Quadtrees and Mesh Generation

1. Problem Statement
   - nonuniform mesh generation
   - the software CGAL
2. Quadtrees for Point Sets
   - definition and construction
   - complexity cost
   - balancing a quadtree subdivision
3. Mesh Generation
   - from quadtrees to meshes
   - conclusions

MCS 481 Lecture 40
Computational Geometry
Jan Verschelde, 23 April 2025

Quadtrees and Mesh Generation

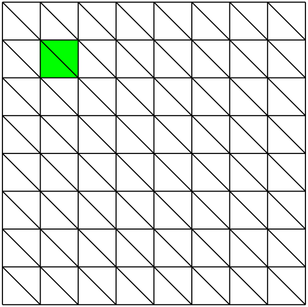
1. Problem Statement
   - nonuniform mesh generation
   - the software CGAL

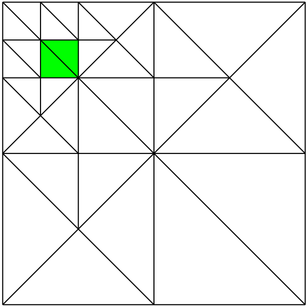
2. Quadtrees for Point Sets
   - definition and construction
   - complexity cost
   - balancing a quadtree subdivision

3. Mesh Generation
   - from quadtrees to meshes
   - conclusions

nonuniform mesh generation

At the left is a uniform mesh, at the right a nonuniform one:





If the mesh needs to be fine only at one spot (the green square), then the uniform mesh is wasteful, 128 versus 22 triangles.

problem statement

Application: solve the heat equation with finite elements methods.

Example: simulate the heat emission on a circuit board
which contains polygonal components.

Requirements on the mesh:

- *nonuniform* : finer near edges of the components, coarser away from the edges,
- *well shaped* : angles of triangles in $[\pi/4, \pi/2]$,
- *respects input* : edges of components are contained in the union of the edges of the mesh triangles,
- *conforming* : no triangle has a vertex of another triangle in the interior of one of its edges.

Quadtrees and Mesh Generation

meshes and quadtrees in CGAL

The package overview has an entire section on mesh generation.

We focus on quadtrees, provided by the Orthtree package.

Mesh generation packages wrapped for Python:

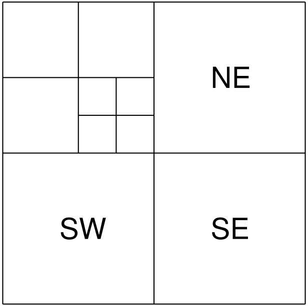
1. 2D Conforming Triangulations and Meshes
2. 3D Surface Mesh Generation
3. 3D Mesh Generation
4. Polygon Mesh Processing
5. 3D Alpha Wrapping

Quadtrees and Mesh Generation

1. Problem Statement
   - nonuniform mesh generation
   - the software CGAL
2. Quadtrees for Point Sets
   - definition and construction
   - complexity cost
   - balancing a quadtree subdivision
3. Mesh Generation
   - from quadtrees to meshes
   - conclusions

subdividing in quadrants

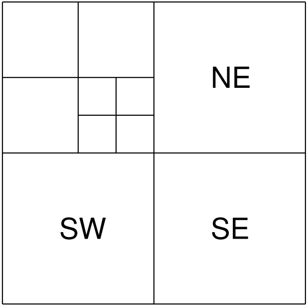
Subdivide a square region into smaller squares:

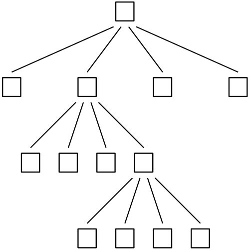


The quadrants are labeled as NW (northwest), NE (northeast), SW (southwest), and SE (Southeast).

