Tight Bounds for
Dynamic Convex Hull Queries
(Again)

Erik D. Demaine    Mihai Pătraşcu

(SoCG ’07)

Presented By: Joseph A. Simons
Convex Hull
Convex Hull

Convex Hull

Convex Hull

- Minimal Convex Set Containing $S$
- Intersection of Convex Sets Containing $S$
- Intersection of Half-Spaces Containing $S$
<table>
<thead>
<tr>
<th>Query</th>
<th>Update</th>
<th>[Overmars and van Leeuwen ’81]</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\lg n$</td>
<td>$\lg^2 n$</td>
<td></td>
</tr>
<tr>
<td>Query</td>
<td>Update</td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>----------</td>
<td></td>
</tr>
<tr>
<td>$\lg n$</td>
<td>$\lg^2 n$</td>
<td>[Overmars and van Leeuwen '81]</td>
</tr>
<tr>
<td>$\lg n$</td>
<td>$\lg^{1+\epsilon} n$</td>
<td>[Chan '99]</td>
</tr>
<tr>
<td>Query</td>
<td>Update</td>
<td>Reference</td>
</tr>
<tr>
<td>--------</td>
<td>----------------------</td>
<td>------------------------------------</td>
</tr>
<tr>
<td>$\lg n$</td>
<td>$\lg^2 n$</td>
<td>[Overmars and van Leeuwen '81]</td>
</tr>
<tr>
<td>$\lg n$</td>
<td>$\lg^{1+\epsilon} n$</td>
<td>[Chan '99]</td>
</tr>
<tr>
<td>$\lg n$</td>
<td>$\lg n \lg \lg n$</td>
<td>[Brodal and Jacob '00]</td>
</tr>
<tr>
<td>Query</td>
<td>Update</td>
<td>Reference</td>
</tr>
<tr>
<td>----------</td>
<td>------------</td>
<td>--------------------------------</td>
</tr>
<tr>
<td>$\lg n$</td>
<td>$\lg^2 n$</td>
<td>[Overmars and van Leeuwen '81]</td>
</tr>
<tr>
<td>$\lg n$</td>
<td>$\lg^{1+\epsilon} n$</td>
<td>[Chan '99]</td>
</tr>
<tr>
<td>$\lg n$</td>
<td>$\lg n \lg \lg n$</td>
<td>[Brodal and Jacob '00]</td>
</tr>
<tr>
<td>$\lg n$</td>
<td>$\lg n$</td>
<td>[Brodal and Jacob FoCS 02]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[Jacob PhD thesis]</td>
</tr>
<tr>
<td>Query</td>
<td>Update</td>
<td>Reference</td>
</tr>
<tr>
<td>---------</td>
<td>------------------</td>
<td>----------------------------------------</td>
</tr>
<tr>
<td>$\lg n$</td>
<td>$\lg^2 n$</td>
<td>[Overmars and van Leeuwen ’81]</td>
</tr>
<tr>
<td>$\lg n$</td>
<td>$\lg^{1+\epsilon} n$</td>
<td>[Chan ’99]</td>
</tr>
<tr>
<td>$\lg n$</td>
<td>$\lg n \lg \lg n$</td>
<td>[Brodal and Jacob ’00]</td>
</tr>
<tr>
<td>$\lg n$</td>
<td>$\lg n$</td>
<td>[Brodal and Jacob FoCS 02]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[Jacob PhD thesis]</td>
</tr>
</tbody>
</table>

1 Amortized Time

2 Only works for decomposable queries
<table>
<thead>
<tr>
<th>Query</th>
<th>Update</th>
<th>Ref</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\lg n$</td>
<td>$\lg^2 n$</td>
<td>[Overmars and van Leeuwen ’81]</td>
</tr>
<tr>
<td>$\lg n$</td>
<td>$\lg^{1+\epsilon} n$</td>
<td>[Chan ’99]</td>
</tr>
<tr>
<td>$\lg n$</td>
<td>$\lg n \lg \lg n$</td>
<td>[Brodal and Jacob ’00]</td>
</tr>
<tr>
<td>$\lg n$</td>
<td>$\lg n$</td>
<td>[Brodal and Jacob FoCS 02] [Jacob PhD thesis]</td>
</tr>
</tbody>
</table>

