Übung Algorithmische Geometrie
Quad-trees

Benjamin Niedermann
16.12.2015
Motivation: Meshing PC Board Layouts

To simulate the heat produced on boards we can use the *finite element method* (FEM):

- decompose the board in small homogeneous elements (e.g., triangles) → mesh
- heat generation and impact on neighbors for each element known
- approximate numerically the entire heat generation of board
Motivation: Meshing PC Board Layouts

To simulate the heat produced on boards we can use the finite element method (FEM):
- decompose the board in small homogeneous elements (e.g., triangles) → mesh
- heat generation and impact on neighbors for each element known
- approximate numerically the entire heat generation of board

Quality properties of FEM:
- the finer the mesh, the better the approximation
- the larger the mesh, the faster the calculation
- the more compact the elements, the faster the convergence
**Motivation: Meshing PC Board Layouts**

To simulate the heat produced on boards we can use the *finite element method* (FEM):

- decompose the board in small homogeneous elements (e.g., triangles) → mesh
- heat generation and impact on neighbors for each element known
- approximate numerically the entire heat generation of board

**Quality properties of FEM:**

- the finer the mesh, the better the approximation
- the larger the mesh, the faster the calculation
- the more compact the elements, the faster the convergence

**Goal:**

- adaptive mesh size (small on materials, otherwise coarser)
- fat triangles (not too narrow)
Adaptive Triangular Mesh

**Given:** Square $Q = [0, U] \times [0, U]$ for power of two $U = 2^j$ with *octilinear*, integer-coordinate polygons inside.

**Goal:** Triangular mesh for $Q$ with the following properties
Adaptive Triangular Mesh

Given: Square $Q = [0, U] \times [0, U]$ for power of two $U = 2^j$ with octilinear, integer-coordinate polygons inside.

Goal: Triangular mesh for $Q$ with the following properties

- no triangle vertex in interior of triangular mesh
Adaptive Triangular Mesh

**Given:** Square $Q = [0, U] \times [0, U]$ for power of two $U = 2^j$ with *octilinear*, integer-coordinate polygons inside.

**Goal:** Triangular mesh for $Q$ with the following properties
- no triangle vertex in interior of triangular mesh
- input edges must be part of the triangulation
Adaptive Triangular Mesh

**Given:** Square $Q = [0, U] \times [0, U]$ for power of two $U = 2^j$ with octilinear, integer-coordinate polygons inside.

**Goal:** Triangular mesh for $Q$ with the following properties:
- no triangle vertex in interior of triangular mesh
- input edges must be part of the triangulation
- triangle angle between 45° and 90°
Adaptive Triangular Mesh

**Given:** Square $Q = [0, U] \times [0, U]$ for power of two $U = 2^j$

with *octilinear*, integer-coordinate polygons inside.

**Goal:** Triangular mesh for $Q$ with the following properties

- no triangle vertex in interior of triangular mesh
- input edges must be part of the triangulation
- triangle angle between $45^\circ$ and $90^\circ$
- **adaptive** (i.e., fine at the polygon edges, otherwise coarser)
Adaptive Triangular Mesh

**Given:** Square $Q = [0, U] \times [0, U]$ for power of two $U = 2^j$ with *octilinear*, integer-coordinate polygons inside.

**Goal:** Triangular mesh for $Q$ with the following properties
- no triangle vertex in interior of triangular mesh
- input edges must be part of the triangulation
- triangle angle between $45^\circ$ and $90^\circ$
- **adaptive** (i.e., fine at the polygon edges, otherwise coarser)

Do we already have meaningful triangulations of $Q$?
Exercise 1

Delaunay Triangulierung $\leftrightarrow$ Meshing

Meshing yields only non-obtuse triangle, i.e., no angle is larger than $90^\circ$. 
Exercise 1

Delaunay Triangulierung $\leftrightarrow$ Meshing

Meshing yields only non-obtuse triangle, i.e., no angle is larger than 90°.
Exercise 1

**Delaunay Triangulierung ↔ Meshing**

Meshing yields only non-obtuse triangle, i.e., no angle is larger than 90°.

