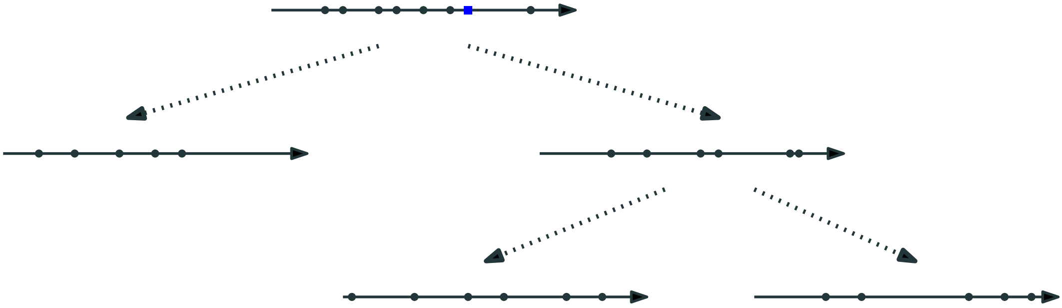
Can handle queries with pointers in $O(1)$ after an initial binary search.



Fractional cascading in 1d lists

Idea: Pass fraction $1/c$ of elements from child lists to parent lists.

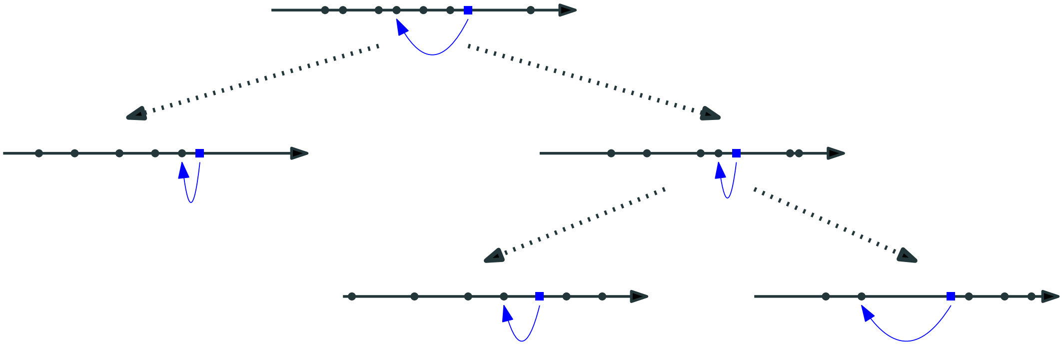
Can handle queries with pointers in $O(1)$ after an initial binary search.



Fractional cascading in 1d lists

Idea: Pass fraction $1/c$ of elements from child lists to parent lists.

Can handle queries with pointers in $O(1)$ after an initial binary search.



Fractional cascading in 2D?

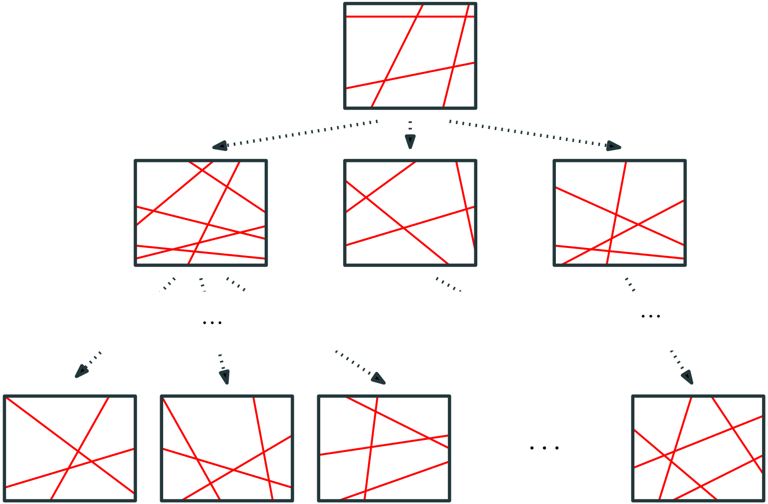
In 2004, Chazelle and Liu proved that fractional cascading in 2d planar subdivisions needs $\Omega(N^2)$ preprocessing.

Fractional cascading in 2D?

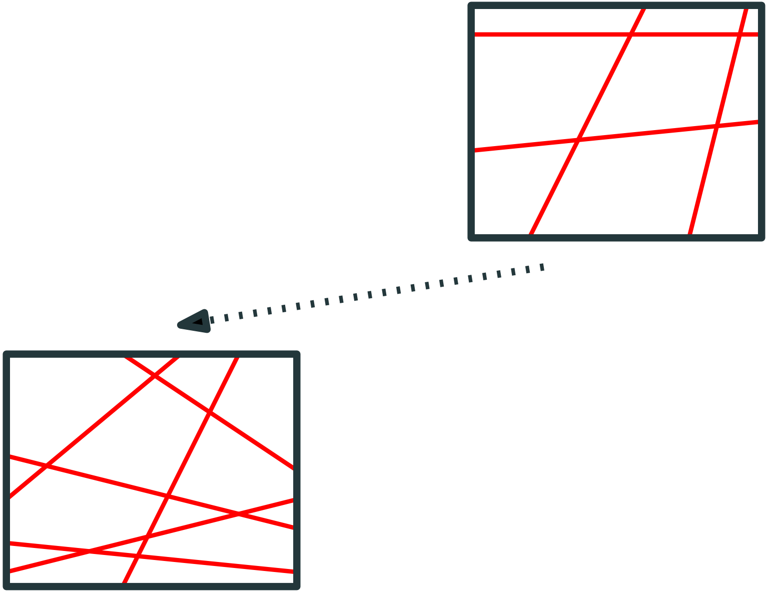
In 2004, Chazelle and Liu proved that fractional cascading in 2d planar subdivisions needs $\Omega(N^2)$ preprocessing.

However, not general planar subdivisions, these are arrangements of lines!

