Mihai's Work in Computational Geometry

Timothy Chan
School of CS
U of Waterloo

Talk Outline

1. Point Location [C.&P., FOCS'06]

2. Offline Point Location [C.&P., STOC'07]

3. Offline Orthogonal Range Counting [C.&P., SODA'10]

4. Orthogonal Range Reporting [C.&Larsen&P., SoCG'11]

Talk Outline

1. Point Location [C.&P., FOCS'06]

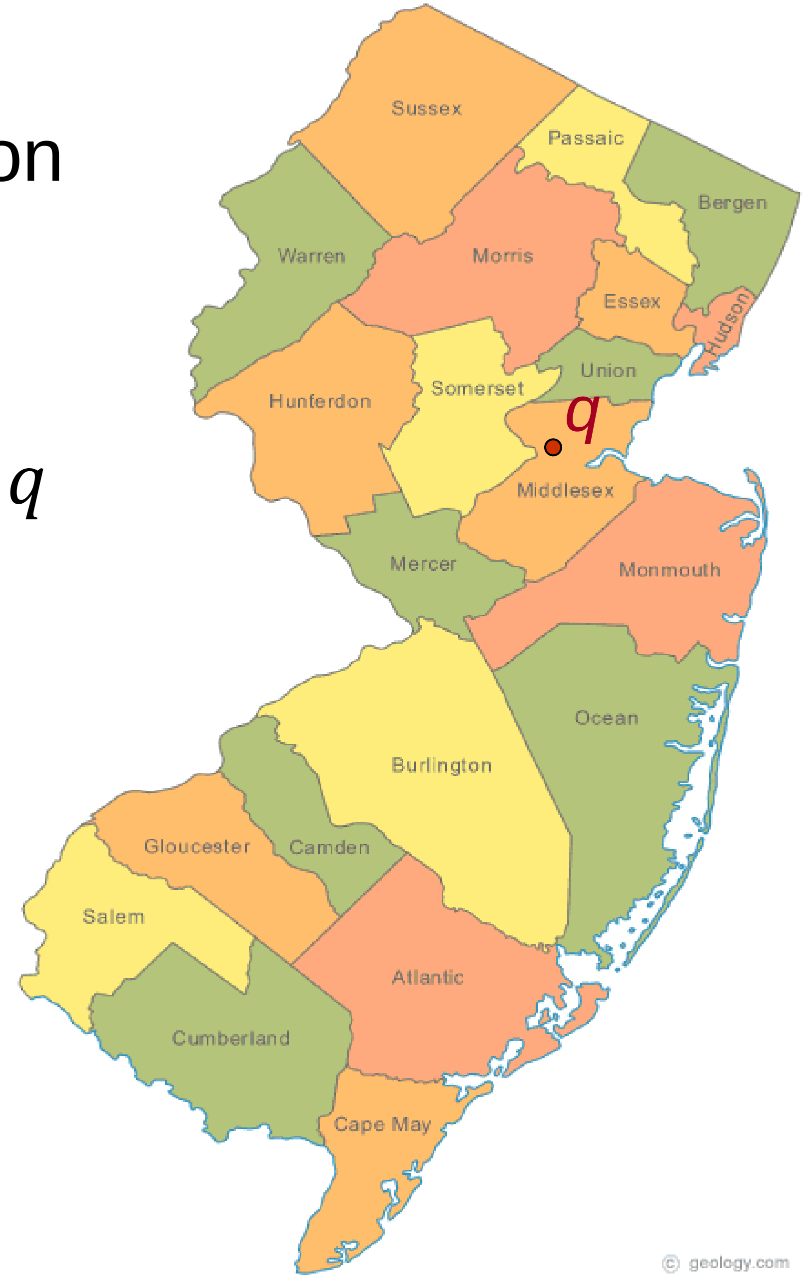
2. Offline Point Location [C.&P., STOC'07]

