Orthogonal Range Queries: 2D

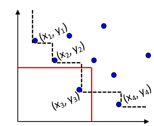
Mihai Pătraşcu
at&t

MADALGO Summer School 2010:
Monday Morning II

Existential, 2D Dominance

We only care about the staircase
$S = \{(x_1,y_1), \ldots, (x_k, y_k)\}$

We need to know the predecessor of
x_{query} among $\{ x_1, \ldots, x_k \}$

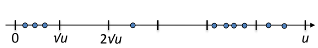


predecessor of α in S = max { β ∈S | β ≤ α }

NB: Predecessor search useful in all orthogonal range queries
general universe ↦ rank space

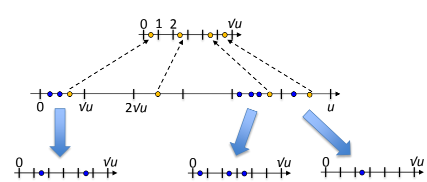
Predecessor Search

[van Emde Boas FOCS'75]
Preprocess $S \subset \{1, \ldots, u\}$ in space O(n)
Answer predecessor queries in O(lglg u)



Predecessor Search

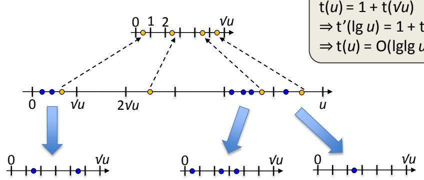
[van Emde Boas FOCS'75]
Preprocess $S \subset \{1, \ldots, u\}$ in space O(n)
Answer predecessor queries in O(lglg u)



Predecessor Search

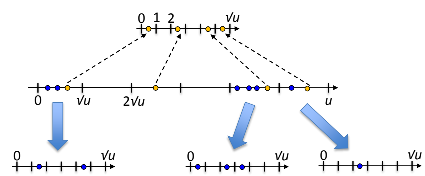
pred(x, S): **if** $\lfloor x/\sqrt{u} \rfloor \in$ hash table,
return pred(x mod $\sqrt{u}$, bottom structure)
else return pred($\lfloor x/\sqrt{u} \rfloor$, top structure)

$t(u) = 1 + t(\sqrt{u})$
$\Rightarrow t'(\lg u) = 1 + t'(½ \lg u)$
$\Rightarrow t(u) = O(\lg\lg u)$



Predecessor Search

Every point $\in$ O(lglg u) structures
$\Rightarrow$ space $S(n) = O(n \lg\lg u)$



Predecessor Search

Every point $\in$ O(lglg u) structures
$\Rightarrow$ space $S(n) = O(n \lg u)$

Bucketing:

Predecessor search:
space: (n/b)lglg u time: O(lglg u)

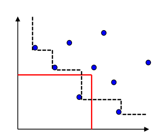
{ x_0, x_1, x_2, …, x_b, x_{b+1}, x_{b+2}, …, x_{2b}, x_{2b+1}, … }

if predecessor=x_0, finish off with binary search among {x_0, x_1,…, x_{b-1}}

Set b=lglg u

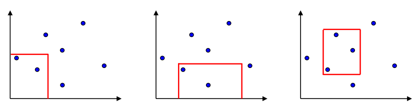
Recap

Existential, 2D dominance
Space: $O(n)$
Query time: O(lglg u)



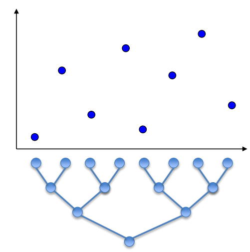
Adding Sides

Theorem: k-sided queries in space S(n), time t(n)
⇒ (k+1)-sided queries in space S(n lg n), time t(n)+O(lglg u)

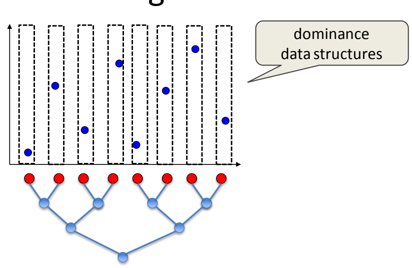


	2-sided	⇒	3-sided	⇒	4-sided
Space	O(n)		O(n lg n)		O(n lg^2n)
Time	O(lglg u)		O(lglg u)		O(lglg u)