Let $\mathcal{T}$ be a triangulation of a finite set $P \subseteq \mathbb{R}^2$ of points, such that each triangle is non-obtuse.

Show that $\mathcal{T}$ is a Delaunay triangulation.
Characterization

Theorem about Voronoi-Diagram:

- point \( q \) is a Voronoy-vertex
  \[ \iff |C_P(q) \cap P| \geq 3, \]
- bisector \( b(p_i, p_j) \) defines a Voronoi-edge
  \[ \iff \exists q \in b(p_i, p_j) \text{ with } C_P(q) \cap P = \{p_i, p_j\}. \]

**Theorem 4:** Let \( P \) be a set of points.

- Points \( p, q, r \) are vertices of the same face of \( DG(P) \iff \) circle through \( p, q, r \) is empty
- Edge \( pq \) is in \( DG(P) \iff \) there is an empty circle \( C_{p,q} \) through \( p \) and \( q \)

**Theorem 5:** Let \( P \) be a set of points and let \( T \) be a triangulation of \( P \). \( T \) is Delaunay-Triangulation
\[ \iff \text{the circumcircle of each triangle has an empty interior.} \]
Thales’s Theorem

**Theorem 2:** If $a$, $b$ and $c$ are points on a circle where the segment $ab$ is a diameter of the circle, then the angle $\angle bca$ is a right angle.
Thales’s Theorem

Theorem 2: If $a$, $b$ and $c$ are points on a circle where the segment $ab$ is a diameter of the circle, then the angle $\angle bca$ is a right angle.

Theorem 2': Consider a circle $C$ through $a$, $b$, $c$. For any point $c'$ on $C$ on the same side of $ab$ as $c$, holds that $\angle acb = \angle ac'b$. For any point $d$ inside $C$ holds that $\angle adb > \angle acb$, and for point $e$ outside $C$, holds that $\angle aeb < \angle acb$.

$\angle aeb < \angle acb = \angle ac'b < \angle adb$
Exercise 1

Assume: There is a triangulation $T$ containing only non-obtuse triangles, but $T$ is not a Delaunay triangulation.
Exercise 1

Assume: There is a triangulation $T$ containing only non-obtuse triangles, but $T$ is not a Delaunay triangulation.

$\exists \Delta = (pqr)$ whose circumcircle contains a point $z \in P \setminus \{p, q, r\}$.
Exercise 1

Assume: There is a triangulation $T$ containing only non-obtuse triangles, but $T$ is not a Delaunay triangulation.

There exists a triangle $\Delta = (pqr)$ whose circumcircle contains a point $z \in P \setminus \{p, q, r\}$. $z$ can be chosen such that it is connected with at least two points of $\Delta$. 

![Diagram showing triangle $\Delta$ with circumcircle and point $z$]
Exercise 1

Assume: There is a triangulation $T$ containing only non-obtuse triangles, but $T$ is not a Delaunay triangulation.

$\exists \Delta = (pqr)$ whose circumcircle contains a point $z \in P \setminus \{p, q, r\}$. $z$ can be chosen, such that it is connected with at least two points of $\Delta$.

Case 1: $z$ lies outside of $\Delta$

w.l.o.g. let $z$ be connected with $p$ and $q$. 

![Diagram](attachment:image.png)
Exercise 1

Assume: There is a triangulation $T$ containing only non-obtuse triangles, but $T$ is not a Delaunay triangulation.

$\exists \Delta = (pqr)$ whose circumcircle contains a point $z \in P \setminus \{p, q, r\}$.

$z$ can be chosen, such that it is connected with at least two points of $\Delta$

Case 1: $z$ lies outside of $\Delta$

w.l.o.g. let $z$ be connected with $p$ and $q$.