3. Offline Orthogonal Range Counting [C.&P., SODA'10]

4. Orthogonal Range Reporting [C.&Larsen&P., SoCG'11]

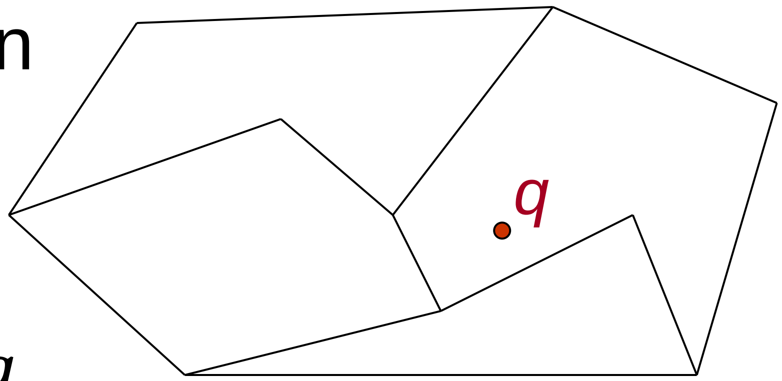
2D Point Location Problem

- Preprocess a planar subdivision with n line segments s.t.
  given query point q,
  find which region contains q



2D Point Location Problem

- Preprocess a planar subdivision with n line segments s.t. given query point q, find which region contains q

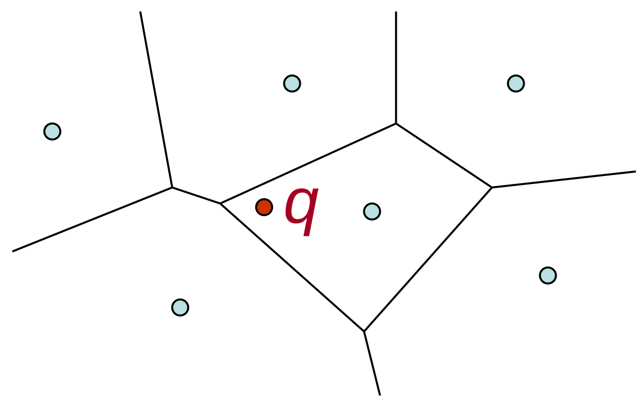


- Standard result in CG: $O(n)$ space, $O(\lg n)$ query time

[Dobkin&Lipton'76, Lee&Preparata'77, Lipton&Tarjan'77*, Kirkpatrick'83*, Preparata'83, Edelsbrunner&Guibas&Stolfi'86*, Cole'86, Sarnak&Tarjan'86*, Mulmuley'90*, ...]

2D Point Location Problem

- Applications:
  - 2D nearest neighbor search in $O(n)$ space, $O(\lg n)$ time



  - explains why many CG problems, e.g., 2D Voronoi diagram, 3D convex hull, 2D line segment intersection, etc., etc., etc. have $O(n \lg n)$ alg'ms

2D Point Location Problem

- The surprise:
  - The problem can be solved in sublogarithmic time!

