Computational Geometry – Exercise
Range Searching

Benjamin Niedermann
25.11.2015
Orthogonal Range Queries for $d = 2$

**Given:** Set $P$ of $n$ points in $\mathbb{R}^2$

**Find:** A data structure that efficiently answers queries of the form $[a_1, b_1] \times \cdots \times [a_d, b_d]$
Orthogonal Range Queries for $d = 2$

**Given:** Set $P$ of $n$ points in $\mathbb{R}^2$

**Find:** A data structure that efficiently answers queries of the form $[a_1, b_1] \times \cdots \times [a_d, b_d]$

**Solutions:**

- one search tree, alternate search for $x$ and $y$ coordinates
  $\rightarrow$ **kd-Tree**

- primary search tree on $x$-coordinates, several secondary search trees on $y$-coordinates
  $\rightarrow$ **Range Tree**
Orthogonal Range Queries for $d = 2$

**Given:** Set $P$ of $n$ points in $\mathbb{R}^2$

**Find:** A data structure that efficiently answers queries of the form $[a_1, b_1] \times \cdots \times [a_d, b_d]$

**Solutions:**

- **one** search tree, alternate search for $x$ and $y$ coordinates
  $\rightarrow$ **kd-Tree**

- **primary** search tree on $x$-coordinates, several **secondary** search trees on $y$-coordinates
  $\rightarrow$ **Range Tree**
$kd$-Trees: Example
$kd$-Trees: Example
$kd$-Trees: Example

\[ p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_9, p_{10} \]
$kd$-Trees: Example

$\ell_1$

$\ell_2$

$p_1$, $p_2$, $p_3$, $p_4$, $p_5$, $p_6$, $p_7$, $p_8$, $p_9$, $p_{10}$
**kd-Trees: Example**

Diagram showing a set of points \( p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_9, p_{10} \) in a 2D space with two lines \( \ell_1 \) and \( \ell_2 \). The points are organized in a tree structure, with \( \ell_2 \) as the root node and \( \ell_1 \) as a child node.
$kd$-Trees: Example
$kd$-Trees: Example
$kd$-Trees: Example
**kd-Trees: Example**

![kd-Trees Diagram]

The diagram illustrates a kd-tree with points labeled $p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_9, p_{10}$ and lines labeled $\ell_1, \ell_2, \ell_3, \ell_4, \ell_5, \ell_6, \ell_7, \ell_8$. The tree structure shows how points are partitioned into subsets along different axes to form a balanced tree.
$kd$-Trees: Example
**$kd$-Trees: Example**

![Diagram of $kd$-Trees]

- $p_1$, $p_2$, $p_3$, $p_4$, $p_5$, $p_6$, $p_7$, $p_8$, $p_9$, $p_{10}$
- $\ell_1$, $\ell_2$, $\ell_3$, $\ell_4$, $\ell_5$, $\ell_6$, $\ell_7$, $\ell_8$

Diagram illustrating the construction of a $kd$-tree with points and planes.
$kd$-Trees: Example
**kd-Trees: Example**

- **kd-Trees** are a type of data structure used to organize data points in a multi-dimensional space. Each node in a kd-tree represents a hyperplane that splits the space into two subspaces.

- In the example shown, the tree is rooted at **$l_1$**, and each node represents a hyperplane. The children of a node in the tree are split based on the dimension of the hyperplane.

- The tree structure allows for efficient search, insertion, and deletion operations in multi-dimensional datasets.
kd-Trees: Example
$kd$-Trees: Example
$kd$-Trees: Example
kd-Trees: Example
kd-Trees: Example
$kd$-Trees: Example
$kd$-Trees: Example
kd-Trees: Example
$kd$-Trees: Example
$kd$-Trees: Example
$kd$-Trees: Example
kd-Trees: Example
$kd$-Trees: Example
kd-Trees: Example
kd-Trees: Example
**kd-Trees: Example**
Range Queries in a $kd$-Tree
Range Queries in a $kd$-Tree

