# Rectilinear Path Problems in Restricted Memory Setup

Binay K. Bhattacharya
Minati De
Anil Maheswari
Subhas C. Nandy
Sasanka Roy

February 8, 2015

# Organization

# Introduction

-

## Rectilinear Path Problem

The Input: A set $\mathcal{R}$ of axis-parallel rectangular obstacles, and a pair of points $p$ and $q$.
Objective: A rectilinear path from $p$ to $q$ of desired type.

## Memory Models considered

Inplace algorithm: Here the elements of the array $\mathcal{R}$ can be swapped during the execution. But, at any time, all the obstacles are available in $\mathcal{R}$
Read-only algorithm: Here the elements in $\mathcal{R}$ can only be read (may be multiple times); the writing in the array is not permitted.

# Introduction

## Problems Considered

### Read-only algorithms

- Checking the existance of a $xy$-monotone path from $p$ to $q$, and if exists then report it.
  Complexity results: $O(\frac{n^2}{s} + n \log s)$ time using $O(s)$ space.
- Reporting an $x$-monotone path from $p$ to $q$.
  Complexity results: $O(\frac{n^2}{s} + n \log s)$ time using $O(s)$ space.

# Introduction

Problems Considered

## Inplace algorithms

Preprocess the input rectangles, and
Given two points $p$ and $q$ as query, report a path between $p$ and $q$.

Arbitrary rectangles: Preprocessing time: $O(n \log n)$
Query answering time: $O(n^{3/4} + \chi)$

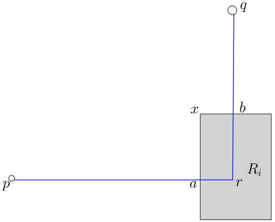
Unit squares: Preprocessing time: $O(n \log n)$
Query answering time: $O(\log n)$

Extra space requirement: $O(1)$ for both the problems.

# Read-only Algorithms

# Reporting a path - a simple algorithm

- Join $p$ and $q$ by an $L$-path.

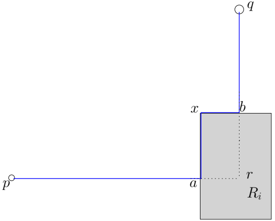


# Reporting a path - a simple algorithm

- Join $p$ and $q$ by an $L$-path.
- The corner $r$ is in a rectangle $R \in \mathcal{R}$.

$a$ - the point of intersection of $\overline{pr}$ and $R$

$b$ - the point of intersection of $\overline{qr}$ and $R$

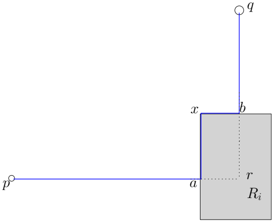


# Reporting a path - a simple algorithm

- Join $p$ and $q$ by an $L$-path.
- The corner $r$ is in a rectangle $R \in \mathcal{R}$.

$a$ - the point of intersection of $\overline{pr}$ and $R$

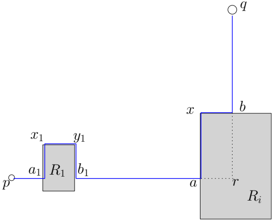
$b$ - the point of intersection of $\overline{qr}$ and $R$




- Report the path $p \rightsquigarrow a \oplus L\_path(a \rightarrow x \rightarrow b) \oplus b \rightsquigarrow q$

# Computation of the path $p \rightsquigarrow a$

1. Set $s = p$
2. Identify a rectangle $R \in \mathcal{R}$ whose $y$-span contains $y(s)$, and its left boundary is closest to $s$.
3. Report the path $s \to a_1 \to x_1 \to y_1 \to b_1$
4. Set $s = b_1$
5. Repeat steps 2 and 3 until $a$ is reached.



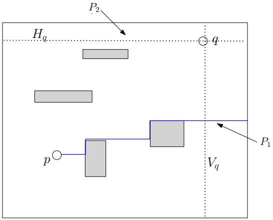
# Complexity Results

**Time Complexity - $O(\frac{n^2}{s} + n \log s)$ using $O(s)$ space**

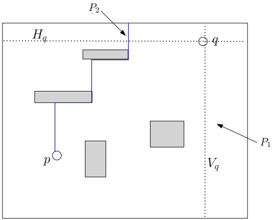
# $xy$-monotone path

Assumption - $p$ is to the bottom-left of $q$.

**$x$-preferred $xy$-monotone path**

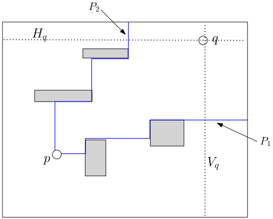



**$y$-preferred $xy$-monotone path**



# $xy$-monotone path

If $q$ lies in the region bounded by $x$-preferred $xy$-path and $y$-preferred $xy$-path from $p$, then there exists $xy$-monotone path from $p$ to $q$.