a quadtree

A recursive data structure stores the subdivision:





recursive definition of a quadtree

Let P be a set of n points in the plane, $n > 1$. Compute

$$x_{\rm mid} = \frac{1}{2}\left(\min_{p \in P} p_x + \max_{p \in P} p_x\right), \quad y_{\rm mid} = \frac{1}{2}\left(\min_{p \in P} p_y + \max_{p \in P} p_y\right),$$

$$\begin{aligned}
P_{\rm NE} &= \{\, p \in P \mid p_x > x_{\rm mid},\ p_y > y_{\rm mid} \,\}, \\
P_{\rm NW} &= \{\, p \in P \mid p_x \leq x_{\rm mid},\ p_y > y_{\rm mid} \,\}, \\
P_{\rm SE} &= \{\, p \in P \mid p_x > x_{\rm mid},\ p_y \leq y_{\rm mid} \,\}, \\
P_{\rm SW} &= \{\, p \in P \mid p_x \leq x_{\rm mid},\ p_y \leq y_{\rm mid} \,\}.
\end{aligned}$$

Definition (recursive definition of a quadtree)

For P, a set of n points in the plane, *the quadtree for P* is

- P if $n \leq 1$, or otherwise
- a tree with as children the quadtrees for $P_{\rm NE}$, $P_{\rm NW}$, $P_{\rm SE}$, and $P_{\rm SW}$.

construction of a quadtree

Algorithm CONSTRUCTQUADTREE(P)

Input: P, a set of points in the plane.
Output: the root of a quadtree which stores P.

1. if $\#P \leq 1$ then
2. return LEAF(P)

   else
3. $x_{\rm mid} = \frac{1}{2}\left(\min_{p \in P} p_x + \max_{p \in P} p_x\right), y_{\rm mid} = \frac{1}{2}\left(\min_{p \in P} p_y + \max_{p \in P} p_y\right)$
4. $P_{\rm NE} = \{ p \in P \mid p_x > x_{\rm mid}, \ p_y > y_{\rm mid} \}$
5. CHILDNE = CONSTRUCTQUADTREE($P_{\rm NE}$)

recursive construction of children continued

6. $P_{\mathrm{NW}} = \{ \, p \in P \mid p_x \leq x_{\mathrm{mid}}, \; p_y > y_{\mathrm{mid}} \, \}$
7. CHILDNW = CONSTRUCTQUADTREE(P_{NW})
8. $P_{\mathrm{SE}} = \{ \, p \in P \mid p_x > x_{\mathrm{mid}}, \; p_y \leq y_{\mathrm{mid}} \, \}$
9. CHILDSE = CONSTRUCTQUADTREE(P_{SE})
10. $P_{\mathrm{SW}} = \{ \, p \in P \mid p_x \leq x_{\mathrm{mid}}, \; p_y \leq y_{\mathrm{mid}} \, \}$
11. CHILDSW = CONSTRUCTQUADTREE(P_{SW})
12. return NODE(CHILDNE, CHILDNW, CHILDSE, CHILDSW)

Quadtrees and Mesh Generation

1. Problem Statement
   - nonuniform mesh generation
   - the software CGAL

2. Quadtrees for Point Sets
   - definition and construction
   - complexity cost
   - balancing a quadtree subdivision

3. Mesh Generation
   - from quadtrees to meshes
   - conclusions

depth and distance

The depth of the quadtree is related to

- the distance between the points, and
- the size of the initial square

Lemma

Let P be a set of points and consider

- $s = \max_{p,q \in P} (|p_x - q_x|, |p_y - q_y|)$
  is the length of the side of the square that contains P,
- $c = \min_{p,q \in P, p \neq q} \sqrt{(p_x - q_x)^2 + (p_y - q_y)^2}$
  is the smallest distance between any two points of P,
- *d is the depth of the quadtree for P,*

then $d \leq \log_2 \left(\frac{s}{c} \right) + \frac{3}{2}$.

three steps in the proof of the lemma

1. Every time we go down one level in the tree, the length s of side of the square is divided by 2. At depth i, the length of the side of the square is $1/2^i$.
2. The maximum distance between 2 points in a square equals the diagonal length of a square, which is $s\sqrt{2}/2^i$ at depth i.