SearchKdTree($v, R$)

if $v$ leaf then
  report point $p$ in $v$ when $p \in R$
else
  if region(lc($v$)) $\subseteq R$ then
    ReportSubtree(lc($v$))
  else
    if region(lc($v$)) $\cap R \neq \emptyset$ then
      SearchKdTree(lc($v$), $R$)
    else
      if region(rc($v$)) $\subseteq R$ then
        ReportSubtree(rc($v$))
      else
        if region(rc($v$)) $\cap R \neq \emptyset$ then
          SearchKdTree(rc($v$), $R$)
Range Queries in a $kd$-Tree

```
SearchKdTree(v, R)
    if v leaf then
        report point p in v when $p \in R$
    else
        if region(lc(v)) $\subseteq$ R then
            ReportSubtree(lc(v))
        else
            if region(lc(v)) $\cap$ R $\neq \emptyset$ then
                SearchKdTree(lc(v), R)
            else
                if region(rc(v)) $\subseteq$ R then
                    ReportSubtree(rc(v))
                else
                    if region(rc(v)) $\cap$ R $\neq \emptyset$ then
                        SearchKdTree(rc(v), R)
```

Range Queries in a $kd$-Tree

\[
\text{SearchKdTree}(v, R) \\
\begin{align*}
\text{if} \ v \ &\text{leaf then} \\
&\text{report point } p \text{ in } v \text{ when } p \in R \\
\text{else} \\
&\text{if region(lc}(v)\text{)) } \subseteq R \text{ then} \\
&\quad \text{ReportSubtree(lc}(v)\text{))} \\
&\text{else} \\
&\quad \text{if region(lc}(v)\text{)) } \cap R \neq \emptyset \text{ then} \\
&\quad \quad \text{SearchKdTree(lc}(v)\text{), } R \\
&\quad \text{if region(rc}(v)\text{)) } \subseteq R \text{ then} \\
&\quad \quad \text{ReportSubtree(rc}(v)\text{))} \\
&\quad \text{else} \\
&\quad \quad \text{if region(rc}(v)\text{)) } \cap R \neq \emptyset \text{ then} \\
&\quad \quad \quad \text{SearchKdTree(rc}(v)\text{), } R
\end{align*}
\]
Range Queries in a $kd$-Tree

SearchKdTree($v, R$)

- if $v$ leaf then
  - report point $p$ in $v$ when $p \in R$
- else
  - if region(lc($v$)) $\subseteq R$ then
    - ReportSubtree(lc($v$))
  - else
    - if region(lc($v$)) $\cap R \neq \emptyset$ then
      - SearchKdTree(lc($v$), $R$)
    - else
      - if region(rc($v$)) $\subseteq R$ then
        - ReportSubtree(rc($v$))
      - else
        - if region(rc($v$)) $\cap R \neq \emptyset$ then
          - SearchKdTree(rc($v$), $R$)
Range Queries in a $kd$-Tree

SearchKdTree($v, R$)

if $v$ leaf then
  report point $p$ in $v$ when $p \in R$

else
  if region(lc($v$)) $\subseteq R$ then
    ReportSubtree(lc($v$))
  else
    if region(lc($v$)) $\cap R \neq \emptyset$ then
      SearchKdTree(lc($v$), $R$)
    else
      if region(rc($v$)) $\subseteq R$ then
        ReportSubtree(rc($v$))
      else
        if region(rc($v$)) $\cap R \neq \emptyset$ then
          SearchKdTree(rc($v$), $R$)
Range Queries in a $kd$-Tree

```
SearchKdTree(v, R)
if v leaf then
    report point p in v when p ∈ R
else
    if region(lc(v)) ⊆ R then
        ReportSubtree(lc(v))
    else
        if region(lc(v)) ∩ R ≠ ∅ then
            SearchKdTree(lc(v), R)
        else
            if region(rc(v)) ⊆ R then
                ReportSubtree(rc(v))
            else
                if region(rc(v)) ∩ R ≠ ∅ then
                    SearchKdTree(rc(v), R)
```

Range Queries in a $kd$-Tree

$$\text{SearchKdTree}(v, R)$$

- **if** $v$ leaf **then**
  - report point $p$ in $v$ when $p \in R$
- **else**
  - **if** region(lc($v$)) $\subseteq R$ **then**
    - ReportSubtree(lc($v$))
  - **else**
    - **if** region(lc($v$)) $\cap R \neq \emptyset$ **then**
      - SearchKdTree(lc($v$), $R$)
    - **if** region(rc($v$)) $\subseteq R$ **then**
      - ReportSubtree(rc($v$))
    - **else**
      - **if** region(rc($v$)) $\cap R \neq \emptyset$ **then**
        - SearchKdTree(rc($v$), $R$)
Range Queries in a $kd$-Tree