1 Amortized Time
2 Only works for decomposable queries

optimal on real-RAM (infinite precision)
\[
\begin{array}{ccc}
\log n & \log \log n & \log^2 n \\
\text{[Overmars and van Leeuwen '81]} & & \\
\log n & \log^{1+\epsilon} n & \text{[Chan '99]} \\
\log n & \log n \log \log n & \text{[Brodal and Jacob '00]} \\
\log n & \log n & \text{[Brodal and Jacob FoCS 02]} \\
& & \text{[Jacob PhD thesis]}
\end{array}
\]

1 Amortized Time

2 Only works for decomposable queries
<table>
<thead>
<tr>
<th>Time Complexity</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \log n / \log \log n )</td>
<td>Overmars and van Leeuwen '81</td>
</tr>
<tr>
<td>( \log^2 n )</td>
<td>Cuth '99</td>
</tr>
<tr>
<td>( \log n \log \log n )</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Time Complexity</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \log n )</td>
<td>Overmars and van Leeuwen '81</td>
</tr>
<tr>
<td>( \log^2 n )</td>
<td>Chan '99</td>
</tr>
<tr>
<td>( \log n \log \log n )</td>
<td>Brodal and Jacob '00</td>
</tr>
</tbody>
</table>

1 Amortized Time
2 Only works for decomposable queries
Optimal on word-RAM
(finite precision)

$\log n / \log \log n \quad \log^2 n \quad \log n \log \log \log n$ [this paper]

\[
\begin{array}{ccc}
\lg n & \lg^2 n & \text{[Overmars and van Leeuwen '81]} \\
\lg n & \lg^{1+\epsilon} n & \text{[Chan '99]} \\
\lg n & \lg n \lg \lg n & \text{[Brodal and Jacob '00]} \\
\lg n & \lg n & \text{[Brodal and Jacob FoCS 02]} \\
& & \text{[Jacob PhD thesis]}
\end{array}
\]

$^1$ Amortized Time
$^2$ Only works for decomposable queries
Tangent Queries
Tangent Queries
Tangent Queries
Query Taxonomy

tangent

(linear programming)

(line decision)

(line stabbing)

(vertical stabbing)

(containment)

(hull membership)

gift wrapping
Query Taxonomy

tangent

linear programming

line decision

gift wrapping

line stabbing

vertical stabbing

containment

hull membership

hardest

easiest
Query Taxonomy

2d
- tangent
- line stabbing

1d
- linear programming
- line decision
- vertical stabbing
- containment

0d
- hull membership
- gift wrapping
Query Taxonomy

2d
- tangent
- line stabbing

1d
- linear programming
- line decision
- vertical stabbing
- containment

0d
- hull membership
- gift wrapping $\Theta(1)$

$\Theta(\lg n / \lg \lg n)$
Computational Model

Geometric View:

   Infinite Precision
Computational Model

Geometric View: Infinite Precision

Computational View: Represented by (finitely many) bits
Computational Model

Geometric View: Infinite Precision

Computational View: Represented by (finitely many) bits

fixed universe $[u]$
Computational Model

Geometric View: Infinite Precision

Computational View: Represented by (finitely many) bits

Faster Algorithms (e.g. Radix Sort) (van Emde Boas) (planar point location)
Computational Model

Geometric View: Infinite Precision

Computational View: Represented by (finitely many) bits

transdichotomous RAM

number of bits $w = \theta(\lg n)$
Divide and Conquer Construction
Divide and Conquer Construction

Divide $P$ in half by $x$-coordinate
Divide and Conquer Construction

Divide $P$ in half by $x$-coordinate

Build Hulls
Divide and Conquer Construction

Divide $P$ in half by $x$-coordinate

Build Hulls

Bridge
Divide and Conquer Construction

Divide $P$ in half by $x$-coordinate

Build Hulls

Bridge

"Throw Away" Obscured Parts
Divide and Conquer Construction

Combined Hull \( UH(P) \)
Divide and Conquer Construction

Gives Tree:

Leaves store individual points
Nodes keep UH of subtree
Tree:

Leaves store individual points

[Overmars and van Leeuwen '81]
Tree:

Leaves store individual points

Nodes keep UH of subtree and dividing line

[Overmars and van Leeuwen ’81]
Tree:

Leaves store individual points

Nodes keep UH of subtree and dividing line

UH stored as concatenable queue (e.g. B-tree)

[Overmars and van Leeuwen '81]
Tree:

Leaves store individual points
Nodes keep UH of subtree and dividing line
UH stored as concatenable queue (e.g. B-tree)
May actually only store "thrown away"
  hull not stored by parent
  and reconstruct by merging and splitting on the fly

[Overmars and van Leeuwen '81]
Dynamic Convex Hull

Query: Search UH stored at root \( O(\lg n) \)

[Overmars and van Leeuwen '81]
Dynamic Convex Hull

Query: Search UH stored at root \( O(\lg n) \)

Update: Update each node on leaf to root path
Dynamic Convex Hull

[Overmars and van Leeuwen ’81]

Query: Search UH stored at root $\mathcal{O}(\log n)$

Update: Update each node on leaf to root path

$\mathcal{O}(\log n)$ nodes $\times \mathcal{O}(\log n)$ time per node

$\mathcal{O}(\log^2 n)$ time.
How to improve query time to $O(\lg n / \lg \lg n)$?
How to improve query time to $O(lg n / lg lg n)$?

We stress that the name of the game here is not developing fancy “bit tricks” to exploit word level parallelism, but rather studying how geometric information such as points and lines can be decomposed in algorithmically useful ways.
How to improve query time to $O(\lg n / \lg \lg n)$?

First idea: in UH tree,

increase branching factor to $\lg^\epsilon n$
How to improve query time to $O(\lg n / \lg \lg n)$?

First idea: in UH tree,

increase branching factor to $\lg^\epsilon n$
How to improve query time to $O(\lg n / \lg \lg n)$?

First idea: in UH tree, increase branching factor to $\lg^\epsilon n$

Tree Height: $O(\lg n / \lg \lg n)$

Decision at each node requires point location
How to improve query time to $O(\lg n / \lg \lg n)$?

First idea: in UH tree,

increase branching factor to $\lg^\epsilon n$

But only leaves $O(1)$ time per node for planar point location
How to improve query time to $O(\lg n / \lg \lg n)$?

First idea: in UH tree,

increase branching factor to $\lg^c n$

Instead:

Allowed to take superconstant time

But $O(1)$ averaged over all $O(\log_B n)$ nodes
How to improve query time to $O(\lg n / \lg \lg n)$?

First idea: in UH tree,

increase branching factor to $\lg^\epsilon n$

Instead:

Allowed to take superconstant time

But $O(1)$ averaged over all $O(\log_B n)$ nodes

Time taken is proportional to info learned
Fact:
Consider a vertical slab $\{x_L, \ldots x_R\} \times [u]$
Fact:
Consider a vertical slab \( \{x_L, \ldots x_R\} \times [u] \)
Fact:
Consider a vertical slab \( \{x_L, \ldots x_R\} \times [u] \)

\( B \) Segments with coordinates \((x_L, \ell_i), (x_R, r_i)\) \( B \leq w \)

\( \ell_i, r_i \) rational
Fact:
Consider a vertical slab \( \{x_L, \ldots x_R\} \times [u] \)

\( B \) Segments with coordinates \((x_L, \ell_i), (x_R, r_i)\) \( B \leq w \)

\( \ell_i, r_i \) rational

if segments are nicely spaced:
(fast point location)

Fact:
Consider a vertical slab \( \{x_L, \ldots x_R\} \times [u] \)

\(B\) segments with coordinates \((x_L, \ell_i), (x_R, r_i)\) \(B \leq w\)

\(\ell_i, r_i\) rational

if segments are nicely spaced:

\(\ell_1 \leq \cdots \leq \ell_B\) \(r_1 \leq \cdots \leq r_B\)
Fact:
Consider a vertical slab \( \{x_L, \ldots x_R\} \times [u] \)

\( B \) Segments with coordinates \((x_L, \ell_i), (x_R, r_i)\) \hspace{1cm} B \leq w

\( \ell_i, r_i \) rational

if segments are nicely spaced:

\( \ell_1 \leq \cdots \leq \ell_B \hspace{1cm} r_1 \leq \cdots \leq r_B \)

\( \ell_{i+1} - \ell_i \geq \frac{\ell_B - \ell_1}{2w/B} \)
(fast point location)

