Algorithms and datastructures II

Lecture 12: Geometric algorithms

Jan Hubička

Department of Applied Mathematics
Charles University
Prague

Dec 20 2020

Convex hull problem

Convex hull

Given vertex of n points in $\mathbb{R}^2$ we want to find shortest closed curve such that its interior contains all points.

Convex hull problem

Convex hull

Given vertex of n points in $\mathbb{R}^2$ we want to find shortest closed curve such that its interior contains all points.

1. We are going to represent the convex hull by a sequence of points on its boundary sorted either clockwise or counter-clockwise.
2. For simplicity we assume that points have different x coordinates.
3. We will use technique called Plane sweep.

Convex hull problem

ConvexHull ($a_1, \ldots, a_n$)

1. Let $b_1, \ldots b_n$ be points $a_1, \ldots, a_n$ sorted by x coordinate
2. Put b_1 into upper and lower envelopes: $U \leftarrow L \leftarrow (b_1)$
3. For $b = b_2, \ldots b_n$:
4. Recompute upper envelope:
5. While $|U| \geq 2$, $U = (\ldots u_{k-1}, u_k)$ and angle $u_{k-1} u_k b$ is oriented to left:
6. Remove u_k from U.
7. Add b to the end of U
8. Handle lower envelope symmetrically
9. Convex hull consists of H and D.

Convex hull problem

ConvexHull ($a_1, \ldots, a_n$)

1. Let $b_1, \ldots b_n$ be points $a_1, \ldots, a_n$ sorted by x coordinate
2. Put b_1 into upper and lower envelopes: $U \leftarrow L \leftarrow (b_1)$
3. For $b = b_2, \ldots b_n$:
4. Recompute upper envelope:
5. While $|U| \geq 2$, $U = (\ldots u_{k-1}, u_k)$ and angle $u_{k-1}u_k b$ is oriented to left:
6. Remove u_k from U.
7. Add b to the end of U
8. Handle lower envelope symmetrically
9. Convex hull consists of H and D.

Time complexity: $O(n \log n)$

Convex hull problem

ConvexHull ($a_1, \ldots, a_n$)

1. Let $b_1, \ldots b_n$ be points $a_1, \ldots, a_n$ sorted by x coordinate
2. Put b_1 into upper and lower envelopes: $U \leftarrow L \leftarrow (b_1)$
3. For $b = b_2, \ldots b_n$:
4. Recompute upper envelope:
5. While $|U| \geq 2$, $U = (\ldots u_{k-1}, u_k)$ and angle $u_{k-1}u_k b$ is oriented to left:
6. Remove u_k from U.
7. Add b to the end of U
8. Handle lower envelope symmetrically
9. Convex hull consists of H and D.

Time complexity: $O(n \log n)$

Determining orientation: Let $\vec{u} = (x_1, y_1)$ and $\vec{v} = (x_2, y_2)$ be row vectors. Let

$$M = \begin{pmatrix} \vec{u} \\ \vec{v} \end{pmatrix} = \begin{pmatrix} x_1 & y_1 \\ x_2 & y_2 \end{pmatrix}$$

If $\det M = x_1 y_2 - x_2 y_1$ is non-negative then the angle is oriented to left.

Convex hull problem

ConvexHull ($a_1, \ldots, a_n$)

1. Let $b_1, \ldots b_n$ be points $a_1, \ldots, a_n$ sorted by x coordinate
2. Put b_1 into upper and lower envelopes: $U \leftarrow L \leftarrow (b_1)$
3. For $b = b_2, \ldots b_n$:
4. Recompute upper envelope:
5. While $|U| \geq 2$, $U = (\ldots u_{k-1}, u_k)$ and angle $u_{k-1}u_k b$ is oriented to left:
6. Remove u_k from U.
7. Add b to the end of U
8. Handle lower envelope symmetrically
9. Convex hull consists of H and D.