SearchKdTree($v, R$)

\[
\begin{cases}
\text{if } v \text{ leaf then} \\
\text{report point } p \text{ in } v \text{ when } p \in R \\
\text{else} \\
\text{if } \text{region}(\text{lc}(v)) \subseteq R \text{ then} \\
\text{ReportSubtree}\left(\text{lc}(v)\right) \\
\text{else} \\
\text{if } \text{region}(\text{lc}(v)) \cap R \neq \emptyset \text{ then} \\
\text{SearchKdTree}\left(\text{lc}(v), R\right) \\
\text{if } \text{region}(\text{rc}(v)) \subseteq R \text{ then} \\
\text{ReportSubtree}\left(\text{rc}(v)\right) \\
\text{else} \\
\text{if } \text{region}(\text{rc}(v)) \cap R \neq \emptyset \text{ then} \\
\text{SearchKdTree}\left(\text{rc}(v), R\right)
\end{cases}
\]
Range Queries in a $kd$-Tree

SearchKdTree($v$, $R$)

if $v$ leaf then
    report point $p$ in $v$ when $p \in R$
else
    if region(lc($v$)) $\subseteq R$ then
        ReportSubtree(lc($v$))
    else
        if region(lc($v$)) $\cap R \neq \emptyset$ then
            SearchKdTree(lc($v$), $R$)
        else
            if region(rc($v$)) $\subseteq R$ then
                ReportSubtree(rc($v$))
            else
                if region(rc($v$)) $\cap R \neq \emptyset$ then
                    SearchKdTree(rc($v$), $R$)
Range Queries in a \(kd\)-Tree

\[
\text{SearchKdTree}(v, R) \\
\quad \text{if } v \text{ leaf then} \\
\quad \quad \text{report point } p \text{ in } v \text{ when } p \in R \\
\quad \text{else} \\
\quad \quad \text{if } \text{region}(\text{lc}(v)) \subseteq R \text{ then} \\
\quad \quad \quad \text{ReportSubtree}(\text{lc}(v)) \\
\quad \quad \text{else} \\
\quad \quad \quad \text{if } \text{region}(\text{lc}(v)) \cap R \neq \emptyset \text{ then} \\
\quad \quad \quad \quad \text{SearchKdTree}(\text{lc}(v), R) \\
\quad \quad \quad \text{if } \text{region}(\text{rc}(v)) \subseteq R \text{ then} \\
\quad \quad \quad \quad \text{ReportSubtree}(\text{rc}(v)) \\
\quad \quad \quad \text{else} \\
\quad \quad \quad \quad \text{if } \text{region}(\text{rc}(v)) \cap R \neq \emptyset \text{ then} \\
\quad \quad \quad \quad \quad \text{SearchKdTree}(\text{rc}(v), R)
\]
Range Queries in a \( kd \)-Tree

\[
\text{SearchKdTree}(v, R) \\
\quad \text{if } v \text{ leaf then} \\
\quad \quad \text{report point } p \text{ in } v \text{ when } p \in R \\
\quad \text{else} \\
\quad \quad \text{if region(lc}(v)\text{)) } \subseteq R \text{ then} \\
\quad \quad \quad \text{ReportSubtree(lc}(v)\text{))} \\
\quad \quad \text{else} \\
\quad \quad \quad \text{if region(lc}(v)\text{)) } \cap R \neq \emptyset \text{ then} \\
\quad \quad \quad \quad \text{SearchKdTree(lc}(v)\text{), } R \\
\quad \quad \text{if region(rc}(v)\text{)) } \subseteq R \text{ then} \\
\quad \quad \quad \text{ReportSubtree(rc}(v)\text{))} \\
\quad \quad \text{else} \\
\quad \quad \quad \text{if region(rc}(v)\text{)) } \cap R \neq \emptyset \text{ then} \\
\quad \quad \quad \quad \text{SearchKdTree(rc}(v)\text{), } R \
\]
Range Queries in a $kd$-Tree

```
SearchKdTree(v, R)
    if v leaf then
        report point $p$ in $v$ when $p \in R$
    else
        if region(lc($v$)) $\subseteq R$ then
            ReportSubtree(lc($v$))
        else
            if region(lc($v$)) $\cap R \neq \emptyset$ then
                SearchKdTree(lc($v$), R)
            else
                if region(rc($v$)) $\subseteq R$ then
                    ReportSubtree(rc($v$))
                else
                    if region(rc($v$)) $\cap R \neq \emptyset$ then
                        SearchKdTree(rc($v$), R)
```