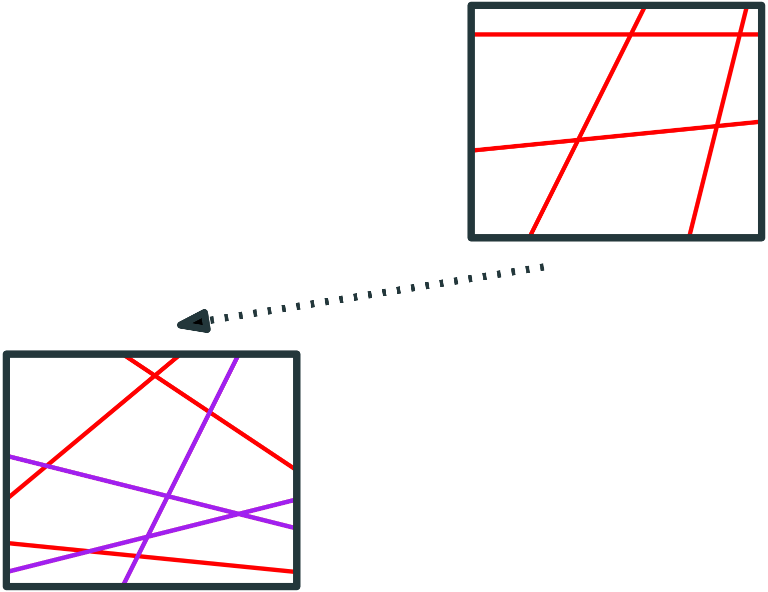
Fractional cascading of line arrangements



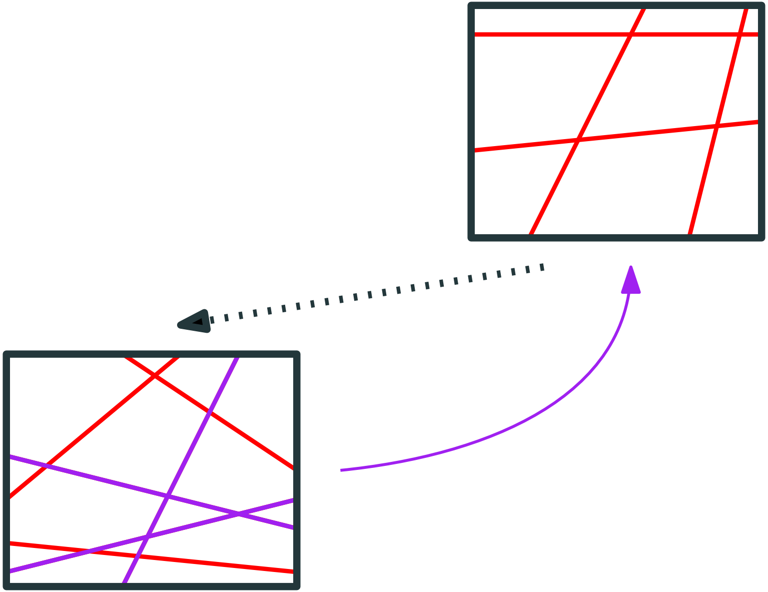
Fractional cascading of line arrangements



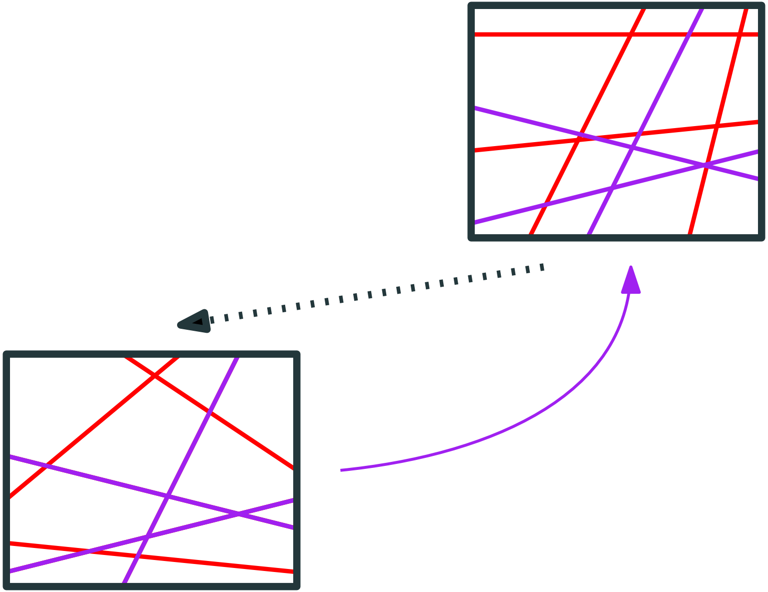
Fractional cascading of line arrangements



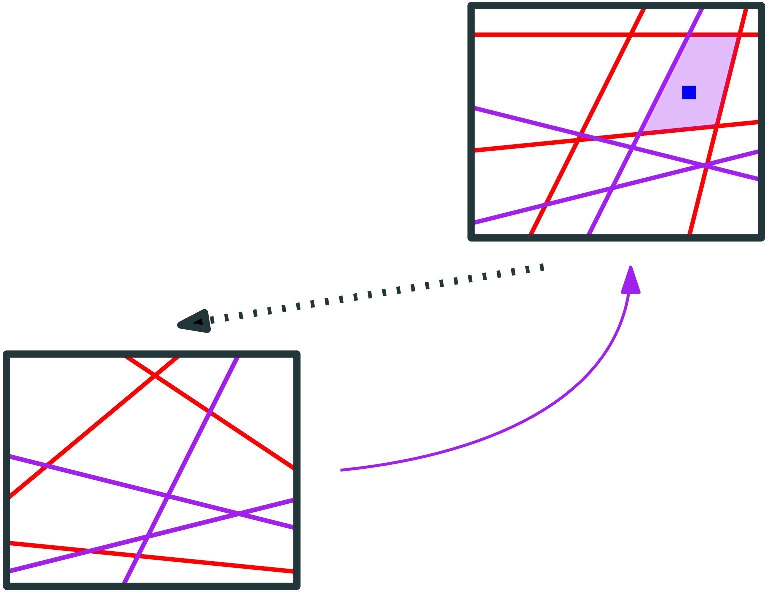
Fractional cascading of line arrangements