Fact:
Consider a vertical slab \( \{x_L, \ldots x_R\} \times [u] \)

\( B \) Segments with coordinates \((x_L, \ell_i), (x_R, r_i)\) \( B \leq w \)

\( \ell_i, r_i \) rational

if segments are nicely spaced:

\( \ell_1 \leq \cdots \leq \ell_B \quad r_1 \leq \cdots \leq r_B \)

\( \ell_{i+1} - \ell_i \geq \frac{\ell_B - \ell_1}{2w/B} \)

\( \frac{D}{d_i} \) is not too large
Fact:
Consider a vertical slab \( \{x_L, \ldots x_R\} \times [u] \)

\( B \) Segments with coordinates \((x_L, \ell_i), (x_R, r_i)\) \( B \leq w \)

\( \ell_i, r_i \) rational

if segments are nicely spaced:

\[ \ell_1 \leq \cdots \leq \ell_B \quad r_1 \leq \cdots \leq r_B \]

\[ \ell_{i+1} - \ell_i \geq \frac{\ell_B - \ell_1}{2w/B} \]

\( \frac{D}{d_i} \) is not too large

Same on the right
Fact:
Consider a vertical slab \( \{x_L, \ldots x_R\} \times \[u\] \)
\( B \) Segments with coordinates \((x_L, \ell_i), (x_R, r_i)\) \quad B \leq w
\( \ell_i, r_i \) rational

if segments are nicely spaced:
Then:

In \( O(B) \) time, we can

build a structure that

supports point location

in constant time.
Fact:
Consider a vertical slab \( \{x_L, \ldots x_R\} \times [u]\) \(B\) Segments with coordinates \((x_L, \ell_i), (x_R, r_i)\) \(B \leq w\)
\(\ell_i, r_i\) rational

if segments are nicely spaced:
Then:

In \(O(B)\) time, we can pack all segments in \(O(1)\) words
build a structure that use parallelism to find answer
supports point location in constant time.
in constant time. \[\text{[Chan FoCS '06]}\]
(fast point location)

What if segments aren’t nicely spaced?
What if segments aren’t nicely spaced?

Choose subset that is nicely spaced and recurse
(fast point location)

What if segments aren’t nicely spaced?

Choose subset that is nicely spaced and recurse

Corollary:

support Query for point between segment \( i \) and \( i + 1 \)

in time \( O \left( 1 + \frac{B}{w} \left( \log \frac{\ell_B - \ell_1}{\ell_{i+2} - \ell_{i-1}} + \log \frac{r_B - r_1}{r_{i+2} - r_{i-1}} \right) \right) \)
sketch: (fast point location)

nicely spaced = \( \ell_{i+1} - \ell_i \geq \frac{\ell_B - \ell_1}{2w/B} \)
sketch: (fast point location)

\[
\text{nicely spaced } = \ell_{i+1} - \ell_i \geq \frac{\ell_B - \ell_1}{2^{w/B}}
\]

In each step, span of remaining segments decreases by factor of \(2^{w/B}\)
sketch: (fast point location)

nicely spaced = \( \ell_{i+1} - \ell_i \geq \frac{\ell_B - \ell_1}{2^{w/B}} \)

In each step, span of remaining segments decreases by factor of \( 2^{w/B} \)

After \( \frac{B}{w} \left( \lg \frac{\ell_B - \ell_1}{\ell_{i+2} - \ell_{i-1}} + \lg \frac{r_B - r_1}{r_{i+2} - r_{i-1}} \right) \) steps:
sketch: (fast point location)

nicely spaced = \( \ell_{i+1} - \ell_i \geq \frac{\ell_B - \ell_1}{2^{w/B}} \)

In each step, span of remaining segments decreases by factor of \( 2^{w/B} \)

After \( \frac{B}{w} \left( \lg \frac{\ell_B - \ell_1}{\ell_{i+2} - \ell_{i-1}} + \lg \frac{r_B - r_1}{r_{i+2} - r_{i-1}} \right) \) steps:

subset can’t include both segments

\( i - 1 \) and \( i + 2 \)
Think of $\lg(\ell_B - \ell_1) + \lg(r_B - r_1)$ as entropy of search region.
Think of $\lg(\ell_B - \ell_1) + \lg(r_B - r_1)$ as entropy of search region

If we take time $t$ for query

then entropy decreases by at least

$$\frac{w}{B} (\Omega(t) - 1))$$ bits
Think of $\lg(\ell_B - \ell_1) + \lg(r_B - r_1)$ as entropy of search region

If we take time $t$ for query

then entropy decreases by at least

$$\frac{w}{B} (\Omega(t) - 1)) \text{ bits}$$

Hope: total cost of recursion in UH tree

$$O(\log_B n + \lg u/\frac{w}{B}) = O(\log_B n + B) \rightarrow O(\lg n/\lg \lg n)$$
How does planar point location relate to tangent query in CH tree?