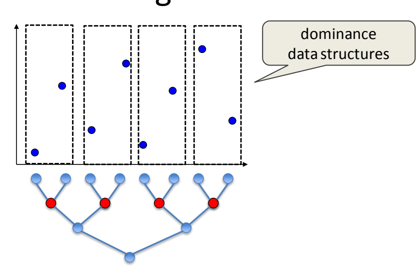
Adding Sides



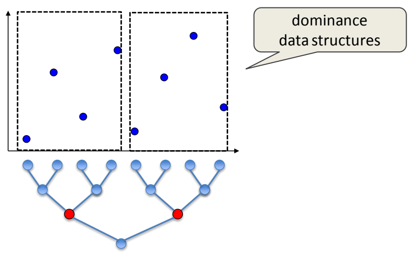
Adding Sides



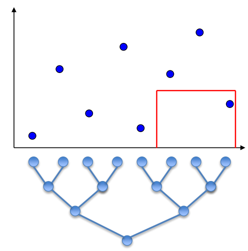
Adding Sides



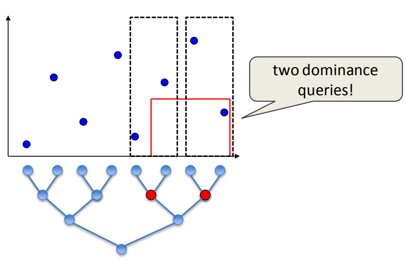
Adding Sides



Adding Sides



Adding Sides

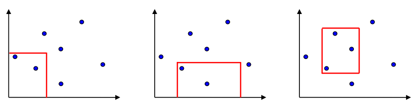


Adding Sides

1st reduce to rank space
- ⇒ tree has height O(lg n)
- ⇒ each point appears in O(lg n) structures
- ⇒ space ≤ S(n lg n)

time ≤ 2·t(n) + predecessor-search

Recap

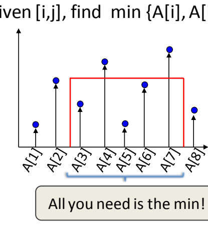


	2-sided ⇒	3-sided ⇒	4-sided
Time	O(lglg u)	O(lglg u)	O(lglg u)
Space	O(n)	~~O(n lg n)~~ O(n)	~~O(n lg^2 n)~~ O(n lg n)

3-sided queries

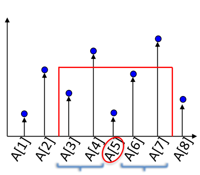
1D range minimum query in rank space

Range minimum query (RMQ):
Preprocess an array A[1..n] in O(n) space
Given [i,j], find min {A[i], A[i+1], …, A[j]} in O(1) time



3-sided queries

Bonus: 3-sided reporting in space O(n), time O(lglg u + k)



If min fits in range:
- report it
- recurse left
- recurse right

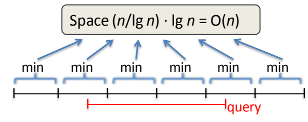
RMQ in O(n) space, O(1) time

Easy: RMQ in O(n lg n) space, O(1) time

Dominance RMQ trivial in O(n) space:
store B[1..n], B[k] = min { A[1], A[2], …, A[k] }

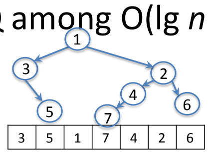
Reduce space: bucket O(lg n) elements

Space (n/lg n) · lg n = O(n)



RMQ among O(lg n) elements

Cartesian tree

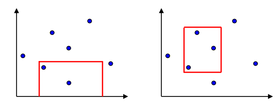


RMQ ⇔ lowest common ancestor in Cartesian tree

Storing Cartesian tree on k nodes
≈ storing $2k$ balanced parentheses ≤ $2k$ bits

So buckets of $\varepsilon \cdot \lg n$ elements can be described with $2\varepsilon \cdot \lg n$ bits
⇒ tabulate all answers in $O(n^{2\varepsilon} \lg^2 n)$ space!

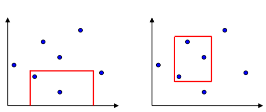
Recap



	rank space 3-sided	⇒	rank space 4-sided
Space:	$O(n)$		$O(n \lg n)$
Time:	$O(1)$		$O(\lg\lg n)$

Afternoon: $\Omega(\lg\lg n)$ needed, even in rank space!

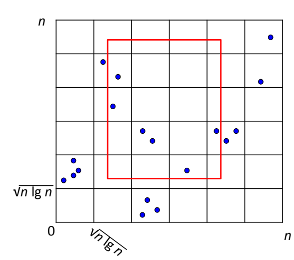
Recap



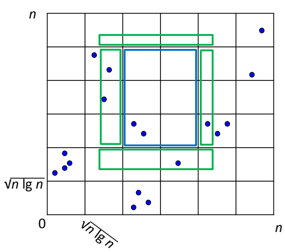
	rank space 3-sided	⇒	rank space 4-sided	[Alstrup, Brodal, Rauhe FOCS'00]	
Space:	$O(n)$		$O(n \lg n)$	$O(n \lg^{\varepsilon} n)$	$O(n \lg\lg n)$
Time:	$O(1)$		$O(\lg\lg n)$	$O(\lg\lg n)$	$O(\lg^2 \lg n)$

Afternoon: $\Omega(\lg\lg n)$ needed, even in rank space!