[Diagram of a $kd$-Tree with points $p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_9, p_{10}, p_{11}, p_{12}, p_{13}$]
Range Queries in a $kd$-Tree

SearchKdTree($v$, $R$)

\[
\begin{aligned}
&\text{if } v \text{ leaf then} \\
&\quad \text{report point } p \text{ in } v \text{ when } p \in R \\
&\text{else} \\
&\quad \text{if } \text{region}(\text{l}c(v)) \subseteq R \text{ then} \\
&\quad \quad \text{ReportSubtree}(\text{l}c(v)) \\
&\quad \text{else} \\
&\quad \quad \text{if } \text{region}(\text{l}c(v)) \cap R \neq \emptyset \text{ then} \\
&\quad \quad \quad \text{SearchKdTree}(\text{l}c(v), R) \\
&\quad \quad \text{if } \text{region}(\text{r}c(v)) \subseteq R \text{ then} \\
&\quad \quad \quad \text{ReportSubtree}(\text{r}c(v)) \\
&\quad \quad \text{else} \\
&\quad \quad \quad \text{if } \text{region}(\text{r}c(v)) \cap R \neq \emptyset \text{ then} \\
&\quad \quad \quad \quad \text{SearchKdTree}(\text{r}c(v), R) \\
\end{aligned}
\]
Range Queries in a $kd$-Tree

$\text{SearchKdTree}(v, R)$

\begin{align*}
\text{if } v \text{ leaf then} & \quad \text{report point } p \text{ in } v \text{ when } p \in R \\
\text{else} & \\
\text{if } \text{region}(\text{lc}(v)) \subseteq R \text{ then} & \quad \text{ReportSubtree}(\text{lc}(v)) \\
\text{else} & \\
\text{if } \text{region}(\text{lc}(v)) \cap R \neq \emptyset \text{ then} & \quad \text{SearchKdTree}(\text{lc}(v), R) \\
\text{if } \text{region}(\text{rc}(v)) \subseteq R \text{ then} & \quad \text{ReportSubtree}(\text{rc}(v)) \\
\text{else} & \\
\text{if } \text{region}(\text{rc}(v)) \cap R \neq \emptyset \text{ then} & \quad \text{SearchKdTree}(\text{rc}(v), R)
\end{align*}
Exercise 1

Claim: Queries in $O(\sqrt{n} + k)$ time

$Q(n) = $ Number of checked regions.

a) Show the following recurrence.

$$Q(n) = \begin{cases} 
O(1) & \text{, for } n = 1 \\
2 + 2Q(n/4) & \text{, for } n > 1 
\end{cases}$$
Range Queries in $kd$-Trees

The diagram illustrates the structure of a $kd$-tree, with points $p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_9, p_{10}, p_{11}, p_{12}, p_{13}$ placed at various nodes. The tree is constructed such that each node represents a hyperrectangle, with the points falling within the hyperrectangle stored at that node.
Range Queries in $kd$-Trees

ReportSubtree in $O(k)$
ReportSubtree in $O(k)$

How many nodes do we consider? Each node corresponds with one recursion step.
Range Queries in $kd$-Trees

How many nodes do we consider? Each node corresponds with one recursion step.

ReportSubtree in $O(k)$
Range Queries in $kd$-Trees

How many nodes do we consider? Each node corresponds with one recursion step.