    ... in the word RAM (or “transdichotomous”) model

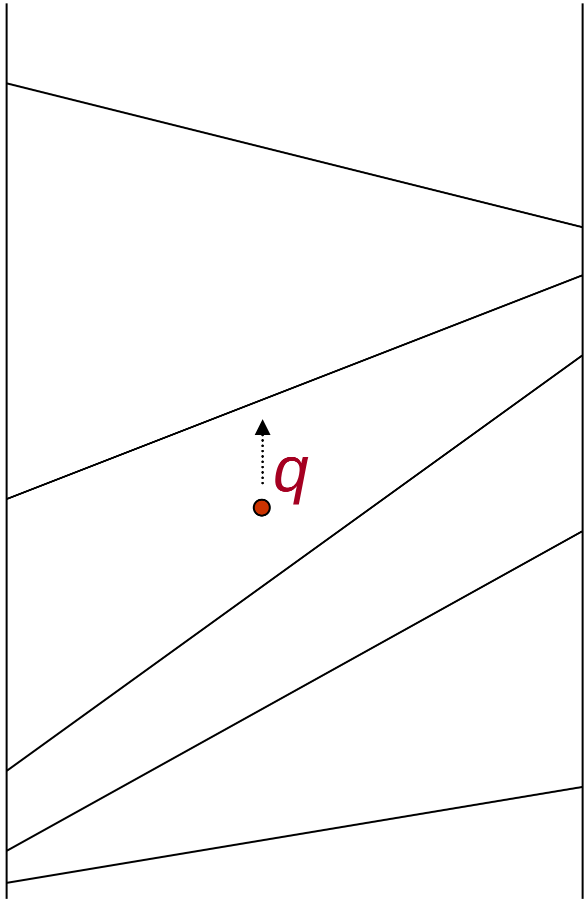
- Assumptions:
  - RAM with word of size w
  - input coordinates are integers in $\{0, ..., U\}$
  - $w \geq \lg n$ and $w \geq \lg U$
  - standard ops on w-bit integers (<, +, -, x, /, bitwise-&, <<, >>) take unit time

2D Point Location Problem

- Example: 1D predecessor search
  - $O(n)$ space, $O(\sqrt{\lg n / \lg\lg n})$ time [fusion tree+Beame&Fich]
    or $O(\lg\lg U)$ time [vEB tree]

- Our results in 2D:
  - $O(n)$ space, $\boxed{O(\lg n / \lg\lg n)}$ time
    or $\boxed{O(\sqrt{\lg U / \lg\lg U})}$ time
  - implies improved alg'ms for 2D Voronoi diagram, 3D convex hull, 2D line segment intersection, etc. by known CG techniques

Key Subproblem: Point Location in a Slab



- Idea: b-ary search

Key Recurrence

$$Q(n, U_L, U_R) \le Q_0(O(b), H, H) + \max\left\{Q\left(\frac{n}{b}, U_L, U_R\right), Q\left(n, \frac{U_L}{H}, U_R\right), Q\left(n, U_L, \frac{U_R}{H}\right)\right\} + O(1)$$

- Base case: $Q_0(b, H, H) = O(1)$ if $b \lg H \approx w$

  by packing multiple input line segments into a word

$$\Rightarrow \quad Q(n, U, U) = O(\log_b n + \log_H U)$$
$$= O(\lg n / \lg b + \lg U / \lg H) = O(\lg n / \lg b + b)$$

- Set $b = (\lg n)^{\varepsilon} \quad \Rightarrow \quad O(\lg n / \lg \lg n)$

Talk Outline

1. Point Location [C.&P., FOCS'06]

2. Offline Point Location [C.&P., STOC'07]

3. Offline Orthogonal Range Counting [C.&P., SODA'10]

4. Orthogonal Range Reporting [C.&Larsen&P., SoCG'11]

Offline 2D Point Location Problem

- Offline (or "batched") setting:

  what if all n query points are given in advance?

- The applications to 2D Voronoi diagram, 3D convex hull, 2D line segment intersection, etc. only need the offline case

Offline 2D Point Location Problem

- Example: 1D predecessor search

online queries $O(\sqrt{\lg n / \lg\lg n})$ or $O(\lg\lg U)$

offline queries $O(\sqrt{\lg\lg n})$ [Han&Thorup]

(since offline predecessor search $\Rightarrow$ sorting)

- Our result in 2D:

online queries $O(\lg n / \lg\lg n)$ or $O(\sqrt{\lg U / \lg\lg U})$

offline queries $\boxed{2^{O(\sqrt{\lg\lg n})}}$

New Key Recurrence

$$Q(n, U_L, U_R) \leq Q(O(b), H, H) + \max\left\{Q\left(\frac{n}{b}, U_L, U_R\right), Q\left(n, \frac{U_L}{H}, U_R\right), Q\left(n, U_L, \frac{U_R}{H}\right)\right\} + \tilde{O}(\ (\lg U_L + \lg U_R)/w\ )$$

by packing multiple query points into a word

$$\Rightarrow\ Q(n, U, U) = 2^{O(\sqrt{\lg \lg n})}$$

Remarks

- Related (vague) question: best word-RAM sorting alg'm that doesn't "rely on" hashing?