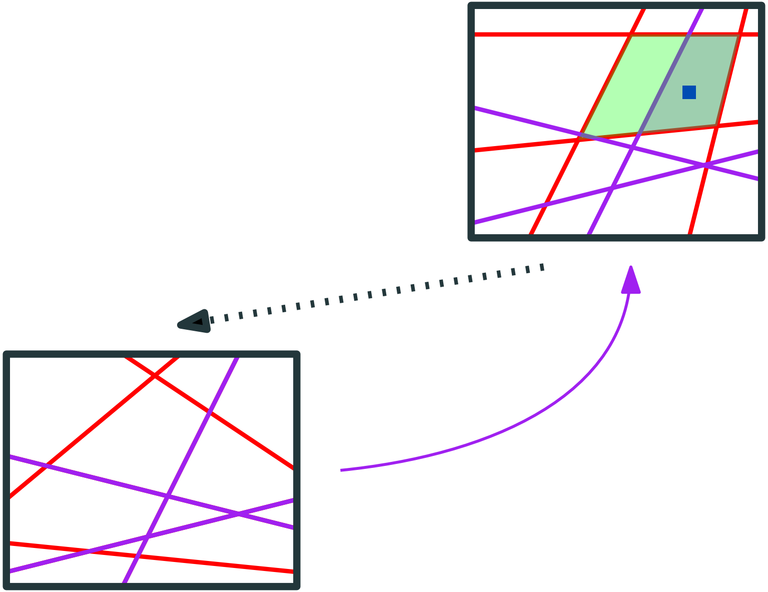
Fractional cascading of line arrangements



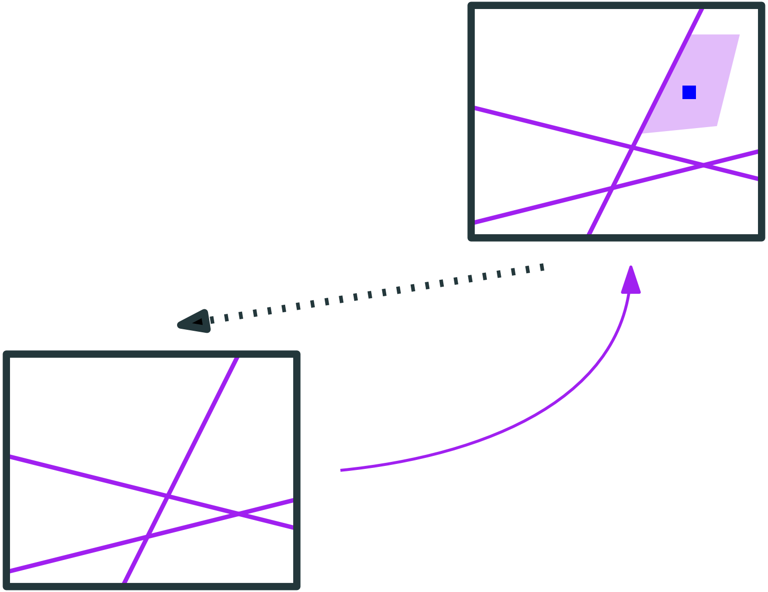
Fractional cascading of line arrangements



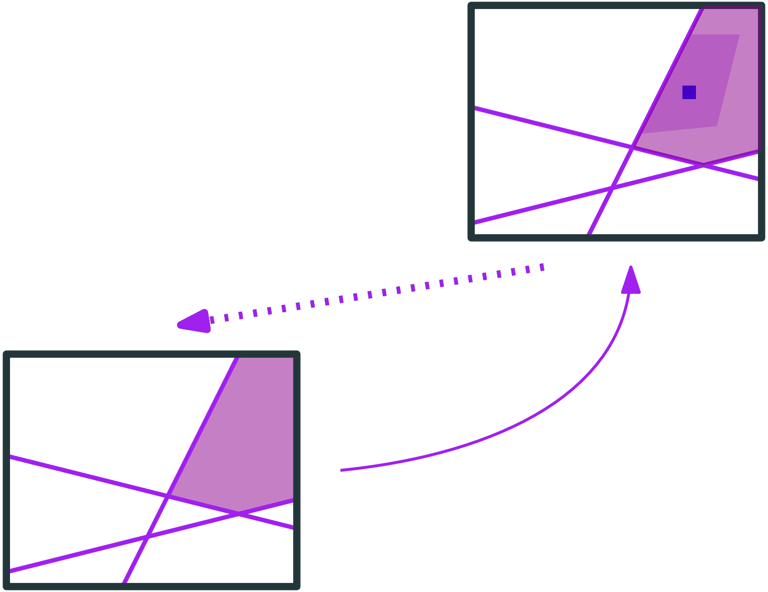
Fractional cascading of line arrangements



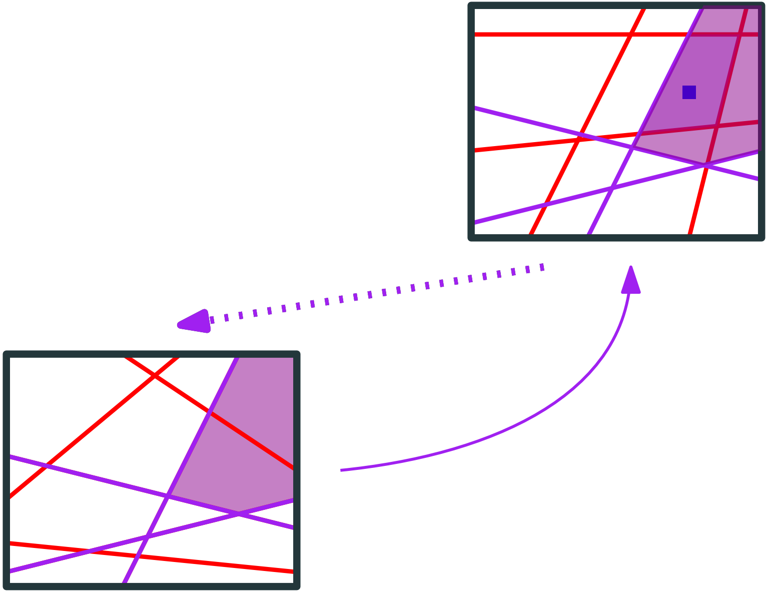
Fractional cascading of line arrangements



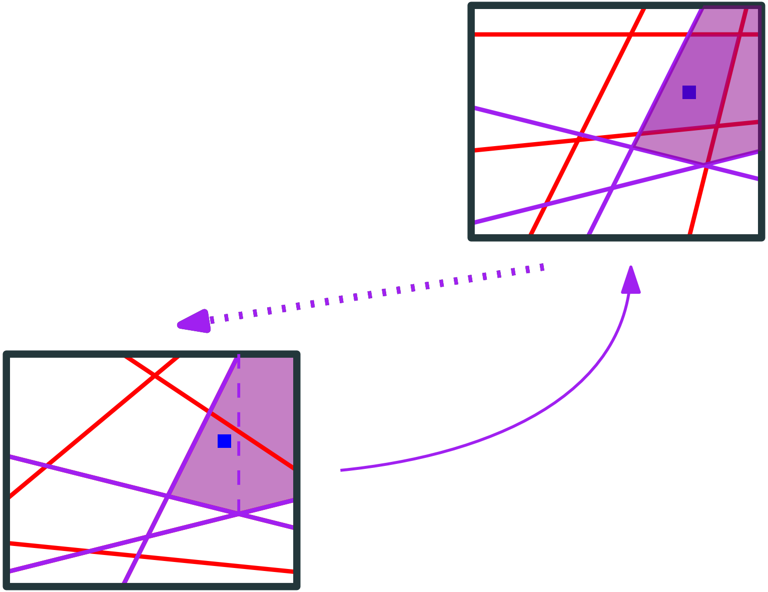
Fractional cascading of line arrangements