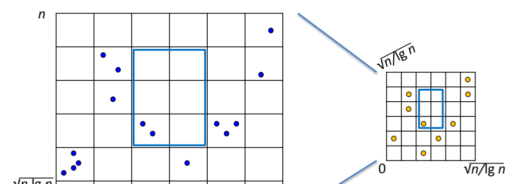
O(n lglg n) space, O(lg²lg n) query



O(n lglg n) space, O($\lg^2\lg n$) query

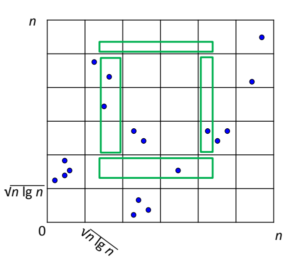


O(n lglg n) space, O($\lg^2\lg n$) query



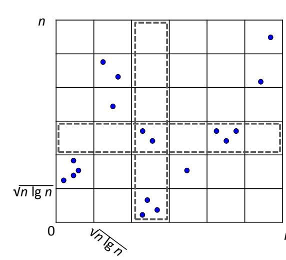
m = O(n/lg n) points
Use space O(m lg m)
time O(lglg m)

O(n lglg n) space, O($\lg^2\lg n$) query



3-sided queries
inside row or column!

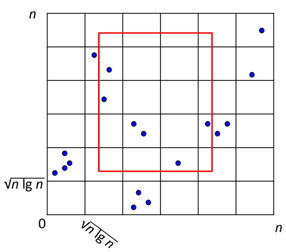
O(n lglg n) space, O($\lg^2\lg n$) query



Store 3-sided structure for each row/column:

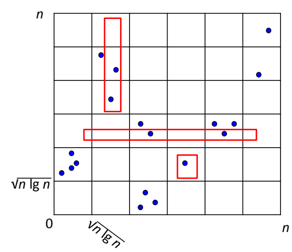
- linear space
- O(lglg n) query time

O(n lglg n) space, O($\lg^2 \lg n$) query



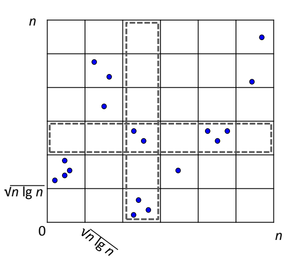
Good query:
- O(n) space
- O(lglg n) time

O(n lglg n) space, O($\lg^2 \lg n$) query



Bad queries

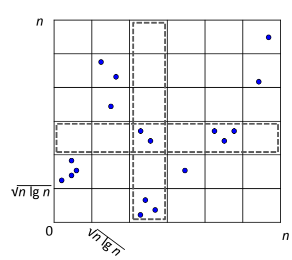
O(n lglg n) space, O($\lg^2 \lg n$) query



Recurse inside each row & column

Depth: O(lglg n)

O(n lglg n) space, O($\lg^2 \lg n$) query

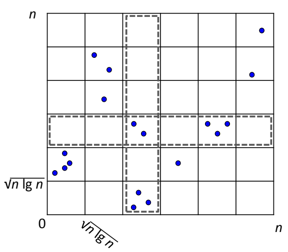


Recurse inside each row & column

Time: O(lglg n) to map global rank space ↦ row rank space

× O(lglg n) levels
= O($\lg^2 \lg n$) overall

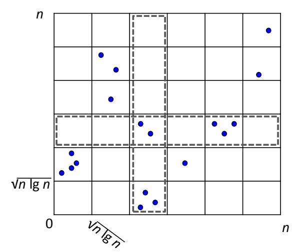
O(n lglg n) space, O(lg^{2}lg n) query



Space:
- level 1 → n points
- level 2 → 2n points
- …
- lglg n → n lg n points

O(n lg n) overall ?