- For 2D Voronoi diagrams specifically: can do much better, in $O\big(\mathrm{sort}(n)\big) = O(n\sqrt{\lg\lg n})$ time [Buchin&Mulzer, FOCS'09]

- Other word-RAM results in CG:
  - dynamic 2D convex hull [Demaine&P., SoCG'07] , …

Talk Outline

1. Point Location [C.&P., FOCS'06]

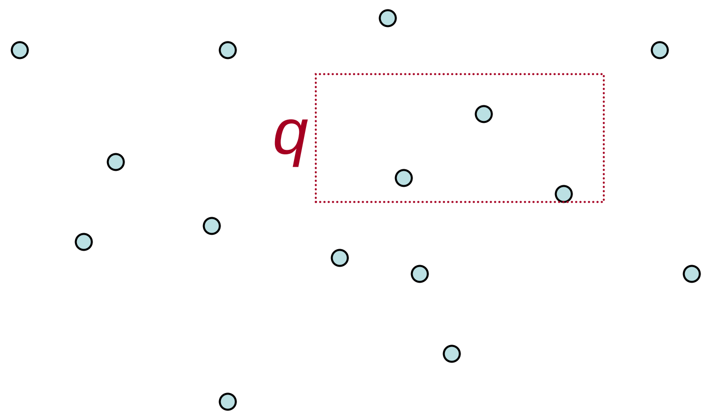
2. Offline Point Location [C.&P., STOC'07]

3. Offline Orthogonal Range Counting [C.&P., SODA'10]

4. Orthogonal Range Reporting [C.&Larsen&P., SoCG'11]

2D Orthogonal Range Searching Problem

- Preprocess n points in 2D s.t. given a query (axis-aligned) rectangle q,
  - report all points inside q
  - count # points inside q
  - decide if q is empty of points



- Textbook result: $O(n \lg n)$ space, $O(\lg n)$ time

(range tree)

2D Orthogonal Range Counting

- Best known result:

  $O(n)$ space, $O(\lg n / \lg\lg n)$ time (optimal)

- What if queries are offline?

- Offline 2D range counting $\Rightarrow$ dynamic 1D range counting (or "rank queries"): $O(\lg n / \lg\lg n)$ time [Dietz'88]

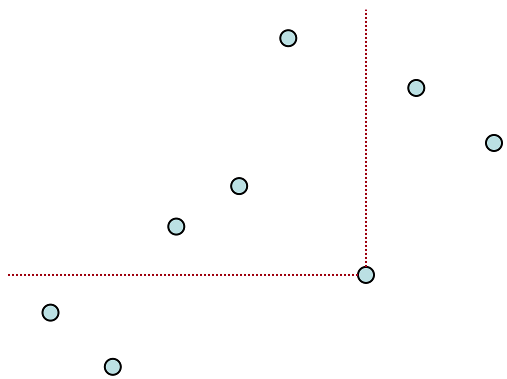
- Our result in 2D:

  offline queries $\boxed{O(\sqrt{\lg n})}$ time

Our Technique:
$O(n\sqrt{\lg n})$ Alg'm for Counting Inversions

- Problem: Given a permutation $a_1, \dots, a_n$ of $\{1, \dots, n\}$, count # (i,j) with $i < j$ & $a_i > a_j$

2 1 4 5 8 3 7 6 $\Rightarrow$



- Standard homework exercise: $O(n \lg n)$ time

- Inversion counting $\Rightarrow$ 2D range counting

by considering points (i, a_i)