Let $C'$ be circle through $p$ and $q$ and let the center $m'$ be the midpoint of $pq$. 

\[
\begin{align*}
\text{Diagram:} & \quad r \\
p & \quad m' \\
m & \quad z \\
q & \quad C \\
C' & \quad \text{Circle through } p, q \\
\end{align*}
\]
Exercise 1

Assume: There is a triangulation $T$ containing only non-obtuse triangles, but $T$ is not a Delaunay triangulation.

$\exists \Delta = (pqr)$ whose circumcircle contains a point $z \in P \setminus \{p, q, r\}$.

$z$ can be chosen, such that it is connected with at least two points of $\Delta$

Case 1: $z$ lies outside of $\Delta$

w.l.o.g. let $z$ be connected with $p$ and $q$.

Let $C'$ be circle through $p$ and $q$ and let the center $m'$ be the midpoint of $pq$.

Theorem 2': $r$ is not contained in $C'$

$\rightarrow$ Since $z$ is contained in $C$, $z$ lies also in $C'$. 
Exercise 1

Assume: There is a triangulation $T$ containing only non-obtuse triangles, but $T$ is not a Delaunay triangulation.

$\exists \Delta = (pqr)$ whose circumcircle contains a point $z \in P \setminus \{p, q, r\}$.

$z$ can be chosen, such that it is connected with at least two points of $\Delta$

Case 1: $z$ lies outside of $\Delta$

w.l.o.g. let $z$ be connected with $p$ and $q$.

Let $C'$ be circle through $p$ and $q$ and let the center $m'$ be the midpoint of $pq$.

Theorem 2': $r$ is not contained in $C'$

$\rightarrow$ Since $z$ is contained in $C$, $z$ lies also in $C'$.

Theorem 2': Angle at $z$ is larger than $90^\circ$
Exercise 1

Assume: There is a triangulation $T$ containing only non-obtuse triangles, but $T$ is not a Delaunay triangulation.

$\exists \Delta = (pqr)$ whose circumcircle contains a point $z \in P \setminus \{p, q, r\}$.

$z$ can be chosen, such that it is connected with at least two points of $\Delta$

Case 1: $z$ lies outside of $\Delta$

w.l.o.g. let $z$ be connected with $p$ and $q$.

Let $C'$ be circle through $p$ and $q$ and let the center $m'$ be the midpoint of $pq$.

Theorem 2': $r$ is not contained in $C'$

$\Rightarrow$ Since $z$ is contained in $C$, $z$ lies also in $C'$.

Theorem 2': Angle at $z$ is larger than $90^\circ$

Case 2: $z$ lies inside of $\Delta$

Similar arguments
**Quadtrees**

**Def.:** A **quadtree** is a rooted tree, where each internal node has 4 children. Each node corresponds to a square, and the squares of the leaves form a partition of the root square.
Def.: A **quadtree** is a rooted tree, where each internal node has 4 children. Each node corresponds to a square, and the squares of the leaves form a partition of the root square.
Example
Example
Example
Example
Example
Example
Quadtree Properties

The recursive definition of quadtrees leads directly to an algorithm for constructing them.
Quadtrees Properties
The recursive definition of quadtrees leads directly to an algorithm for constructing them.

What is the depth of a quadtree on $n$ points?
Quadtree Properties

The recursive definition of quadtrees leads directly to an algorithm for constructing them.

What is the depth of a quadtree on \( n \) points?

**Lemma 1:** The depth of \( \mathcal{T}(P) \) is at most \( \log(s/c) + 3/2 \), where \( c \) is the smallest distance in \( P \) and \( s \) is the length of a side of \( Q \).
Quadtree Properties

The recursive definition of quadtrees leads directly to an algorithm for constructing them.

What is the depth of a quadtree on \( n \) points?

**Lemma 1:** The depth of \( T(P) \) is at most \( \log(s/c) + 3/2 \), where \( c \) is the smallest distance in \( P \) and \( s \) is the length of a side of \( Q \).