Is information progress actually maintained?

look at the geometry...
Upper Convex Chain

\[ P = \langle p_1, p_2, \ldots, p_m \rangle \]
Exterior of $P$

Upper Convex Chain

$P = \langle p_1, p_2, \ldots, p_m \rangle$
Upper Convex Chain

\[ P = \langle p_1, p_2, \ldots, p_m \rangle \]
$Z_P(p_i, p_j)$  “Zorro”

Upper Convex Chain

$P = \langle p_1, p_2, \ldots, p_m \rangle$
Fact:
Let $q$ be exterior to $P$
Test: $q \in Z_P$ in $O(1)$ time.
(compare with 3 lines)

$Z_P(p_i, p_j)$ “Zorro”

$P = \langle p_1, p_2, \ldots, p_m \rangle$

Upper Convex Chain
Fact:
for $1 \leq i < k < j < m$:

$$Z_P(p_i, p_j) = Z_P(p_i, p_k) \cup Z_P(p_k, p_j)$$
Fact:
for \(1 \leq i < k < j < m\):

\[
Z_P(p_i, p_j) = Z_P(p_i, p_k) \cup Z_P(p_k, p_j)
\]

\[
Z_P(p_i, p_k) \cap Z_P(p_k, p_j) = \emptyset
\]
Fact:

$q$ is in $Z_P(p_i, p_j)$ iff the right tangent of $q$ is in $\{p_{i+1}, \ldots, p_j\}$

$Z_P(p_i, p_j)$

\[ P = \langle p_1, p_2, \ldots, p_m \rangle \]

Upper Convex Chain
Fact:
$q$ is in $Z_P(p_i, p_j)$ iff right tangent of $q$ is in $\{p_{i+1}, \ldots, p_j\}$

Upper Convex Chain
$P = \langle p_1, p_2, \ldots, p_m \rangle$
Fact:

$q$ is in $Z_P(p_i, p_j)$ iff right tangent of $q$ is in $\{p_{i+1}, \ldots p_j\}$

$P = \langle p_1, p_2, \ldots p_m \rangle$
Fact:
$q$ is in $Z_P(p_i, p_j)$ iff
right tangent of $q$
is in $\{p_{i+1}, \ldots, p_j\}$
Fact:

$q$ is in $Z_P(p_i, p_j)$ iff right tangent of $q$ is in \{ $p_{i+1}, \ldots, p_j$ \}

$P = \langle p_1, p_2, \ldots, p_m \rangle$
Left Slab

$Z_{i,j}$

$x = 0$

$x = x(p_{i+1})$
$x = 0$

$x = x(p_{i+1})$

Left Slab

$Z_{i,j}$

Left Vertical Extent $L(Z)$

Right Vertical Extent $R(Z)$
Definition: Entropy

\[ H(Z) = \log L(Z) + \log R(Z) \]
Definition: Entropy

\[ H(Z) = \lg L(Z) + \lg R(Z) \]

Fact: If \( 1 \leq i \leq i' \leq j' \leq j < m \) then

\[ Z_P(i', j') \subseteq Z_P(i, j) \]
Definition: Entropy

\[ H(Z) = \lg L(Z) + \lg R(Z) \]

Fact: \( 1 \leq i \leq i' \leq j' \leq j < m \) then
\[ Z_P(i', j') \subseteq Z_P(i, j) \]
and
\[ H(Z_P(i', j')) \leq H(Z_P(i, j)) \]
Upper Convex Chain \( S \)

\[ H(Z) = \lg L(Z) + \lg R(Z) \]
Upper Convex Chain \( S \)