O(n lglg n) space, O(lg^{2}lg n) query



Space:
- level 1 → n points
- level 2 → 2n points
- …
- lglg n → n lg n points

BUT: a "word" is
- level 1 → lg n bits
- level 2 → ½ lg n bits
- level 3 → ¼ lg n bits
- …

O(n lglg n) space, O(lg^{2}lg n) query

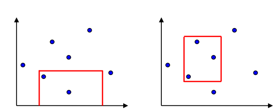
Space / level: O(n)

Total space: O(n lglg n)

Space:
- level 1 → n points
- level 2 → 2n points
- …
- lglg n → n lg n points

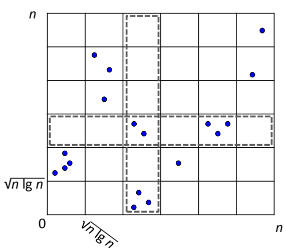
BUT: a "word" is
- level 1 → lg n bits
- level 2 → ½ lg n bits
- level 3 → ¼ lg n bits
- …

Recap



	rank space 3-sided	⇒	rank space 4-sided	Alstrup, Brodal, Rauhe FOCS'00]	
Space:	O(n)		O(n lg n)	O(n lg$^{\varepsilon}$$n$)	O(n lglg n)
Time:	O(1)		O(lglg n)	O(lglg n)	O(lg^{2}lg n)

$O(n \lg\lg n)$ space, ~~$O(\lg^2 \lg n)$~~ query
$O(\lg\lg n)$

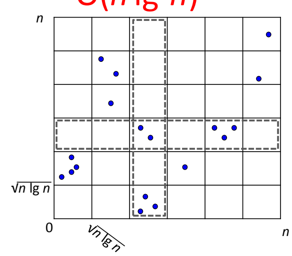


Recurse inside each row & column

Time: $O(\lg\lg n)$ to map global rank space ↦ row rank space

× $O(\lg\lg n)$ levels
= $O(\lg^2 \lg n)$ overall

~~$O(n \lg\lg n)$~~ space, ~~$O(\lg^2 \lg n)$~~ query
$O(n \lg^{\varepsilon} n)$ $O(\lg\lg n)$

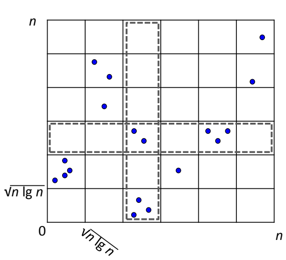


Recurse inside each row & column

Time: $O(\lg\lg n)$ to map global rank space ↦ row rank space

Once in $\varepsilon \cdot \lg\lg n$ levels
⇒ $(1/\varepsilon) \lg\lg n$ time

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



- level 1 → n points
- level 2 → $2n$ points
  …
- $\varepsilon \lg\lg n$ → $n \lg^{\varepsilon} n$ points

Coordinates are:
- level 1 → $\lg n$ bits
- level 2 → $\lg n$ bits
  …
- $\varepsilon \lg\lg n$ → $\lg^{1-\varepsilon} n$ bits

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

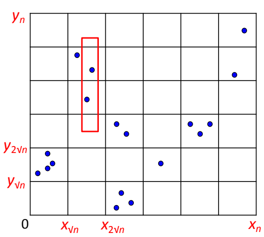
Space:
- level 1 → n
- level 2 → $2n$
…
- $(\varepsilon \lg\lg n)$-1 → $n \lg^{\varepsilon} n$
- $(\varepsilon \lg\lg n)$ → n

Total space: $O(n \lg^{\varepsilon} n)$

- level 1 → n points
- level 2 → $2n$ points
  …
- $\varepsilon \lg\lg n$ → $n \lg^{\varepsilon} n$ points

Coordinates are:
- level 1 → $\lg n$ bits
- level 2 → $\lg n$ bits
  …
- $\varepsilon \lg\lg n$ → $\lg^{1-\varepsilon} n$ bits

Life outside rank space



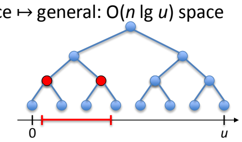
Which column to recurse in? Predecessor search?

Grid not uniformly spaced!

1D Range Reporting

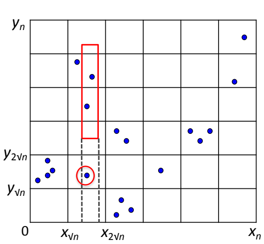
[Alstrup, Brodal, Rauhe STOC'01]
Range reporting in 1D with O(n) space, O(1) time

- dominance 1D reporting: store minimum ☺
- dominance ↦ general: O(n lg u) space



- O(n) space with hashing idea

Life outside rank space

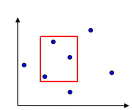


Store points sorted by x
+ pointer to column

1D range reporting
⇒ find point in x-range
⇒ know the column

Grid not uniformly spaced!

Recap



	Alstrup, Brodal, Rauhe FOCS'00]		We think we can do
Space:	O(n lg$^{\varepsilon}$$n$)	O(n lglg n)	O(n lglg n)
Time:	O(lglg n)	O(lg^{2}lg n)	O(lglg n)