Time complexity for testing: $O(\frac{n^2}{s} + n \log s)$ using $O(s)$ space

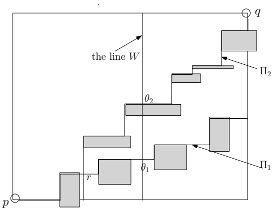
# Reporting the path

$H_a$ - Horizontal line through a point $a$.
$V_a$ - Vertical line through a point $a$.

$\Pi_1$: $x$-preferred $xy$-path from $p$ up to $V_q$.
$\Pi_2$: $(-y)$-preferred $(-x)(-y)$-path from $q$ up to $H_p$.

Let $\Pi_1$ and $\Pi_2$ intersect at a point $r$.

Now, report
the $x$-preferred $xy$-path from $p$ to $r$, and
the $(-y)$-preferred $(-x)(-y)$-path from $q$ to $r$.



(a)

# Computing the point $r$

**Objective:** Compute a vertical line $W$ on which $r$ lies.

## Useful rectangles

A set of rectangles lying within a pair of vertical lines $x = \tau_1$ and $x = \tau_2$ on the two sides of $W$.
Initially, $\tau_1 = x(p)$ and $\tau_2 = x(q)$.

Apply binary search to compute $W$.

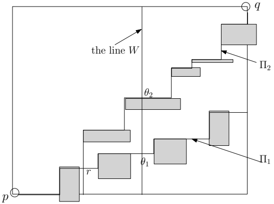
- Compute the median $\mu$ of the left boundaries of the *useful rectangles* to define $W : x = \mu$.
- Compute the $x$-preferred $xy$-path from $p$ that intersects $W$ at a point $\theta_1$.
- Compute the $(-y)$-preferred $(-x)(-y)$-path from $q$ that intersects $W$ at a point $\theta_2$.

# Computing the point $r$

$y(\theta_1) < y(\theta_2)$: $r$ is to the left of $W$.

.

.



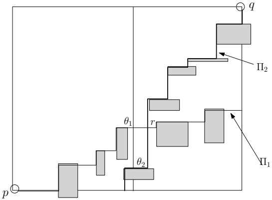

(a)

We set $\tau_2 = \mu$.

$y(\theta_1) > y(\theta_2)$: $r$ is to the right of $W$.

.

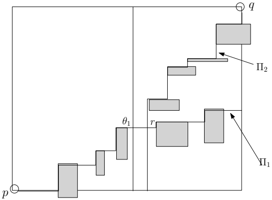
.




(b)

We set $\tau_1 = \mu$.

$\Pi_2$ intersects $H_p$ to the right of $W$. Here also, $r$ is to the right of $W$.




(b)

We set $\tau_1 = \mu$.

# Time Complexity

$M_s$: - Time required to compute median with $O(s)$ extra space.

- Number of calls of the median finding – $O(\log n)$.
- In each level of recursion we can prune the set of useful rectangles.

Thus, the total time to compute $r$ is $O(\frac{n^2}{s} + n \log s + M_s \log n)$ using $O(s)$ space.

## Drawback

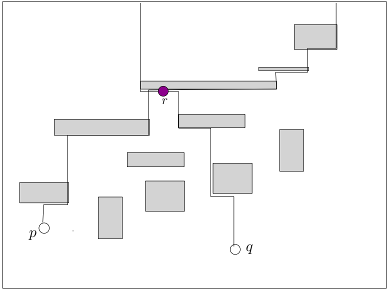
Printing of the path is not at a stretch from $p$ to $q$.
It is a concatenation of paths $p \rightsquigarrow r$ and $q \rightsquigarrow r$.

# $x$-monotone path

- Compute $x$-preferred $xy$-path from $p$. Let it hit the top boundary of the bounding box at point $\alpha$
- Compute $(-x)$-preferred $(-x)y$-path from $q$. Let it hit the top boundary of the bounding box at point $\beta$

$x(\alpha) < x(\beta)$: These two paths do not intersect.
Report $p \rightsquigarrow \alpha \rightarrow \beta \rightsquigarrow q$.

$x(\alpha) > x(\beta)$: Compute the point of intersection $r$ of these two paths.
Report the path
$p \rightsquigarrow r \rightsquigarrow q$.




Time Complexity: $O(\frac{n^2}{s} + n \log s)$ using $O(s)$ space.

# Inplace Algorithms

# Path query among arbitrary obstacles

**Problem Statement**

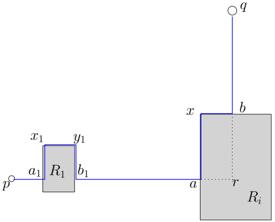
**Input:** A set of axis-parallel rectangles $\mathcal{R}$ in an array. $R_i = [(a_i, b_i), (c_i, d_i)]$, $i = 1, 2, \ldots, n$.

**Objective:** To preprocess these rectangles in $\mathcal{R}$ such that given any pair of query points $p$ and $q$, a manhattan path can be reported easily.