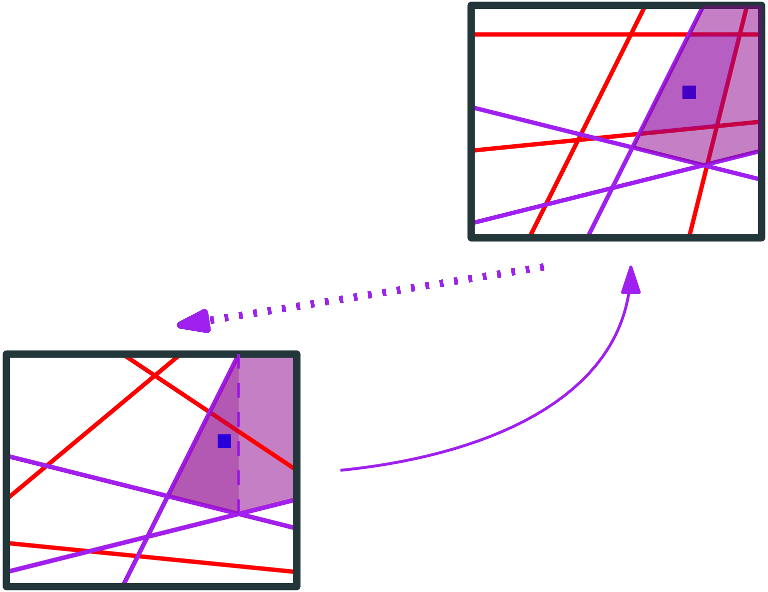
Fractional cascading of line arrangements



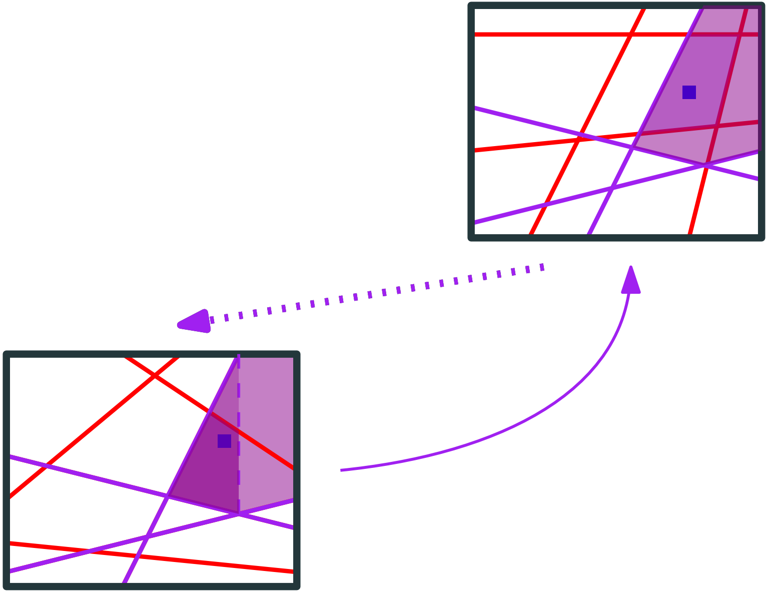
Fractional cascading of line arrangements



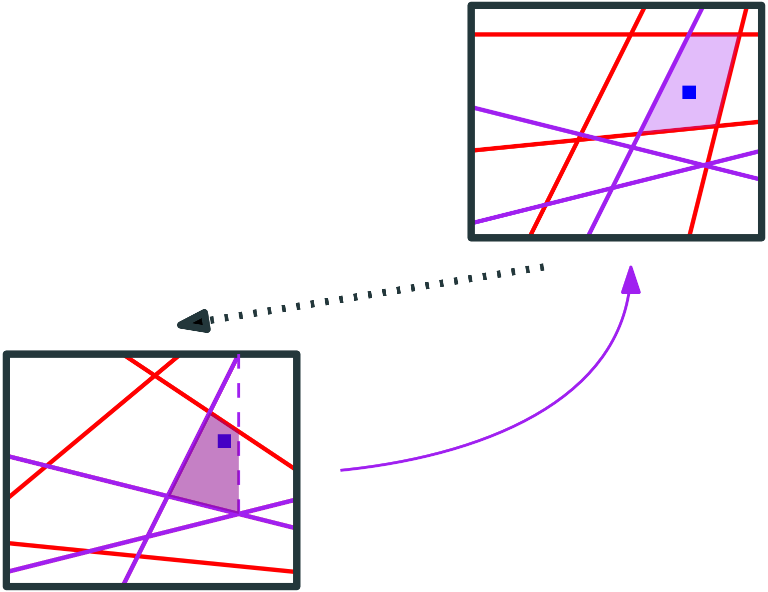
Fractional cascading of line arrangements



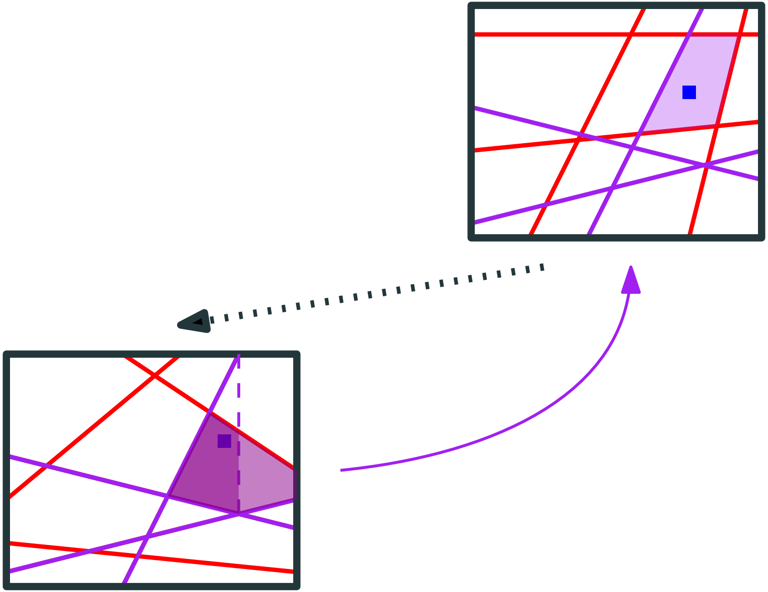
Fractional cascading of line arrangements