Our Technique: $O(n\sqrt{\lg n})$ Alg'm for Counting Inversions

- Problem (slightly generalized): Given sequence $S = \langle a_1, \dots, a_n \rangle$, where each element in $\{1, \dots, U\}$ appears n/U times, count # (i, j) with $i < j$ & $a_i > a_j$

- Idea: B-way divide&conquer

Our Technique:
$O(n\sqrt{\lg n})$ Alg'm for Counting Inversions

- Problem (slightly generalized): Given sequence $S = \langle a_1, \ldots, a_n \rangle$, where each element in $\{1, \ldots, U\}$ appears n/U times, count # (i, j) with $i < j$ & $a_i > a_j$

1. For each $k = 1, \ldots, B$,

   recurse in the subsequence of S containing

   all elements in $\{(k-1)\left(\frac{U}{B}\right) + 1, \ldots, k\left(\frac{U}{B}\right)\}$

2. Recurse in $\left\langle \left\lceil a_1/(\frac{U}{B}) \right\rceil, \ldots, \left\lceil a_n/(\frac{U}{B}) \right\rceil \right\rangle$

$$\Rightarrow \quad T(n, U) \le B\, T\left(\frac{n}{B}, \frac{U}{B}\right) + T(n, B) + O(n)$$

Our Technique:
$O(n\sqrt{\lg n})$ Alg'm for Counting Inversions

$$\Rightarrow\ T(n,U) \le\ B\,T\left(\frac{n}{B},\frac{U}{B}\right) +\ T(n,B) +\ O(n)$$

- Example ($B = 2$): $\boxed{T(n,U) \le 2\,T\left(\frac{n}{2},\frac{U}{2}\right) + O(n)}$

  $\Rightarrow\ \ T(n,U) = O(n \lg U)$

- Hypothetical: if $T(n,B) = O(n)$ for a larger B,

  $\Rightarrow\ T(n,U) = O(n \log_B U)$...

Our Technique: $O(n\sqrt{\lg n})$ Alg'm for Counting Inversions

- Example ($B = 2$): $\boxed{T(n,U) \leq 2\,T\left(\frac{n}{2},\frac{U}{2}\right) + O(n\,(\lg U)/w)}$

  by word packing

  $$\Rightarrow \quad T(n,U) = O(n \lg U\,(\lg U)/w)$$

  $$\Rightarrow \quad T\left(n, 2^{\sqrt{w}}\right) = O(n)$$

- Now: set $B = 2^{\sqrt{w}}$

  $$\Rightarrow \; T(n,U) = O(n \log_B U) = O(n \lg U/\sqrt{w})$$

  $$= \; O(n\sqrt{\lg U})$$

Remarks

- Online dynamic 1D rank queries:

  $O(\lg n / \lg\lg n)$ query time,

  $O((\lg n)^{1/2+\varepsilon})$ update time

  [Dietz'88 had $O(\lg n / \lg\lg n)$ query & update]

- Offline orthogonal range counting in d dims:

  $O(n(\lg n)^{d-2+1/d})$ time

Talk Outline

1. Point Location [C.&P., FOCS'06]

2. Offline Point Location [C.&P., STOC'07]

3. Offline Orthogonal Range Counting [C.&P., SODA'10]

4. Orthogonal Range Reporting [C.&Larsen&P., SoCG'11]

2D Orthogonal Range Reporting

- Previous results – long story:
  - Lueker/Willard'78: $O(n \lg n)$ space, $O(\lg n + k)$ time (range trees)
  - Chazelle [FOCS'85]:

    $O(n(\lg n)^{\varepsilon})$ space, $O(\lg n + k)$ time

    $O(n \lg\lg n)$ space, $O(\lg n + k \lg\lg n)$ time

    $O(n)$ space, $O(\lg n + k(\lg n)^{\varepsilon})$ time
  - Overmars'88: $O(n \lg n)$ space, $O(\lg\lg U + k)$ time
  - Alstrup&Brodal&Rauhe [FOCS'00]:

    $O(n(\lg n)^{\varepsilon})$ space, $O(\lg\lg U + k)$ time

    $O(n \lg\lg n)$ space, $O((\lg\lg U)^2 + k \lg\lg U)$ time
  - Nekrich'07: $O(n)$ space, $O(\lg U / \lg\lg U + k(\lg U)^{\varepsilon})$ time
- Our result:

  $O(n \lg\lg n)$ space, $O\big((1 + k) \lg\lg U\big)$ time

  $O(n)$ space, $O\big((1 + k)(\lg U)^{\varepsilon}\big)$ time

2D Orthogonal Range Emptiness $(k = 0)$

- Previous results – short version:
  - Alstrup&Brodal&Rauhe [FOCS'00]:

    $O(n(\lg n)^{\varepsilon})$ space, $O(\lg\lg U)$ time

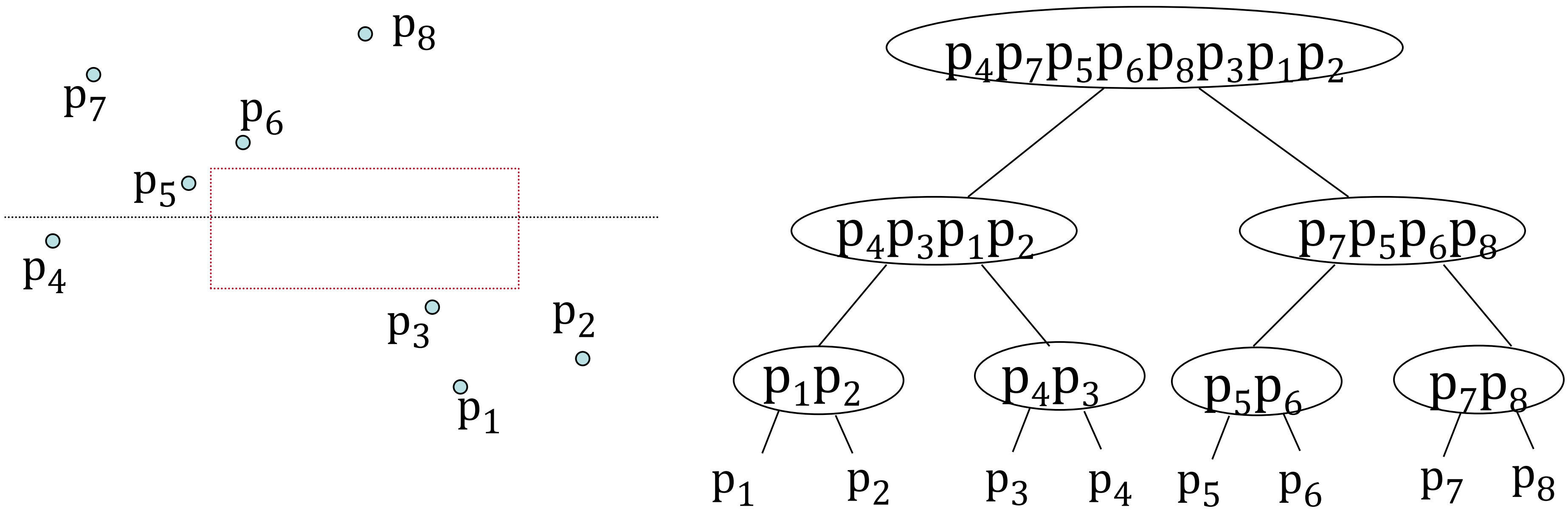
    $O(n\lg\lg n)$ space, $O\left((\lg\lg U)^2\right)$ time

- Our result:

  $\boxed{O(n\lg\lg n)}$ space, $\boxed{O(\lg\lg U)}$ time

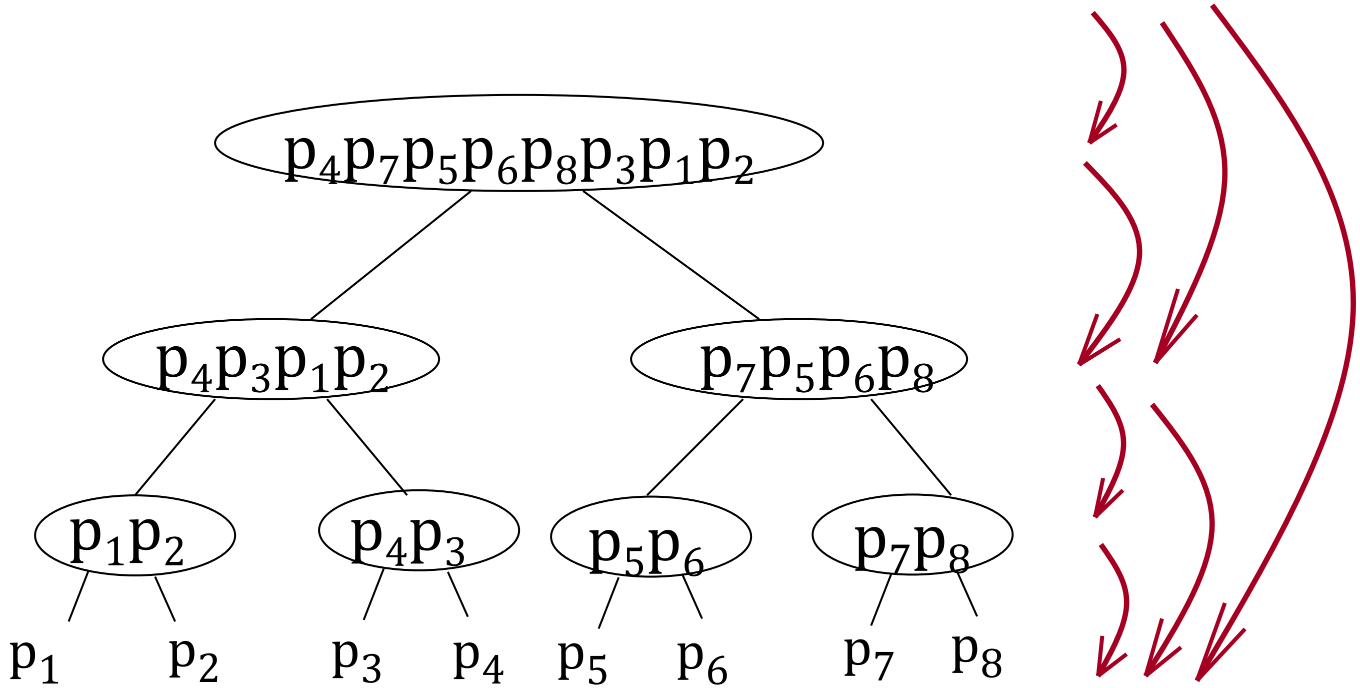
Mihai's Approach

- Idea: go back to standard range trees!