**Theorem 1:** A quadtree \( T(P) \) on \( n \) points with depth \( d \) has \( O((d + 1)n) \) nodes and can be constructed in \( O((d + 1)n) \) time.
Exercise 2

Compressed Quadtrees
Exercise 2

Compressed Quadtrees

[Diagram of a compressed quadtree structure]

[Image of a color-coded quadtree partitioning]
Exercise 2

Compressed Quadtrees
Exercise 2

Compressed Quadtrees
Exercise 2

Compressed Quadtrees
Exercise 2

Compressed Quadtrees
Exercise 2

Compressed Quadtrees
Exercise 2

Compressed Quadtrees

- Number of nodes is $O(n)$ (instead of $O((d + 1)n)$)
- Running time for transformation?
- Running time for construction?
Exercise 3

Balanced Quadtrees
Balanced Quadtrees

Def.: A quadtree is called **balanced** if any two neighboring squares differ at most a factor two in size. A quadtree is called balanced if its subdivision is balanced.
Balanced Quadtrees

Def.: A quadtree is called balanced if any two neighboring squares differ at most a factor two in size. A quadtree is called balanced if its subdivision is balanced.
Balanced Quadtrees

**Def.** A quadtree is called **balanced** if any two neighboring squares differ at most a factor two in size. A quadtree is called balanced if its subdivision is balanced.
Balancing Quadtrees

BalanceQuadtree($\mathcal{T}$)

**Input:** Quadtree $\mathcal{T}$

**Output:** A balanced version of $\mathcal{T}$

$L \leftarrow$ List of all leaves of $\mathcal{T}$;

while $L$ not empty do

$\mu \leftarrow$ extract leaf from $L$;

if $\mu.Q$ too large then

Divide $\mu.Q$ into four parts and put four leaves in $\mathcal{T}$;
add new leaves to $L$;

if $\mu.Q$ now has neighbors that are too large then add it to $L$;

return $\mathcal{T}$
Balancing Quadtrees

BalanceQuadtree($\mathcal{T}$)

**Input:** Quadtree $\mathcal{T}$

**Output:** A balanced version of $\mathcal{T}$

$L \leftarrow$ List of all leaves of $\mathcal{T}$;

**while** $L$ not empty **do**

\[ \mu \leftarrow \text{extract leaf from } L; \]

**if** $\mu.Q$ too large **then**

- Divide $\mu.Q$ into four parts and put four leaves in $\mathcal{T}$;
- add new leaves to $L$;

**if** $\mu.Q$ now has neighbors that are too large **then** add it to $L$;

**return** $\mathcal{T}$

**Thm 3:** Let $\mathcal{T}$ be a quadtree with $m$ nodes and depth $d$. The balanced version $\mathcal{T}_B$ of $\mathcal{T}$ has $O(m)$ nodes and can be constructed in $O((d + 1)m)$ time.
Exercise 3

**Balanced Quadtrees**

Change: All adjacent rectangles have the same size.
Exercise 3

**Balanced Quadtrees**

Change: All adjacent rectangles have the same size.

**Question:** How many leaves?
Problem: Range Query

Given point set $P$ and rectangle $R$, find all points of $P$ that lie in $R$.
Exercise 4

Quadtrees for Range Queries?
Range Query
Range Query

Start at root
Range Query

Start at root
→ Recursion for all four children.
Range Query

Start at root

→ Recursion for all four children.
→ South-East-Node: stop and report contained nodes.
Range Query

Start at root
→ Recursion for all four children.
→ **South-East-Node**: stop and report contained nodes.

→ **South-West-Node**: Recursion on two children.
→ **North-West-Node**: Recursion on one child.
Range Query

Start at root
→ Recursion for all four children.
→ **South-East-Node:** stop and report contained nodes.

→ **South-West-Node:** Recursion on two children.
→ **North-West-Node:** Recursion on one child.
→ When reach a leaf: return resulting node.
Range Query

Start at root
→ Recursion for all four children.
→ **South-East-Node**: stop and report contained nodes.