   To relate s to c, the distance between 2 points in P:

   $$\begin{aligned} s\sqrt{2}/2^i \geq c \quad &\Rightarrow \quad s\sqrt{2} \geq c2^i \Rightarrow \frac{s}{c}\sqrt{2} \geq 2^i \\ &\Rightarrow \quad \log_2\left(\frac{s}{c}\sqrt{2}\right) \geq \log_2(2^i) = i. \end{aligned}$$

   This leads to $i \leq \log_2(s/c) + 1/2$, as $\log_2(\sqrt{2}) = 1/2$.
3. The depth of the quadtree is exactly the maximal depth $+1$.

complexity and cost

Theorem (complexity of a quadtree)

A quadtree of depth d storing n points has $O((d+1)n)$ nodes and takes $O((d+1)n)$ operations.

The theorem states the following.

1. The size of the quadtree is $O((d+1)n)$.
2. The running time is $O((d+1)n)$.

The running time follows from the size.

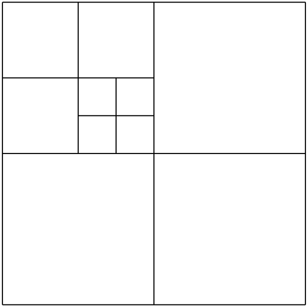
Exercise 1: Because a quadtree is simpler than a kd tree,
a quadtree can be constructed faster than a kd tree.

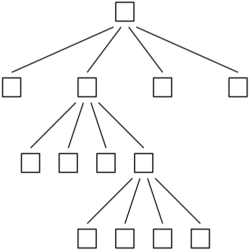
Use CGAL on a sufficiently large collection of points to verify that a quadtree can be constructed faster than a kd tree.

Quadtrees and Mesh Generation

1. Problem Statement
   - nonuniform mesh generation
   - the software CGAL
2. Quadtrees for Point Sets
   - definition and construction
   - complexity cost
   - balancing a quadtree subdivision
3. Mesh Generation
   - from quadtrees to meshes
   - conclusions

an unbalanced quadtree





Definition (balanced quadtree)

A quadtree subdivision is *balanced* if any two neighboring squares differ at most a factor two in size.

In a balanced quadtree subdivision,
any two neighboring leaves differ at most one in depth.

from the CGAL documentation

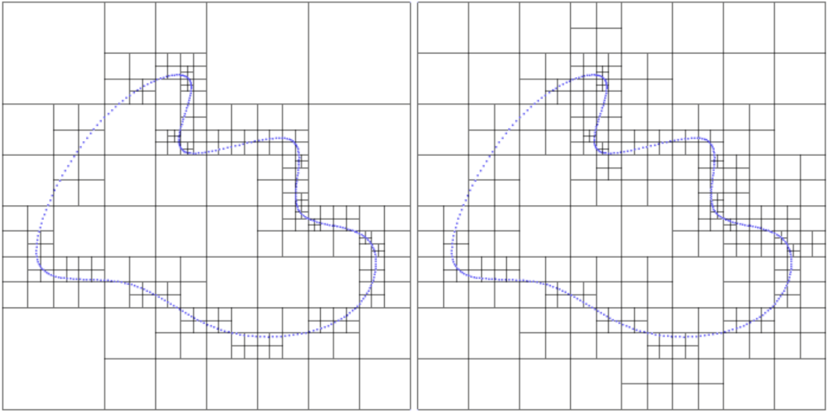
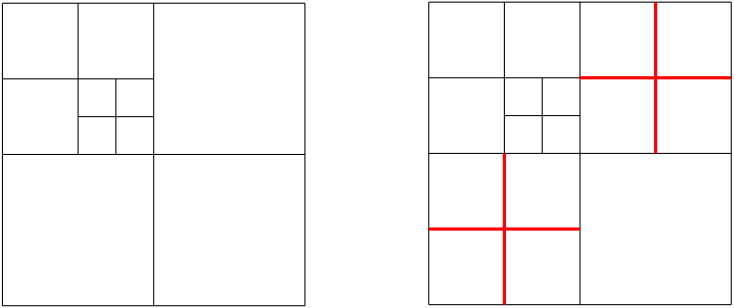


Figure 93.3 Quadtree before and after being graded.