ReportSubtree in $O(k)$

each node $\triangleq$ intersected region
Range Queries in $kd$-Trees

![kd-tree diagram]
How many regions are intersected by a vertical line?
Range Queries in $kd$-Trees
Range Queries in $kd$-Trees

The diagram illustrates a $kd$-Tree with the root node marked with the label "root." The tree structure is shown with points $p_1$ to $p_{13}$ scattered throughout the area. The points are connected to form the tree structure, with branches indicating the hierarchical division of the space.
Range Queries in $kd$-Trees

root
Range Queries in $kd$-Trees

Conjecture:

$$Q(n) = 1 + Q(n/2)$$
Range Queries in $kd$-Trees

Conjecture:

$Q(n) = 1 + Q(n/2)$

Problem?

ℓ intersects both children of left child of root
Conjecture:

\[ Q(n) = 1 + Q(n/2) \]

Problem?

\( \ell \) intersects both children of left child of root
Range Queries in $kd$-Trees

**Conjecture:**

$$Q(n) = 1 + Q(n/2)$$

**Problem?**

\( \ell \) intersects both children of left child of root
Range Queries in $kd$-Trees

Conjecture:

$$Q(n) = 1 + Q(n/2)$$

Problem?

$\ell$ intersects both children of left child of root
Range Queries in $kd$-Trees

**Conjecture:**

$$Q(n) = 1 + Q(n/2)$$

**Problem?**

\(\ell\) intersects both children of left child of root
Range Queries in $kd$-Trees

**Conjecture:**

\[ Q(n) = 1 + Q(n/2) \]

**Problem?**

\( \ell \) intersects both children of left child of root
Conjecture:

\[ Q(n) = 1 + Q(n/2) \]

Problem?

\( \ell \) intersects both children of left child of root
Range Queries in $kd$-Trees

Conjecture:

\[ Q(n) = 1 + Q(n/2) \]

Problem?

\( \ell \) intersects both children of left child of root

Enforce same situation!
Range Queries in $kd$-Trees

\[ Q(n) = \begin{cases} 
\mathcal{O}(1), & \text{for } n = 1 \\
2 + 2Q(n/4), & \text{for } n > 1 
\end{cases} \]

$Q(n) =$ number of intersected regions of a $kd$-tree whose root contains a vertical splitting line.
Range Queries in \( kd \)-Trees

\[
Q(n) = \begin{cases} 
O(1), & \text{for } n = 1 \\
2 + 2Q(n/4), & \text{for } n > 1 
\end{cases}
\]

\( Q(n) \) = number of intersected regions of a \( kd \)-tree whose root contains a vertical splitting line.
Range Queries in $kd$-Trees

$Q(n) = \begin{cases} 
\mathcal{O}(1), & \text{for } n = 1 \\
2 + 2Q(n/4), & \text{for } n > 1 
\end{cases}$

$Q(n) =$ number of intersected regions of a $kd$-tree whose root contains a vertical splitting line.
Exercise 1

Claim: Queries in $O(\sqrt{n} + k)$ time

$Q(n) = \text{Number of checked regions.}$

a) Show the following recurrence.

$$Q(n) = \begin{cases} O(1) \quad , \text{ for } n = 1 \\ 2 + 2Q(n/4) \quad , \text{ for } n > 1 \end{cases}$$
Exercise 1

Claim: Queries in $O(\sqrt{n} + k)$ time

$Q(n) =$ Number of checked regions.

a) Show the following recurrence.

$$Q(n) = \begin{cases} 
O(1) & \text{, for } n = 1 \\
2 + 2Q(n/4) & \text{, for } n > 1 
\end{cases}$$

b) Resolve recurrence: $Q(n) = O(\sqrt{n})$. 
Exercise 1

Claim: Queries in $O(\sqrt{n} + k)$ time

$Q(n) =$ Number of checked regions.

a) Show the following recurrence.

$$Q(n) = \begin{cases} 
O(1) & , \text{ for } n = 1 \\
2 + 2Q(n/4) & , \text{ for } n > 1 
\end{cases}$$

b) Resolve recurrence: $Q(n) = \Theta(\sqrt{n})$.