→ **South-West-Node**: Recursion on two children.
→ **North-West-Node**: Recursion on one child.
→ When reach a leaf: return resulting node.

Fast implementation?
Exercise 4(b)

Given:
- rooted tree $T$
- at most one point in each leaf.
- one node $v$ in $T$

Find: All points stored in the subtree of $v$
Exercise 4(b)

Given:
- rooted tree $T$
- at most one point in each leaf.
- one node $v$ in $T$

Find: All points stored in the subtree of $v$

Procedure
traverse subtree $T_v$

Time $O(|T_v|)$
Storage $\times$
Preprocessing $\times$
Exercise 4(b)

Given:
- rooted tree $T$
- at most one point in each leaf.
- one node $v$ in $T$

Find: All points stored in the subtree of $v$

Procedure
- traverse subtree $T_v$
- store all successive points

<table>
<thead>
<tr>
<th>Time</th>
<th>Storage</th>
<th>Preprocessing</th>
</tr>
</thead>
<tbody>
<tr>
<td>$O(</td>
<td>T_v</td>
<td>)$</td>
</tr>
</tbody>
</table>
Exercise 4(b)

Given:
- rooted tree $T$
- at most one point in each leaf.
- one node $v$ in $T$

Find: All points stored in the subtree of $v$

<table>
<thead>
<tr>
<th>Procedure</th>
<th>Time</th>
<th>Storage</th>
<th>Preprocessing</th>
</tr>
</thead>
<tbody>
<tr>
<td>traverse subtree $T_v$</td>
<td>$O(</td>
<td>T_v</td>
<td>)$</td>
</tr>
<tr>
<td>store all successive points</td>
<td>$O(k)$</td>
<td>$O(d \cdot</td>
<td>P</td>
</tr>
</tbody>
</table>

$k =$ reported points
Exercise 4(b)

Given:
- rooted tree $T$
- at most one point in each leaf.
- one node $v$ in $T$

Find: All points stored in the subtree of $v$

Procedure

- traverse subtree $T_v$
- store all successive points
- store start and end in list of points

Time
- $O(|T_v|)$
- $O(k)$
- $O(d \cdot |P|)$

Storage
- $\times$
- $O(d \cdot |P|)$

Preprocessing
- $\times$
- $O(d \cdot |P|)$
Exercise 4(b)

Given:
- rooted tree $T$
- at most one point in each leaf.
- one node $v$ in $T$

Find: All points stored in the subtree of $v$

**Procedure**

<table>
<thead>
<tr>
<th>Time</th>
<th>Storage</th>
<th>Preprocessing</th>
</tr>
</thead>
<tbody>
<tr>
<td>$O(</td>
<td>T_v</td>
<td>)$</td>
</tr>
</tbody>
</table>

*store all successive points (k = reported points)*

<table>
<thead>
<tr>
<th>Time</th>
<th>Storage</th>
<th>Preprocessing</th>
</tr>
</thead>
<tbody>
<tr>
<td>$O(k)$</td>
<td>$O(</td>
<td>T</td>
</tr>
</tbody>
</table>

*store start and end in list of points*
Exercise 4

Quadtrees for Range Queries?

- Running time of procedure?
Exercise 4

Quadtrees for Range Queries?

- Running time of procedure?
Exercise 4

Quadtrees for Range Queries?

- Running time of procedure?
Exercise 4

Quadtrees for Range Queries?

- Running time of procedure?
Exercise 4

Quadtrees for Range Queries?

- Running time of procedure?
Exercise 4

Quadtrees for Range Queries?

- Running time of procedure?
Exercise 4

Quadtrees for Range Queries?

- Running time of procedure?
Exercise 4

- Was, wenn Anfragebereich eine durch eine vertikale Gerade begrenzte Halbebene ist?
Exercise 4

- What if query region is halfplane bounded by vertical line?