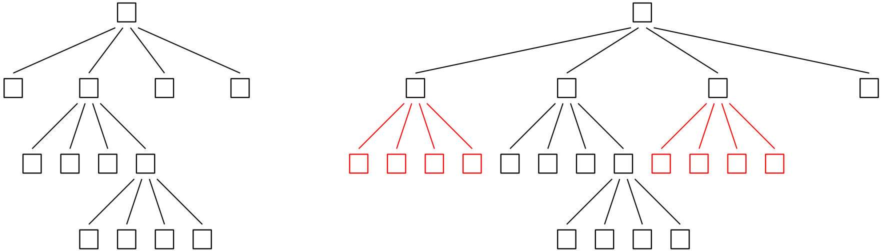
An quadtree is *graded* if the difference of depth between two adjacent leaves is at most one for every pair of leaves.

balancing the subdivision



The SE sector of the NW sector has squares with differ with a factor four in size, compared to the adjacent NE and SW sectors.

balancing the quadtree



The NE and SW sectors of the quadtree are refined.

finding neighboring squares

In the balancing of a quadtree,
we have to decide if a square has a neighbor at the same depth.

The algorithm of computing a neighbor is recursive:

- To find a neighbor at the same level,
  we go up to the siblings of the parent node.
- If there are no sibling nodes, then we go to the parent.

Theorem (cost of finding a neighbor)

Let T be a quadtree of depth d.
A neighbor of a node can be computed in $O(d+1)$ time.

In the extreme case when the tree is very unbalanced,
we may have to go all the way up to the root of the tree.

a second comparison with kd trees

The simplicity of a quadtree relative to a kd tree has consequences.

Exercise 2: From the CGAL documentation:

A kd tree is expected to outperform the finding of the nearest neighbor.

- Either verify this experimentally with timings; or
- provide arguments to justify this statement.

cost of balancing a quadtree

As we add more nodes when balancing, how does the tree grow?

Theorem (cost of balancing a quadtree)

Let T be a quadtree with m nodes. Balancing T leads to a quadtree with $O(m)$ nodes and takes $O((d+1)m)$ operations.

For n points, the size of a quadtree of depth d is $O((d+1)n)$.

As before, the theorem constains two statements:

1. The balanced quadtree has $O(m)$ nodes.
2. The running time of balancing is $O((d+1)m)$.

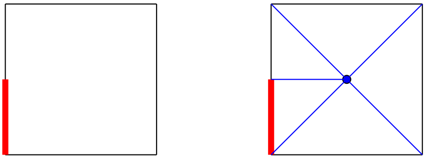
The running time follows from the size of the balanced tree.

Quadtrees and Mesh Generation

1. Problem Statement
   - nonuniform mesh generation
   - the software CGAL
2. Quadtrees for Point Sets
   - definition and construction
   - complexity cost
   - balancing a quadtree subdivision
3. Mesh Generation
   - from quadtrees to meshes
   - conclusions

Steiner points

If an edge of a component intersects a square at a side,
an extra vertex in the square is needed to get a well shaped mesh.



Such extra vertex is called a *Steiner point*.

mesh generation

The algorithm to generate a mesh has three stages.

1. Make a quadtree T till the depth is such that
   - the side length is still larger than the unit size, and
   - the sides of the square intersect the boundary of components.
2. Make T balanced.
3. Make a triangulation, adding Steiner points when needed.

Theorem (cost of the mesh generation)

Let U be the length of the side of the square at the start and let $p(S)$ be the sum of the perimeters of all components. A nonuniform triangular mesh has size $O(p(S) \log(U))$ and takes time $O(p(S) \log^2(U))$ to construct.

Quadtrees and Mesh Generation

1. Problem Statement
   - nonuniform mesh generation
   - the software CGAL
2. Quadtrees for Point Sets
   - definition and construction
   - complexity cost
   - balancing a quadtree subdivision
3. Mesh Generation
   - from quadtrees to meshes
   - conclusions

conclusions

We considered triangulations before:

- triangulating y-monotone polygons, and
- Delaunay triangulations.

But with previously considered triangulation algorithms,
we cannot generate nonuniform meshes.

Let P be a set of n points.

- The size of the quadtree T of depth d for P and the running time to compute T are both $O((d+1)n)$.
- The cost to balance T is linear in the size m of T and balancing takes running time $O((d+1)m)$.

The cost to generate a mesh is $O(p(S)\log^2(U))$ where

- U is the length of the side of the square at the start, and
- $p(S)$ is the sum of the perimeters of all components.