Time complexity: $O(n \log n)$

Observation

Problem of sorting real numbers can be reduced to convex hull problem.

intersections of line segments

Intersection of line segments

Given line segments $\ell_1, \ldots \ell_n$ determine all their mutual intersections

For simplicity we will assume that lines are in general position:

1. no 3 line segments intersect in one point,
2. intersection of any two disjoint lines is at most one point, and
3. every line has different starting and ending y-coordinate.

intersections of line segments

Intersection of line segments

Given line segments $\ell_1, \ldots \ell_n$ determine all their mutual intersections

For simplicity we will assume that lines are in general position:

1. no 3 line segments intersect in one point,
2. intersection of any two disjoint lines is at most one point, and
3. every line has different starting and ending y-coordinate.

intersections of line segments

Intersection of line segments

Given line segments $\ell_1, \ldots \ell_n$ determine all their mutual intersections

For simplicity we will assume that lines are in general position:

1. no 3 line segments intersect in one point,
2. intersection of any two disjoint lines is at most one point, and
3. every line has different starting and ending y-coordinate.

Intersections ($\ell_1, \ldots \ell_n$)

1. Initialize I (intersections with the moving line) to $\emptyset$.
2. Add to calendar C all initial points of all lines.
3. Until C is not empty:
4. Remove first event.
5. If event was beginning of a line: add new line to I.
6. If event was the end of line: remove it from I.
7. If event was intersection: report it and exchange lines in I.
8. Update all intersection events (at most 2 are added or removed)

intersections of line segments

Intersections ($\ell_1, \ldots \ell_n$)

1. Initialize I (intersections with the moving line) to $\emptyset$.
2. Add to calendar C all initial points of all lines.
3. Until C is not empty:
4. Remove first event.
5. If event was beginning of a line: add new line to I.
6. If event was the end of line: remove it from I.
7. If event was intersection: report it and exchange lines in I.
8. Update all intersection events (at most 2 are added or removed)

Representation of calendar: we need operations Insert, ExtractMin, Remove.

intersections of line segments

Intersections ($\ell_1, \ldots \ell_n$)

1. Initialize I (intersections with the moving line) to $\emptyset$.
2. Add to calendar C all initial points of all lines.
3. Until C is not empty:
4. Remove first event.
5. If event was beginning of a line: add new line to I.
6. If event was the end of line: remove it from I.
7. If event was intersection: report it and exchange lines in I.
8. Update all intersection events (at most 2 are added or removed)

Representation of calendar: we need operations Insert, ExtractMin, Remove.
There are at most $3n$ events in calendar. Binary tree will work in $O(\log n)$ for operation

Representation of the intersection: we need operations Insert, Remove, Next and Previous

intersections of line segments

Intersections ($\ell_1, \ldots \ell_n$)

1. Initialize I (intersections with the moving line) to $\emptyset$.
2. Add to calendar C all initial points of all lines.
3. Until C is not empty:
4. Remove first event.
5. If event was beginning of a line: add new line to I.
6. If event was the end of line: remove it from I.
7. If event was intersection: report it and exchange lines in I.
8. Update all intersection events (at most 2 are added or removed)

Representation of calendar: we need operations Insert, ExtractMin, Remove.
There are at most $3n$ events in calendar. Binary tree will work in $O(\log n)$ for operation

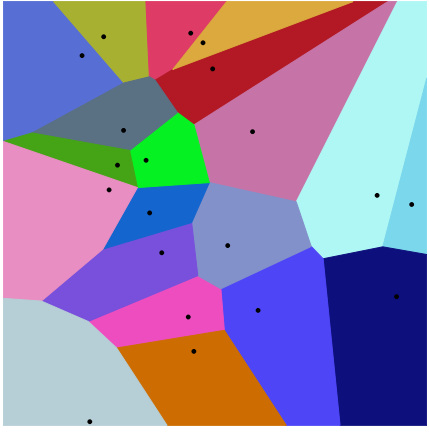
Representation of the intersection: we need operations Insert, Remove, Next and Previous
We can again use binary tree. x coordinates are not good as keys since they keep changing, however we can keep references to lines since relative order does not change from step to step.