Subsequence \( P \subseteq S \)

\[
H(Z) = \lg L(Z) + \lg R(Z)
\]
Upper Convex Chain $S$

Subsequence $P \subseteq S$

Indices $1 < i < j < m$

$$H(Z) = \lg L(Z) + \lg R(Z)$$
Upper Convex Chain $S$

Subsequence $P \subseteq S$

Indices $1 < i < j < m$

$$H(Z) = \lg L(Z) + \lg R(Z)$$
Upper Convex Chain \( S \)

Subsequence \( P \subseteq S \)

Indices \( 1 < i < j < m \)

\[
H(Z) = \lg L(Z) + \lg R(Z)
\]
Upper Convex Chain $S$

Subsequence $P \subseteq S$

Indices $1 < i < j < m$

$$Z_P(i, j - 1) \cap \text{exterior}(S) \subseteq Z_S(i, j) \subseteq Z_P(i - 1, j)$$

$$H(Z) = \lg L(Z) + \lg R(Z)$$
Upper Convex Chain $S$

Subsequence $P \subseteq S$

Indices $1 < i < j < m$

$Z_P(i, j - 1) \cap \text{exterior}(S) \subseteq Z_S(i, j) \subseteq Z_P(i - 1, j)$

$H(Z_P(i, j - 1)) \leq H(Z_S(i, j)) \leq H(Z_P(i - 1, j))$

$H(Z) = \lg L(Z) + \lg R(Z)$
Lemma 9: Upper convex chain \( P = \langle p_1, \ldots, p_B \rangle \)

Given:
indices \( 1 < i < j < B - 1 \)
point \( q \) in \( Z_p(p_i, p_j) \) and its left slab
Lemma 9: Upper convex chain $P = \langle p_1, \ldots, p_B \rangle$

Given:
indices $1 < i < j < B - 1$

point $q$ in $Z_p(p_i, p_j)$ and its left slab

Build data structure that can:
Lemma 9: Upper convex chain $P = \langle p_1, \ldots, p_B \rangle$

Given:
indices $1 < i < j < B - 1$

point $q$ in $Z_p(p_i, p_j)$ and its left slab

Build data structure that can:

find index $i < k < j$ such that $q$ in $Z_p(p_k, p_{k+1})$
Lemma 9: Upper convex chain $P = \langle p_1, \ldots, p_B \rangle$

Given:
indices $1 < i < j < B - 1$

point $q$ in $Z_p(p_i, p_j)$ and its left slab

Build data structure that can:

find index $i < k < j$ such that $q$ in $Z_p(p_k, p_{k+1})$

in time:

$$t = O \left( 1 + \frac{B}{w} \left( H(Z_p(p_i, p_j)) - H(Z_p(p_{k-1}, p_{k+2})) \right) \right)$$
Sketch:

Use point location corollary

support Query for point between segment $i$ and $i + 1$
in time $O \left(1 + \frac{B}{w} \left(\lg \frac{\ell_B - \ell_1}{\ell_{i+2} - \ell_{i-1}} + \lg \frac{r_B - r_1}{r_{i+2} - r_{i-1}}\right)\right)$

Find which Zorro contains $q$
Data Structure: B-tree over Upper Hull

Node \( v \)

\[ S_v = \text{points in subtree} \]

\[ S_v \supseteq P_v = \text{at most } B \text{ points stored in } v \]
Data Structure: B-tree over Upper Hull

Node $v$

$S_v = \text{points in subtree}$

$S_v \supseteq P_v = \text{at most } B \text{ points stored in } v$
Data Structure: B-tree over Upper Hull

Node $v$

$S_v = \text{points in subtree}$

$S_v \supseteq P_v = \text{at most } B \text{ points stored in } v$
Data Structure: B-tree over Upper Hull

Node $v$

$S_v = \text{points in subtree}$

$S_v \supseteq P_v = \text{at most } B \text{ points stored in } v$
Data Structure: B-tree over Upper Hull

Node $v$

\[ S_v = \text{points in subtree} \]

\[ S_v \supseteq P_v = \text{at most } B \text{ points stored in } v \]

augmented with:

atomic heap for $x$ coordinates

PL structure of Lemma 9
Query

\( S_v \)
Query

$v$

$q$

$S_v$
Query

\[ \subseteq \]

