Data structures for totally monotone matrices
Submatrix maximum queries in (inverse) Monge matrices

**Input:** $n \times n$ (inverse) Monge matrix $M$ represented implicitly (constant time oracle access)

**Output:** data structure answering maximum queries in any rectangular submatrix of $M$
Main Theorem:
A data structure with the following performance:
- preprocessing time: $O(n \log^2 n)$
- space: $O(n \log n)$
- query time: $O(\log^2 n)$

Extends to rectangular matrices and certain partial matrices

Application first...
For a set $P$ of $n$ points in the plane build a data structure such that:

Given a query $q$, efficiently finds the largest empty (of pts of $P$) rectangle containing $q$
Naïve Approach

Store all $N$ maximal empty rectangles in a two-dimensional segment tree.
(detour) A 1D Segment tree
A 2D Segment tree

The size of the primary tree is prop. to the number of different x-coordinates.

Since rectangles share x-coordinates there could be many rectangles mapped even to deep nodes.
The size of the primary tree is prop. to the number of different $x$-coordinates.

Since rectangles share $x$-coordinates there could be many rectangles mapped even to deep nodes.

In particular many rectangles can share their $x$-projections.
A 2D Segment tree

Each rectangle is mapped to $O(\log(n))$ primary nodes.

So if we have $N$ rectangles the total size of the secondary trees is $O(N\log(n))$.
Naïve approach

Store all $N$ maximal empty rectangles in a two-dimensional segment tree.

Space required is $O(N \log N)$, preprocessing time $O(N \log^2 N)$ and query time is $O(\log^2 N)$.

How many such rectangles could there be?
Naïve approach

Caveat: There may be $N = \Theta(n^2)$ maximal empty rectangles

There could be a Quadratic number of rectangles supported by two points in each of the 1st and 3rd quadrants (symmetrically, 2nd and 4th quadrants)
Different kinds of rectangles

- We will classify them, starting with rectangles that have an edge on the boundary
3 Edges on the boundary
2 edges on the boundary (type 1)
2 Edges on the boundary (type 2)
1 Edge on the boundary
Rectangle with edges on the boundary (summary)

- $O(n)$ rectangles, compute then in $O(n \log(n))$ time and put them in the segment tree in $O(n \log^2(n))$
Rectangle without edges on the boundary
2D Range tree
2D Range tree
An internal secondary node in the range tree

Corresponds to some rectangle with two dividers
A maximal empty rectangle

Contains the “origin” in some internal node $v$

Each node is responsible for the maximal empty rectangles containing its “origin”
Classify maximal empty rectangles containing the origin.

We are now focusing on one secondary node $v$ of the range tree, $n_v$ is the size of the subtree of $v$. 
3 points on one side of a divider

$O(n_v)$ of these, can compute then in $O(n_v \log n_v)$ time
1 point in each quadrant

Linearly many $O(n_v)$, can compute then in $O(n_v \log n_v)$ time
Summary so far

- Summing up over the nodes of the range tree we have $O(n \log^2(n))$ rectangles that we compute in $O(n \log^3(n))$ time.
- We insert them into the segment tree in $O(n \log^4(n))$ time.
- The segment tree takes $O(n \log^3 n)$ space and can answer a query in $O(\log^2 n)$ time.
We can compute the chains in $O(n_v \log(n_v))$ time, but there are too many rectangles to put in the segment tree.
The (inverse) Monge property

Define a matrix $M$ such that $M_{ik}$ is the area of maximal rectangle supported by $i^{th}$ pair in $3^{rd}$ quadrant and $k^{th}$ pair in $1^{st}$ quadrant.
Define a matrix $M$ such that $M_{ik}$ is area of maximal rectangle supported by $i^{th}$ pair in $3^{rd}$ quadrant and $k^{th}$ pair in $1^{st}$ quadrant.

The (inverse) Monge property
The (inverse) Monge property

For every $i < j$, $k < l$, $M_{ik} + M_{jl} \geq M_{il} + M_{jk}$
The matrix is not fully defined
The relation between the defined parts of consecutive rows
The relation between the defined parts of consecutive rows
The relation between the defined parts of consecutive rows
The matrix is "staircase"
The matrix is “staircase”
Query

A query point $q$ defines ranges of relevant pairs in the 1\textsuperscript{st} and 3\textsuperscript{rd} quadrants.

Use the Monge submtarix maximum data structure to find the maximum-area rectangle containing $q$ in the relevant range.
Analysis

It takes $O(n_v a(n_v) \log(n_v))$ space $O(n_v a(n_v) \log^2 n_v)$ preprocessing time per secondary node $v$

Summing over all secondary nodes of the range tree we get that the total space is $O(n a(n) \log^3(n))$ and the preprocessing time is $O(n a(n) \log^4(n))$

Query time is $O(\log^2 n_v)$ at a secondary node $v$ and $O(\log^4 n)$ overall
That's it as far as the empty rectangle application is concerned
Back to our data structure

**Input:** $n \times n$ Monge matrix $M$
represented implicitly
(constant time oracle access)

**Output:** Data structure answering
maximum queries in any rectangular
submatrix of $M$
Submatrix maximum queries in Monge matrices

Main Theorem:
A data structure with the following performance:
preprocessing time \( O(n \log^2 n) \)
space \( O(n \log n) \)
query time \( O(\log^2 n) \)
Think of row \( i \) of \( M \) as a discrete function mapping column \( j \) to \( M_{ij} \). Extend the domain to the interval \([1,n]\) by linear interpolation.
Totally monotone matrices and pseudo-lines

These are pseudo-lines:
Every pair of curves intersect in at most one point
Upper envelopes of pseudo-lines

- Column maxima are on the upper envelope of the pseudo-lines
- For m rows there are $O(m)$ breakpoints
- Represent an upper envelope by a search tree over the breakpoints
Column maxima in a range of rows
Column maxima in a range of rows

- Balanced search tree over the rows
- Each node stores the upper envelope of its rows
- Takes $O(n \log n)$ time in a bottom-up computation
A query is "covered" by $O(\log n)$ canonical nodes.
- Search the envelope of each canonical node for the interval containing the column.
- Return max of entries of the appropriate rows.
A query can be answered in $O(\log^2 n)$ time by querying each of the canonical nodes in $O(\log n)$ time.

Using fractional cascading reduces query time to $O(\log n)$
Maximum in a general submatrix
Maximum in a general submatrix

From each envelope we need the maximum in a sequence of complete intervals and a prefix+suffix of two intervals
Store additional information on the upper envelope

- Compute the maxima in each interval of the envelope
- This is a maxima in a subinterval of a row
- Can compute using a structure as before over the columns
- Store subtree maxima in the search tree of the breakpoints
• Find the maxima in the complete intervals of the envelope contained in the query
Submatrix query

- Find the maxima in the suffix and the prefix by queries to the structure for maxima in subintervals of rows
Submatrix queries: summary

preprocessing time \( O(n \log n) \) (the total number of different intervals on all envelopes is \( O(n) \))
space \( O(n \log n) \)
query time \( O(\log^2 n) \)
Submatrix queries: Extensions

- extension to certain Monge partial matrices using pseudo-segments instead of pseudo-lines
- preprocessing time: $O(n\alpha(n)\log^2 n)$
  - space: $O(n\alpha(n)\log n)$
  - query time: $O(\log^2 n)$