Fractional cascading of line arrangements

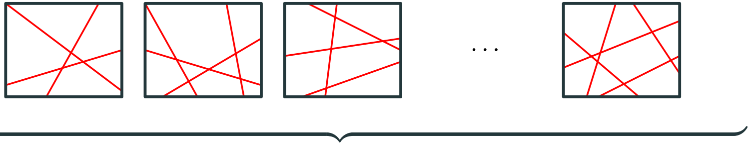


Fractional cascading of line arrangements



Fractional cascading of line arrangements

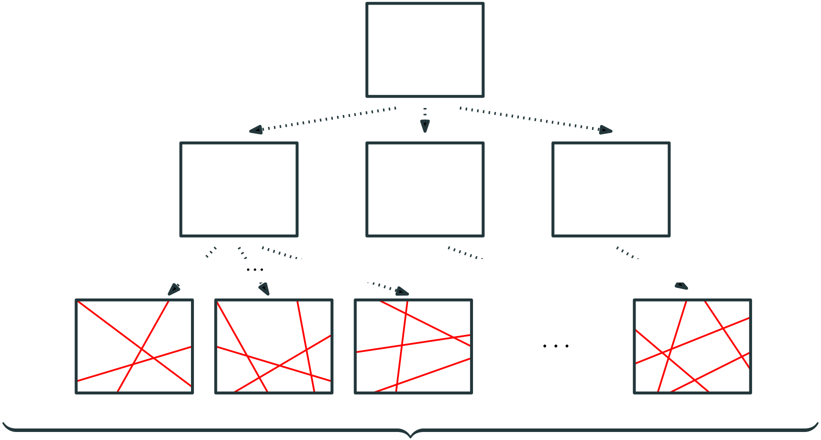
Where is our tree?



$O(n^{2/3})$ arrangements of $O(n^{1/3})$ lines and $O(n^{2/3})$ points.

Fractional cascading of line arrangements

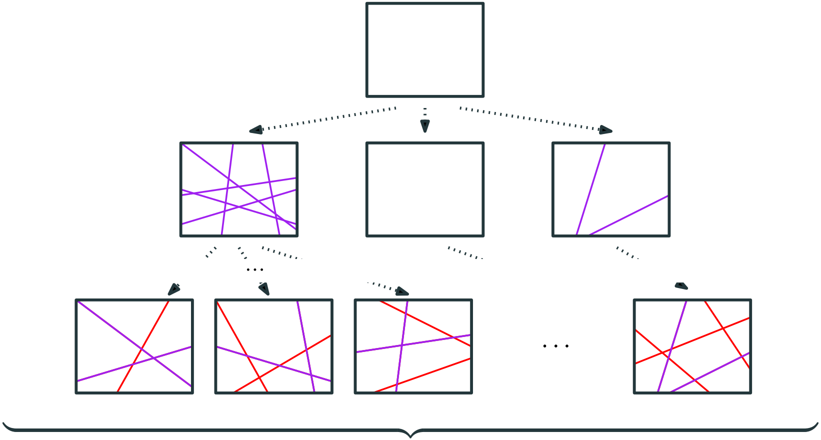
Where is our tree? From the cutting, as they give a hierarchical tree structure!



$O(n^{2/3})$ arrangements of $O(n^{1/3})$ lines and $O(n^{2/3})$ points.

Fractional cascading of line arrangements

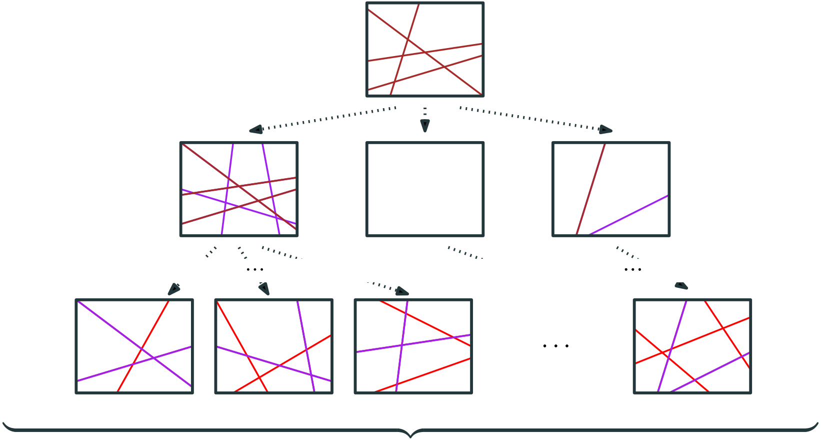
Where is our tree? From the cutting, as they give a hierarchical tree structure!



$O(n^{2/3})$ arrangements of $O(n^{1/3})$ lines and $O(n^{2/3})$ points.

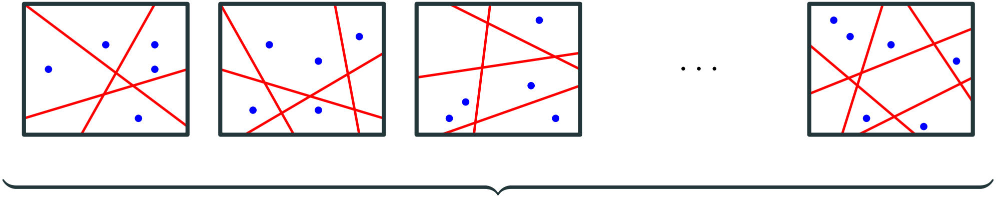
Fractional cascading of line arrangements

Where is our tree? From the cutting, as they give a hierarchical tree structure!