Overall there are $O(n + i)$ events where i is number of intersections and runtime is $O((n + i) \log n)$

Vornoi diagram

Definition (Vornoi diagram)

Vornoi diagram for a given set of locations $x_1, \ldots x_n \in \mathbb{R}^2$ is a system of areas $B_1, \ldots, B_n \subseteq R^2$ such that B_i consists of all points $p \in \mathbb{R}^2$ with the property that the distance from p to x_i is smaller or equal to the distance from p to any other location.



Vornoi diagram

Definition (Vornoi diagram)

Vornoi diagram for a given set of locations $x_1, \ldots x_n \in \mathbb{R}^2$ is a system of areas $B_1, \ldots, B_n \subseteq R^2$ such that B_i consists of all points $p \in \mathbb{R}^2$ with the property that the distance from p to x_i is smaller or equal to the distance from p to any other location.

Vornoi diagram

Definition (Vornoi diagram)

Vornoi diagram for a given set of locations $x_1, \ldots x_n \in \mathbb{R}^2$ is a system of areas $B_1, \ldots, B_n \subseteq R^2$ such that B_i consists of all points $p \in \mathbb{R}^2$ with the property that the distance from p to x_i is smaller or equal to the distance from p to any other location.

Lemma

Vornoi diagram has linear combinatorial complexity. For n locations it consists of $O(n)$ vertices and faces.

Fact

Let G be a connected planar graph with no edge multiplicities. Let $v \geq 3$ be number of vertices, e number of edges and f number of faces. Then:

1. $e \leq 3v - 6$
2. $v + f = e + 2$ (Euler formula)

Vornoi diagram

Definition (Vornoi diagram)

Vornoi diagram for a given set of locations $x_1, \dots x_n \in \mathbb{R}^2$ is a system of areas $B_1, \dots, B_n \subseteq R^2$ such that B_i consists of all points $p \in \mathbb{R}^2$ with the property that the distance from p to x_i is smaller or equal to the distance from p to any other location.

Lemma

Vornoi diagram has linear combinatorial complexity. For n locations it consists of $O(n)$ vertices and faces.

Fact

Let G be a connected planar graph with no edge multiplicities. Let $v \geq 3$ be number of vertices, e number of edges and f number of faces. Then:

1. $e \leq 3v - 6$
2. $v + f = e + 2$ (Euler formula)

Proof (of lemma).

Vornoi diagram for n locations has n areas so it leads to $f = n + 1$ faces.

Vornoi diagram

Definition (Vornoi diagram)

Vornoi diagram for a given set of locations $x_1, \dots x_n \in \mathbb{R}^2$ is a system of areas $B_1, \dots, B_n \subseteq R^2$ such that B_i consists of all points $p \in \mathbb{R}^2$ with the property that the distance from p to x_i is smaller or equal to the distance from p to any other location.

Lemma

Vornoi diagram has linear combinatorial complexity. For n locations it consists of $O(n)$ vertices and faces.

Fact

Let G be a connected planar graph with no edge multiplicities. Let $v \geq 3$ be number of vertices, e number of edges and f number of faces. Then:

1. $e \leq 3v - 6$
2. $v + f = e + 2$ (Euler formula)

Proof (of lemma).

Vornoi diagram for n locations has n areas so it leads to $f = n + 1$ faces.
Its dual has $v' = f$ vertices and has no edge multiplicities. Thus $e' \leq 3v' - 6$ and $e \leq 3f - 6 = 3n - 3$.

Vornoi diagram

Definition (Vornoi diagram)

Vornoi diagram for a given set of locations $x_1, \dots x_n \in \mathbb{R}^2$ is a system of areas $B_1, \dots, B_n \subseteq R^2$ such that B_i consists of all points $p \in \mathbb{R}^2$ with the property that the distance from p to x_i is smaller or equal to the distance from p to any other location.

Lemma

Vornoi diagram has linear combinatorial complexity. For n locations it consists of $O(n)$ vertices and faces.