c) $\Omega(\sqrt{n})$ lower bound for range queries in $kd$-trees
SearchKdTree\((v, R)\)

\[
\text{if } v \text{ leaf then}
\]
report point \(p\) in \(v\) when \(p \in R\)

\[
\text{else}
\]
\[
\text{if } \text{region(lc}(v)) \subseteq R \text{ then}
\]
| ReportSubtree(lc\((v)\))
\[
\text{else}
\]
\[
\text{if } \text{region(lc}(v)) \cap R \neq \emptyset \text{ then}
\]
| SearchKdTree(lc\((v), R)\))
\[
\text{if } \text{region(rc}(v)) \subseteq R \text{ then}
\]
| ReportSubtree(rc\((v)\))
\[
\text{else}
\]
\[
\text{if } \text{region(rc}(v)) \cap R \neq \emptyset \text{ then}
\]
| SearchKdTree(rc\((v), R)\))

\(c) \ \Omega(\sqrt{n})\) lower bound for range queries \(kd\)-trees
Range Trees

**Idea:** Use 1-dimensional search trees on two levels:
- a 1d search tree $T_x$ on $x$-coordinates
Range Trees

**Idea:** Use 1-dimensional search trees on two levels:
- a 1d search tree $T_x$ on $x$-coordinates
- in each node $v$ of $T_x$ a 1d search tree $T_y(v)$ stores the canonical subset $P(v)$ on $y$-coordinates
Range Trees

**Idea:** Use 1-dimensional search trees on two levels:
- a 1d search tree $T_x$ on $x$-coordinates
- in each node $v$ of $T_x$ a 1d search tree $T_y(v)$ stores the canonical subset $P(v)$ on $y$-coordinates
- Compute the points by $x$-query in $T_x$ and subsequent $y$-queries in the auxiliary structures $T_y$ for the subtrees in $T_x$
Exercise 2

'partial match queries'

Report all points with $y = 7$. 
Exercise 2

'partial match queries'

Report all points with $y = 7$.

a) How to apply $kd$-trees?
Exercise 2

'partial match queries'

Report all points with $y = 7$.

b) How to apply range-trees?
Exercise 2

`'partial match queries'`

Report all points with $y = 7$.

c) Find: Datatstructure that solves problem in $\mathcal{O}(\log n + k)$ time and $\mathcal{O}(n)$ storage.
Exercise 3

How many points lie in that rectangle?

range counting query

Requirement: Without additive constant $\mathcal{O}(k)$ in running time.
Exercise 3

How many points lie in that rectangle?

range counting query

**Requirement:** Without additive constant $O(k)$ in running time.

a) Adapt 1-dim range-tree for range counting queries in $O(\log n)$. 
Exercise 3

1dRangeQuery$(T, x, x')$

\[
v_{\text{split}} \leftarrow \text{FindSplitNode}(T, x, x')
\]

\[\text{if } v_{\text{split}} \text{ ist Blatt then prüfe } v_{\text{split}}\]

\text{else}

\[
v \leftarrow \text{l}(v_{\text{split}})
\]

\[\text{while } v \text{ kein Blatt do}
\]

[\text{if } x \leq x_v \text{ then}
\]

\[
\text{ReportSubtree}(\text{rc}(v)); v \leftarrow \text{l}(v)
\]

[\text{else } v \leftarrow \text{rc}(v)
\]

\[\text{prüfe } v \]

[\text{// analog für } x' \text{ und } \text{rc}(v_{\text{split}})\]

\[\]

a) Adapt 1-dim range-tree for range counting queries in $O(\log n)$.
Exercise 3

How many points lie in that rectangle?

range counting query

**Requirement:** Without additive constant $O(k)$ in running time.

b) How to solve the $d$-dimensional problem.
Exercise 3

2dim Range-Trees:

**Requirement:** Without additive constant $O(k)$ in running time.

b) How to solve the $d$-dimensional problem.
Exercise 4
Exercise 4

Which rectangles completely lie in this rectangle?
Exercise 4

Which rectangles completely lie in this rectangle?

Data structure with $O(n \log^3 n)$ storage and $O(\log^4 n + k)$ query time.
Exercise 4

Data structure with $O(n \log^3 n)$ storage and $O(\log^4 n + k)$ query time.
Exercise 4

Which polygons completely lie in this rectangles?

Data structure with $O(n \log^3 n)$ storage and $O(\log^4 n + k)$ query time.