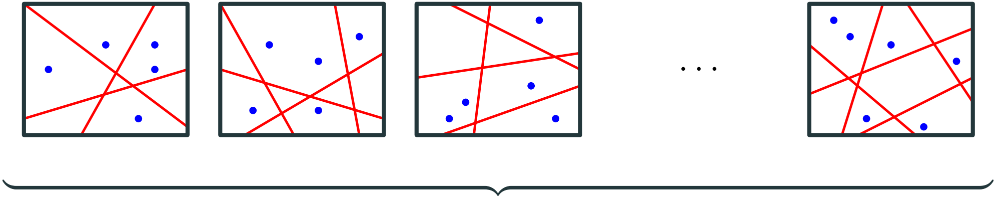
$O(n^{2/3})$ arrangements of $O(n^{1/3})$ lines and $O(n^{2/3})$ points.

Back to Hopcroft



$O(n^{2/3})$ arrangements of $O(n^{1/3})$ lines and $O(n^{2/3})$ points.

Back to Hopcroft



$O(n^{2/3})$ arrangements of $O(n^{1/3})$ lines and $O(n^{2/3})$ points.

$O(n^{4/3})$ time to do $O(n^{4/3})$ point location queries!

Remarks on Fractional Cascading of Lines

Limitations of 2D fractional cascading of lines:

Remarks on Fractional Cascading of Lines

Limitations of 2D fractional cascading of lines:

- Only works in 2D (relies on vertical decompositions)
- Randomized

Remarks on Fractional Cascading of Lines

Limitations of 2D fractional cascading of lines:

- Only works in 2D (relies on vertical decompositions)
- Randomized

For higher dimensions, we need a different approach.

Remarks on Fractional Cascading of Lines

Limitations of 2D fractional cascading of lines:

- Only works in 2D (relies on vertical decompositions)
- Randomized

For higher dimensions, we need a different approach.

Main idea: Easier to avoid logs in the decision tree model.

Outline

Low depth decision trees implies faster runtimes

Claim: If Hopcroft's problem has $O(n^{4/3})$ decision tree complexity, there exists an $O(n^{4/3})$ algorithm for Hopcroft's problem.

Low depth decision trees implies faster runtimes

Claim: If Hopcroft's problem has $O(n^{4/3})$ decision tree complexity, there exists an $O(n^{4/3})$ algorithm for Hopcroft's problem.

$$T(n,n) = O\left(\frac{n^{4/3}}{\log^4 n}\right) T\left(\log^3 n, \log^3 n\right) + O(n^{4/3})$$

Low depth decision trees implies faster runtimes

Claim: If Hopcroft's problem has $O(n^{4/3})$ decision tree complexity, there exists an $O(n^{4/3})$ algorithm for Hopcroft's problem.

$$T(n,n) = O\left(\frac{n^{4/3}}{\log^4 n}\right) T\left(\log^3 n, \log^3 n\right) + O(n^{4/3})$$

Repeating gives $O(n^{4/3}/(\log\log\log n)^4)$ subproblems of size $b = O((\log\log\log n)^3)$.

Low depth decision trees implies faster runtimes

Claim: If Hopcroft's problem has $O(n^{4/3})$ decision tree complexity, there exists an $O(n^{4/3})$ algorithm for Hopcroft's problem.

$$T(n,n) = O\left(\frac{n^{4/3}}{\log^4 n}\right) T\left(\log^3 n, \log^3 n\right) + O(n^{4/3})$$

Repeating gives $O(n^{4/3}/(\log\log\log n)^4)$ subproblems of size $b = O((\log\log\log n)^3)$.

We can afford to build a decision tree T because b is very small.

Low depth decision trees implies faster runtimes

Claim: If Hopcroft's problem has $O(n^{4/3})$ decision tree complexity, there exists an $O(n^{4/3})$ algorithm for Hopcroft's problem.

$$T(n,n) = O\left(\frac{n^{4/3}}{\log^4 n}\right) T\left(\log^3 n, \log^3 n\right) + O(n^{4/3})$$

Repeating gives $O(n^{4/3}/(\log\log\log n)^4)$ subproblems of size $b = O((\log\log\log n)^3)$.

We can afford to build a decision tree T because b is very small.

Low depth decision trees implies faster runtimes

Claim: If Hopcroft's problem has $O(n^{4/3})$ decision tree complexity, there exists an $O(n^{4/3})$ algorithm for Hopcroft's problem.

$$T(n,n) = O\left(\frac{n^{4/3}}{\log^4 n}\right) T\left(\log^3 n, \log^3 n\right) + O(n^{4/3})$$

Repeating gives $O(n^{4/3}/(\log\log\log n)^4)$ subproblems of size $b = O((\log\log\log n)^3)$.

We can afford to build a decision tree T because b is very small.

This is not new, mentioned in [Matoušek, 1993], useful for 3SUM and APSP.

(Warmup) Sorting with decision trees [Fredman, 1976]

(Warmup) Sorting with decision trees [Fredman, 1976]

Problem: Given a set $X = \{x_1, ..., x_n\}$ and a set $Y = \{y_1, ..., y_n\}$, sort the set:

$$X + Y := \{x_i + y_j \mid x_i \in X, y_j \in Y\}$$

(Warmup) Sorting with decision trees [Fredman, 1976]

Problem: Given a set $X = \{x_1, ..., x_n\}$ and a set $Y = \{y_1, ..., y_n\}$, sort the set:

$$X + Y := \{x_i + y_j \mid x_i \in X, y_j \in Y\}$$

Theorem [Fredman, '76] Sorting $X + Y$ can be done in $O(n^2)$ comparisons.

Sorting with decision trees [Fredman, 1976]

Sorting with decision trees [Fredman, 1976]

View input as $\mathbf{x} = (x_1, \ldots, x_n, y_1, \ldots, y_n) \in \mathbb{R}^N$ for $N = 2n$.

Sorting with decision trees [Fredman, 1976]

View input as $\mathbf{x} = (x_1, ..., x_n, y_1, ..., y_n) \in \mathbb{R}^N$ for $N = 2n$.

If we knew the outcomes of every comparison:

- $x_i + y_j < x_k + y_h$ or
- $x_i + y_j = x_k + y_h$ or
- $x_i + y_j > x_k + y_h$.

Then we would could sort $X + Y$.

Sorting with decision trees [Fredman, 1976]

View input as $\mathbf{x} = (x_1, ..., x_n, y_1, ..., y_n) \in \mathbb{R}^N$ for $N = 2n$.

If we knew the outcomes of every comparison:

- $x_i + y_j < x_k + y_h$ or
- $x_i + y_j = x_k + y_h$ or
- $x_i + y_j > x_k + y_h$.

Then we would could sort $X + Y$.