\( S_v \subseteq P_v \)
Query

$v$

$q$

$S_v \subseteq P_v$

Children
Query

\[ S_v \subseteq P_v \]

Zorros

Children
Query

Point Location

Zorros

$S_v \subseteq P_v$

Children
Query

Point Location

Zorros

$S_v \subseteq P_v$

Recurse
Query

Zorro shrinks on each recursion

⇒ $H(Z)$ also decreases
Query

Zorro shrinks on each recursion

\[ \Rightarrow \ H(Z) \ also \ decreases \]

Each point location:

\[ H(Z) \ decreases \ by \ \frac{w}{B} \ (\Omega(t) - 1) \quad [\text{Lemma 9}] \]
Query

Zorro shrinks on each recursion

\[ \Rightarrow H(Z) \text{ also decreases} \]

Each point location:

\[ H(Z) \text{ decreases by } \frac{w}{B} \left( \Omega(t) - 1 \right) \]  

[Lemma 9]

\[ \Rightarrow \text{Total cost of point location:} \]

\[ \max H(\cdot) \text{ divided by } \frac{w}{B} \]
Query

Points on \([u] \times [u]\) grid

\[\Rightarrow -2 \log u \leq H(\cdot) \leq 2 \log u\]
Query

Points on \([u] \times [u]\) grid

\[\Rightarrow -2 \lg u \leq H(\cdot) \leq 2 \lg u\]

\[w \geq \lg u\]
Query

Points on \([u] \times [u]\) grid

\[\Rightarrow -2 \log u \leq H(\cdot) \leq 2 \log u\]

\(w \geq \log u\)

Total cost of point location:

\((\leq 2w/\frac{w}{B}) = O(B)\)
Total cost of query:

\[ O(\log_B n + B) = O(\log n / \log \log n) \]

by choosing \( B = \log^\epsilon n \)

Total cost of point location:

\[ (\leq 2w / w_B) = O(B) \]
Query: Search UH stored at root $O(\lg n / \lg \lg n)$
Query: Search UH stored at root $O(\lg n / \lg \lg n)$

Update: Update each node on leaf to root path
Query: Search UH stored at root $O(\lg n / \lg \lg n)$

Update: Update each node on leaf to root path

$O(\lg n)$ nodes $\times O(\lg n)$ time per node

$O(\lg^2 n)$ time.
Divide and Conquer Construction
Divide and Conquer Construction

Divide $P$ in half by $x$-coordinate
Divide and Conquer Construction

Divide $P$ in half by $x$-coordinate

Build Hulls
Divide and Conquer Construction

Divide $P$ in half by $x$-coordinate

Build Hulls

Bridge
Divide and Conquer Construction

Divide $P$ in half by $x$-coordinate

Build Hulls

Bridge

"Throw Away" Obscured Parts

UH(L)

Obscured Parts

UH(R)

$L$

$R$
Divide and Conquer Construction

Combined Hull UH(P)
Divide and Conquer Construction

$L$

$R$
Divide and Conquer Construction

Build Hulls

\( UH(L) \)

\( UH(R) \)

\( L \)

\( R \)

New Point
Divide and Conquer Construction

Build Hulls

UH(L)  UH(R)

New Point
Divide and Conquer Construction

Build Hulls
Bridge

UH(L)

New Point

UH(R)

L
R
Divide and Conquer Construction

Bridge

Combined Hull  UH(P)

New Point

L  R
Updates

“B-tree” actually built by compressing $\lg B - 2$ levels of binary tree
Updates

“B-tree” actually built by compressing \( \log B - 2 \) levels of binary tree

Update may change bridges on root-to-leaf path in binary tree

\[ \Rightarrow \log_B n \text{ nodes in B-tree are changed} \]
Updates

“B-tree” actually built by compressing $\lg B - 2$ levels of binary tree

Update may change bridges on root-to-leaf path in binary tree

$\Rightarrow \log_B n$ nodes in B-tree are changed

Rebuilding associated structures:

$$\log_B n \cdot B^{O(1)} = O(\lg^{1+\epsilon} n)$$
Lower Bound:
Lower Bound:
Lower Bound:

Isosceles

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$

$\theta$
Lower Bound:
Lower Bound:
Lower Bound:
Lower Bound:
Lower Bound:
Lower Bound:

isosceles

Mandatory

Marked