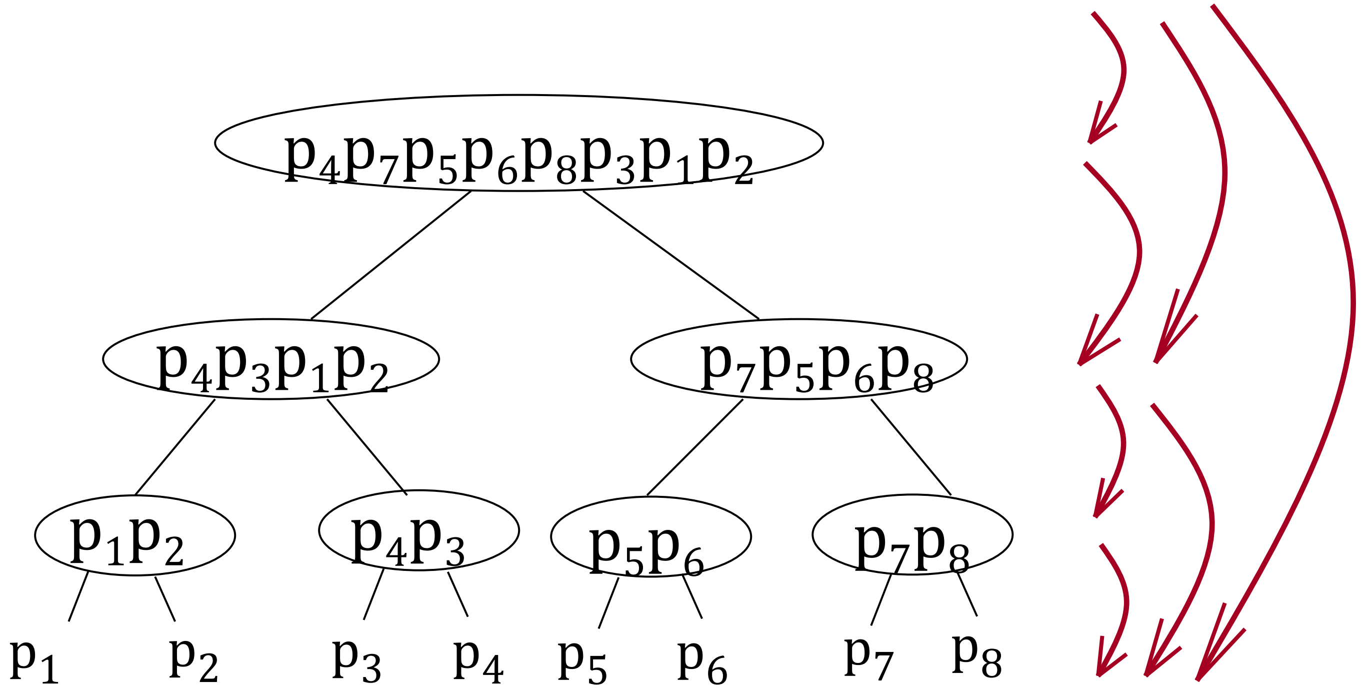
- Range emptiness query $\Rightarrow$ 1D range min/max queries + predecessor search at a node
- Save space by using succinct data structures…

Mihai's Approach



- Subproblem ("Ball Inheritance"): encode range tree s.t. given any node v & index i, can select i-th element of the list at v
- Solution: use known succinct "rank/select" data structures for strings...

Mihai's Approach



- Subproblem ("Ball Inheritance"): encode range tree s.t. given any node v & index i, can select i-th element of the list at v

$\Rightarrow$ $O(n \lg\lg n)$ space, $O(\lg\lg n)$ time

Mihai’s Approach

- One final idea: for predecessor search on a list, there is a data structure (based on vEB tree+fusion tree) which uses sublinear space!
- … provided there is an oracle that gives access to the i-th smallest element of the list for any given i
- Query time = $O(\lg\lg U)$ + time for $O(1)$ oracle calls

Remark

- Nekrich&Navarro [SWAT'12]: applies this approach to 1D range successor & 2D sorted orthogonal range reporting

- C.&Wilkinson [SODA'13]: applies this approach to 2D output-sensitive orthogonal range counting & approximate orthogonal range counting…

Open Problems

- Prove lower bounds!
  - (online) 2D point location in $\Omega(\lg n / \lg \lg n)$ or $\Omega(\sqrt{\lg U / \lg \lg U})$ time??
  - (online) dynamic 1D rank queries with $\mathrm{polylog}$ query time: $\Omega(\sqrt{\lg n})$ update time??
  - 2D orthogonal range emptiness (or just ball inheritance) with $\lg \lg n$ query time: $\Omega(n \lg \lg n)$ space??
  - 4D orthogonal range emptiness with $n \,\mathrm{polylog}\, n$ space: $\Omega(\lg n)$ or $O(\lg n / \lg \lg n)$ query time??