Each comparison is a hyperplane H in $\mathbb{R}^N$. It suffices to know where $\mathbf{x}$ is.

Sorting with decision trees [Fredman, 1976]

View input as $\mathbf{x} = (x_1, ..., x_n, y_1, ..., y_n) \in \mathbb{R}^N$ for $N = 2n$.

If we knew the outcomes of every comparison:

- $x_i + y_j < x_k + y_h$ or
- $x_i + y_j = x_k + y_h$ or
- $x_i + y_j > x_k + y_h$.

Then we would could sort $X + Y$.

Each comparison is a hyperplane H in $\mathbb{R}^N$. It suffices to know where $\mathbf{x}$ is.

$O(n^4)$ such hyperplanes, can show there are $O(n^{8n})$ different cells.

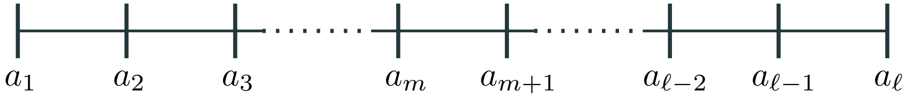
Sorting with decision trees [Fredman, 1976]

Idea: Insertion sort + weighted binary search.

Sorting with decision trees [Fredman, 1976]

Idea: Insertion sort + weighted binary search.

We have sorted $a_1, \ldots, a_\ell \in X + Y$. Want to insert the next element $q \in X + Y$ in.

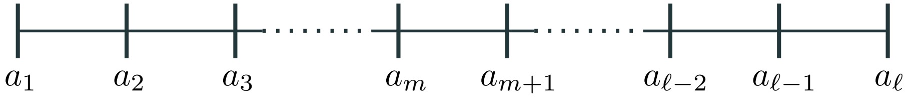


Sorting with decision trees [Fredman, 1976]

Idea: Insertion sort + weighted binary search.

We have sorted $a_1, \ldots, a_\ell \in X + Y$. Want to insert the next element $q \in X + Y$ in.

Let Π denote the set of cells that are consistent with this ordering.

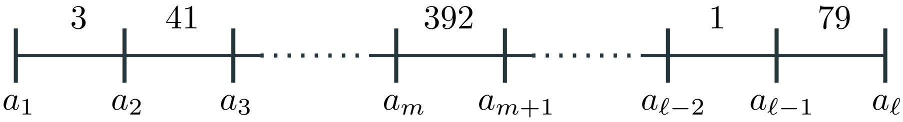


Sorting with decision trees [Fredman, 1976]

Idea: Insertion sort + weighted binary search.

We have sorted $a_1, \ldots, a_\ell \in X + Y$. Want to insert the next element $q \in X + Y$ in.

Let Π denote the set of cells that are consistent with this ordering.

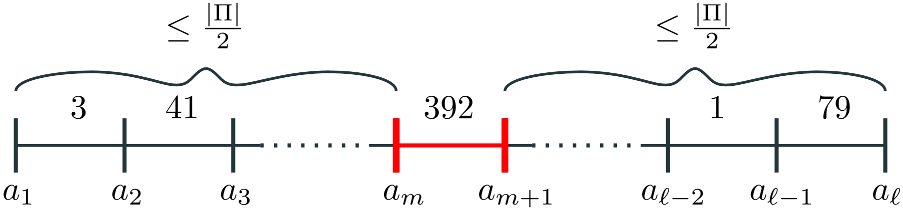


Sorting with decision trees [Fredman, 1976]

Idea: Insertion sort + weighted binary search.

We have sorted $a_1, \dots, a_\ell \in X + Y$. Want to insert the next element $q \in X + Y$ in.

Let Π denote the set of cells that are consistent with this ordering.

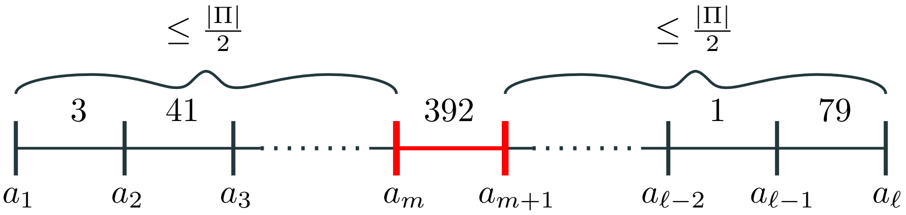


Sorting with decision trees [Fredman, 1976]

Idea: Insertion sort + weighted binary search.

We have sorted $a_1, \ldots, a_\ell \in X + Y$. Want to insert the next element $q \in X + Y$ in.

Let Π denote the set of cells that are consistent with this ordering.



Comparing q with a_m and a_{m+1} results in one of the following:

(1) We know that q lies between a_m and a_{m+1}.

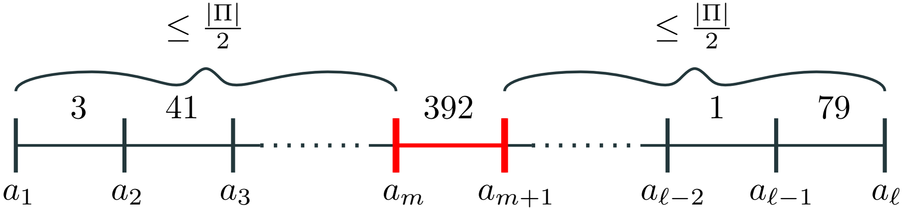
(2) The number of consistent cells is halved.

Sorting with decision trees [Fredman, 1976]

Idea: Insertion sort + weighted binary search.

We have sorted $a_1, \ldots, a_\ell \in X + Y$. Want to insert the next element $q \in X + Y$ in.

Let Π denote the set of cells that are consistent with this ordering.



Comparing q with a_m and a_{m+1} results in one of the following:

(1) We know that q lies between a_m and a_{m+1}. $O(n^2)$

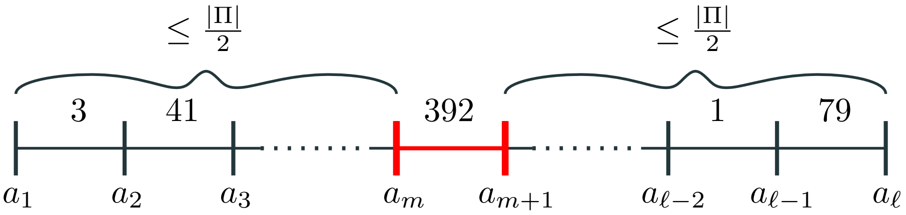
(2) The number of consistent cells is halved.