Fact

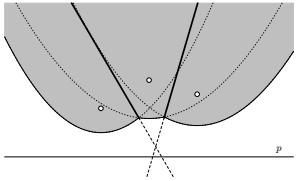
Let G be a connected planar graph with no edge multiplicities. Let $v \geq 3$ be number of vertices, e number of edges and f number of faces. Then:

1. $e \leq 3v - 6$
2. $v + f = e + 2$ (Euler formula)

Proof (of lemma).

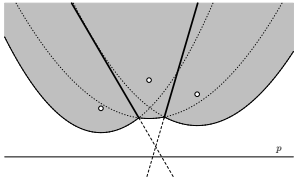
Vornoi diagram for n locations has n areas so it leads to $f = n + 1$ faces.
Its dual has $v' = f$ vertices and has no edge multiplicities. Thus $e' \leq 3v' - 6$ and $e \leq 3f - 6 = 3n - 3$.
Number of vertices is $v = e + 2 - f \leq (3n - 3) + 2 - (n + 1) = 2n - 2$. □

Fortune's algorithm



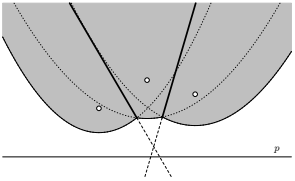
Beachline

Fortune's algorithm

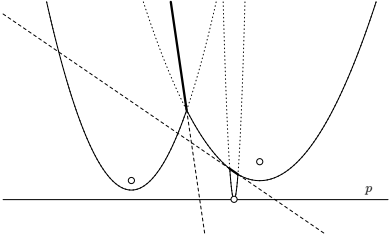


Beachline

Fortune's algorithm

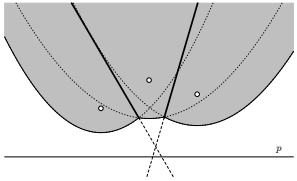


Beachline

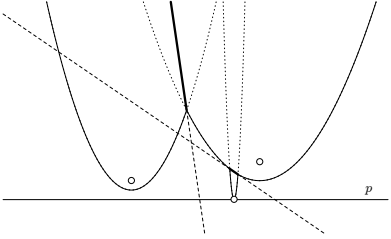


Location event

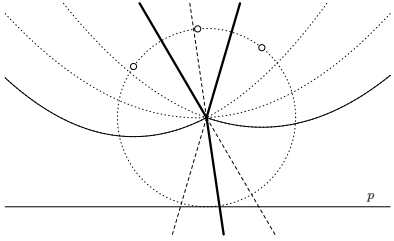
Fortune's algorithm



Beachline

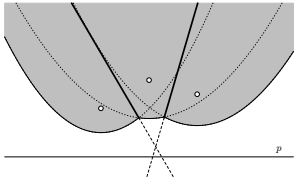


Location event

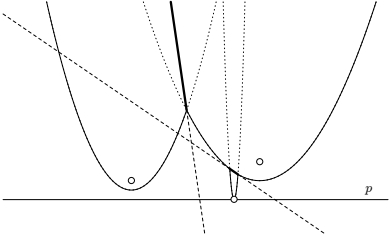


Circle event

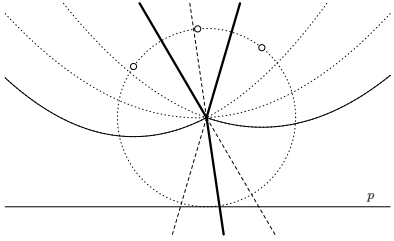
Fortune's algorithm



Beachline



Location event



Circle event

Fortune's algorithm

Fortune $(x_1, \ldots, x_n)$

1. Create empty calendar C and add all location events.
2. Create empty beachline B.
3. While C is nonempty:
4. Remove first event.
5. If it was location event:
6. Find in S parabolla corresponding to the x-coordinate.
7. Split it and insert new parabolla.
8. Add new edge to the diagram (with no enpoints).
9. If it was circle event:
10. Remove parabolla P.
11. Add a new vertex to the diagram where two edges ends and one starts.
12. Recompute beachline events $O(1)$ will be removed and $O(1)$ will be created..

Time complexity: $O(n \log n)$.


Merry Christmas!