**Useful Tool:** An inplace k-d tree data structure of Bronnimann et al. maintained with the tuples $(a_i, c_i, b_i, d_i)$, $i = 1, 2, \ldots, n$.

# Reporting a path

- Join $p$ and $q$ by an $L$-path.
- The corner $r$ is in a rectangle $R \in \mathcal{R}$.
  $a$ - the point of intersection of $\overline{pr}$ and $R$
  $b$ - the point of intersection of $\overline{qr}$ and $R$
  - Report the path
    $p \rightsquigarrow a \oplus L\_path(a \to x \to b) \oplus b \rightsquigarrow q$
- The corner $r$ is not inside any rectangle.
  - Report the path $p \rightsquigarrow r \oplus r \rightsquigarrow q$



# Reporting a path from $p$ to $a$

Using 4-d tree, the rectangles intersected by the line $[p, a]$ can be reported in order.

## Time Complexity

Preprocessing: $O(n \log n)$

Reporting the path: $O(n^{3/4} + \chi)$, where $\chi$ is the number of links in the path.

Extra space requirement: $O(1)$

# Path Query Among Non-Overlapping Unit Squares

## The Problem

- Preprocess the given obstacles $\mathcal{R}$ in an inplace manner
- to report a path between a given pair of query points using $O(1)$ extra space.

## Targeted Results

- There exists a path of length $O(\log n)$.
- Preprocessing time $O(n \log n)$
- Query time $O(\log n)$

# Preprocessing

## Definition:

For a unit square $R \in \mathcal{R}$

- its center-point is the point of intersection of its two diagonals,
- its left-point is its top-left corner, and
- its right-point is its top-right corner.

## Data Structure

An inplace priority search tree $\mathcal{T}$ with the center-points of the members in $\mathcal{R}$.

# Query

**Path from left-point of root $R_r$ to one of the query points $p$**

Easy case: $p$ is above $R_r$. Join the left-point of $R_r$ and $p$ by a $L$-path.

Otherwise: Search $\mathcal{T}$ to report the manhattan path from the left-point of $R_r$ and $p$.

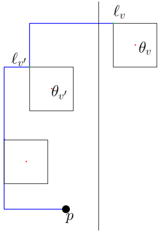
# Recursive Query Algorithm

- Let a path from $R_r$ to $R_v$ is already computed, and
- $p$ is in the left partition of node $v$.

Also let $v'$ be the left child of node $v$.

**$p$ is below the top boundary of $R_{v'}$:**

Here join the left-point of $R_v$ and the left-point of $R_{v'}$, and perform the recursive call with $R_{v'}$.

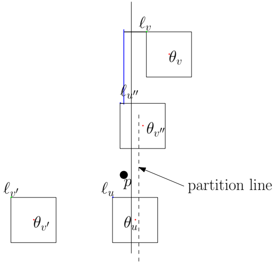


# Recursive Query Algorithm

**$p$ is above the top boundary of $R_{v'}$:**

Here, in the path between the left-point of $R_v$ and the point $p$,

- no rectangle of the left partition will intersect,
- but rectangles in the right partition that may intersect.



# Recursive Query Algorithm

Let $v''$ be the right child of $v$.

**Simple case**

$p$ is above $R_{v''}$:
Join the left-point of $R_v$ and the point $p$ by an $L$-path, and
The process stops.

# Recursive Query Algorithm

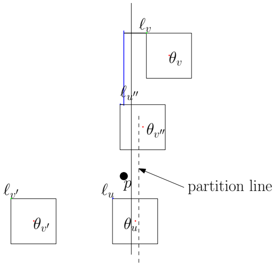
Let $v''$ be the right child of $v$.

## Simple case

$p$ is above $R_{v''}$:
Join the left-point of $R_v$ and the point $p$ by an $L$-path, and
The process stops.

## The other case:

$p$ is below $R_{v''}$:
Here, $R_{v''}$ may or may not intersect the $L$-path from $R_v$ to the point $p$.

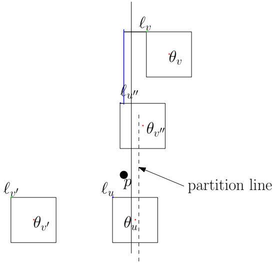


# Recursive Query Algorithm

**The other case:**

$p$ is below $R_{v''}$:

Here, $R_{v''}$ may or may not intersect the $L$-path from $R_v$ to the point $p$.




Former case: Join left-point of $R_v$ and the left-point of $R_{v''}$

In both cases: Recurse with $p$ and $R_{v''}$

# Complexity Results

## Result

The length of the path from $R_r$ to $p$ is $O(\log n)$.

- Traverse the tree to find a node $v$ such that $p$ and $q$ are in the different side of the discriminant line of that node.
- Choose node $v$ as the root $r$,
- Process QUERY($p$, $R_r$)
- Process QUERY($q$, $R_r$)

## Complexity Results

Preprocessing: $O(n \log n)$ for constructing $\mathcal{T}$.

Query: $O(\log n)$

Space: $O(1)$