Sorting with decision trees [Fredman, 1976]

Idea: Insertion sort + weighted binary search.

We have sorted $a_1, \ldots, a_\ell \in X + Y$. Want to insert the next element $q \in X + Y$ in.

Let Π denote the set of cells that are consistent with this ordering.



Comparing q with a_m and a_{m+1} results in one of the following:

(1) We know that q lies between a_m and a_{m+1}. $O(n^2)$

(2) The number of consistent cells is halved. $O(n \log n)$

Generalizing Fredman's algorithm for Hopcroft's Problem (Sketch)

Idea: Fredman's trick extends to point location.

Generalizing Fredman's algorithm for Hopcroft's Problem (Sketch)

Idea: Fredman's trick extends to point location.

Goal: Do $O(n^{4/3})$ point location queries that arose from n points and n lines.

Generalizing Fredman's algorithm for Hopcroft's Problem (Sketch)

Idea: Fredman's trick extends to point location.

Goal: Do $O(n^{4/3})$ point location queries that arose from n points and n lines.

- Algebraic decision tree makes constant-degree algebraic comparisons of the form $x \in \gamma$ for semi-algebraic $\gamma \in \Gamma$, where $|\Gamma| = O(n^c)$.

Generalizing Fredman's algorithm for Hopcroft's Problem (Sketch)

Idea: Fredman's trick extends to point location.

Goal: Do $O(n^{4/3})$ point location queries that arose from n points and n lines.

- Algebraic decision tree makes constant-degree algebraic comparisons of the form $x \in \gamma$ for semi-algebraic $\gamma \in \Gamma$, where $|\Gamma| = O(n^c)$.
- Viewed in $\mathbb{R}^N$, we can pre-compute arrangement $\mathcal{A}(\Gamma)$.

Generalizing Fredman's algorithm for Hopcroft's Problem (Sketch)

Idea: Fredman's trick extends to point location.

Goal: Do $O(n^{4/3})$ point location queries that arose from n points and n lines.

- Algebraic decision tree makes constant-degree algebraic comparisons of the form $x \in \gamma$ for semi-algebraic $\gamma \in \Gamma$, where $|\Gamma| = O(n^c)$.
- Viewed in $\mathbb{R}^N$, we can pre-compute arrangement $\mathcal{A}(\Gamma)$.
- In $\mathcal{A}(\Gamma)$, we get cells Π that correspond to result of comparisons.

Generalizing Fredman's algorithm for Hopcroft's Problem (Sketch)

Idea: Fredman's trick extends to point location.

Goal: Do $O(n^{4/3})$ point location queries that arose from n points and n lines.

- Algebraic decision tree makes constant-degree algebraic comparisons of the form $x \in \gamma$ for semi-algebraic $\gamma \in \Gamma$, where $|\Gamma| = O(n^c)$.
- Viewed in $\mathbb{R}^N$, we can pre-compute arrangement $\mathcal{A}(\Gamma)$.
- In $\mathcal{A}(\Gamma)$, we get cells Π that correspond to result of comparisons.
- Milnor-Thom theorem gives $|\Pi| := \#$ of cells $\leq |\Gamma|^N = n^{O(n)}$.

Generalizing Fredman's algorithm for Hopcroft's Problem (Sketch)

Idea: Fredman's trick extends to point location.

Goal: Do $O(n^{4/3})$ point location queries that arose from n points and n lines.

- Algebraic decision tree makes constant-degree algebraic comparisons of the form $x \in \gamma$ for semi-algebraic $\gamma \in \Gamma$, where $|\Gamma| = O(n^c)$.
- Viewed in $\mathbb{R}^N$, we can pre-compute arrangement $\mathcal{A}(\Gamma)$.
- In $\mathcal{A}(\Gamma)$, we get cells Π that correspond to result of comparisons.
- Milnor-Thom theorem gives $|\Pi| :=$ # of cells $\leq |\Gamma|^N = n^{O(n)}$.
- As we make γ-comparisons, number of cells consistent with result decreases OR we successfully do a point location.

Generalizing Fredman's algorithm for Hopcroft's Problem (Sketch)

Idea: Fredman's trick extends to point location.

Goal: Do $O(n^{4/3})$ point location queries that arose from n points and n lines.

- Algebraic decision tree makes constant-degree algebraic comparisons of the form $x \in \gamma$ for semi-algebraic $\gamma \in \Gamma$, where $|\Gamma| = O(n^c)$.
- Viewed in $\mathbb{R}^N$, we can pre-compute arrangement $\mathcal{A}(\Gamma)$.
- In $\mathcal{A}(\Gamma)$, we get cells Π that correspond to result of comparisons.
- Milnor-Thom theorem gives $|\Pi| :=$ # of cells $\leq |\Gamma|^N = n^{O(n)}$.
- As we make γ-comparisons, number of cells consistent with result decreases OR we successfully do a point location.
- To find the right γ to compare with, can use hierarchical cutting tree (and use the weighted centroid).

Outline

Introduction

Approach I - Fractional Cascading

Approach II - Algebraic Decision Trees

Conclusion

Final Remarks

- Improving runtime of Hopcroft's problem cleans up runtimes for many problems.

Final Remarks

- Improving runtime of Hopcroft's problem cleans up runtimes for many problems.
- Approach I extends to online 2D data structures for halfspace range counting.

Final Remarks

- Improving runtime of Hopcroft's problem cleans up runtimes for many problems.
- Approach I extends to online 2D data structures for halfspace range counting.
- Approach II works for shallow cuttings.

Final Remarks

- Improving runtime of Hopcroft's problem cleans up runtimes for many problems.
- Approach I extends to online 2D data structures for halfspace range counting.
- Approach II works for shallow cuttings.

Open Questions

Conclusion

Final Remarks

- Improving runtime of Hopcroft's problem cleans up runtimes for many problems.
- Approach I extends to online 2D data structures for halfspace range counting.
- Approach II works for shallow cuttings.

Open Questions

- Is there an analogue of our fractional cascading approach for higher dimensions?

Conclusion

Final Remarks

- Improving runtime of Hopcroft's problem cleans up runtimes for many problems.
- Approach I extends to online 2D data structures for halfspace range counting.
- Approach II works for shallow cuttings.

Open Questions

- Is there an analogue of our fractional cascading approach for higher dimensions?
- Are there other problems where we can improve decision tree complexity in this way and result in faster algorithms?

Thanks for listening!