STORING THE VORONOI DIAGRAM

Vera Sacristán

Discrete and Algorithmic Geometry
Facultat de Matemàtiques i Estadística
Universitat Politècnica de Catalunya

Storing the Voronoi diagram

Possible options, advantages and disadvantages

Storing the Voronoi diagram

Possible options, advantages and disadvantages

Storing the list of all the edges of the diagram

Storing the Voronoi diagram

Possible options, advantages and disadvantages

Storing the list of all the edges of the diagram

Advantage: small memory usage.

Disadvantage: it suffices to draw the diagram, but it does not contain the proximity information. For example, given a site p_i, finding its neighbors or reporting the vertices and edges of its Voronoi region is too expensive.

Storing the Voronoi diagram

Possible options, advantages and disadvantages

Storing the list of all the edges of the diagram

Advantage: small memory usage.

Disadvantage: it suffices to draw the diagram, but it does not contain the proximity information. For example, given a site p_i, finding its neighbors or reporting the vertices and edges of its Voronoi region is too expensive.

For each site p_i, storing the sorted list of vertices and edges of its Voronoi region, as well as the sorted list of its neighbors, etc.

Storing the Voronoi diagram

Possible options, advantages and disadvantages

Storing the list of all the edges of the diagram

Advantage: small memory usage.

Disadvantage: it suffices to draw the diagram, but it does not contain the proximity information. For example, given a site p_i, finding its neighbors or reporting the vertices and edges of its Voronoi region is too expensive.

For each site p_i, storing the sorted list of vertices and edges of its Voronoi region, as well as the sorted list of its neighbors, etc.

Advantage: allows to quickly recover neighborhood information.

Disadvantage: the stored data is redundant and it uses more space than required.

Storing the Voronoi diagram

Possible options, advantages and disadvantages

Storing the list of all the edges of the diagram

Advantage: small memory usage.

Disadvantage: it suffices to draw the diagram, but it does not contain the proximity information. For example, given a site p_i, finding its neighbors or reporting the vertices and edges of its Voronoi region is too expensive.

For each site p_i, storing the sorted list of vertices and edges of its Voronoi region, as well as the sorted list of its neighbors, etc.

Advantage: allows to quickly recover neighborhood information.

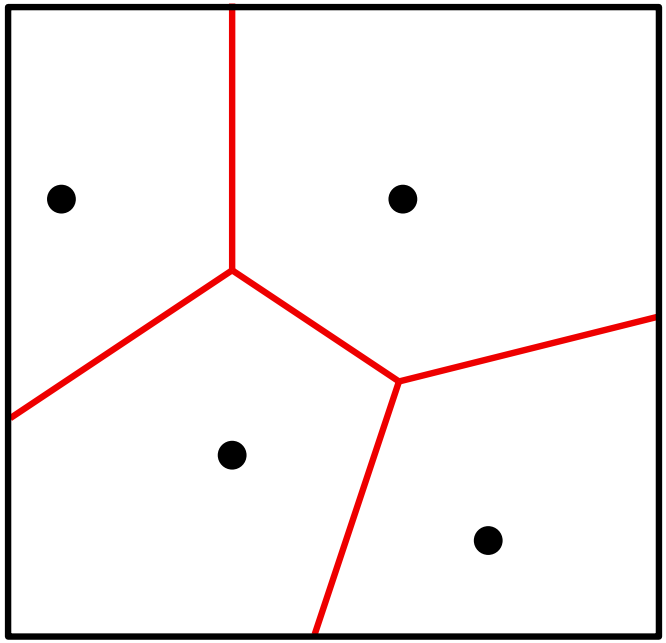
Disadvantage: the stored data is redundant and it uses more space than required.

The data structure which is most frequently used to store Voronoi diagrams is the DCEL (doubly connected edge list).

The DCEL is also used to store plane partitions, polyhedra, meshes, etc.

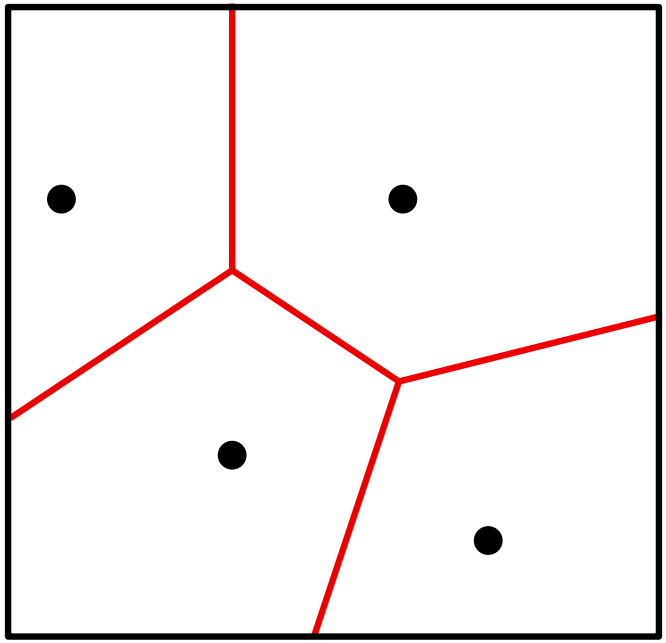
Storing the Voronoi diagram

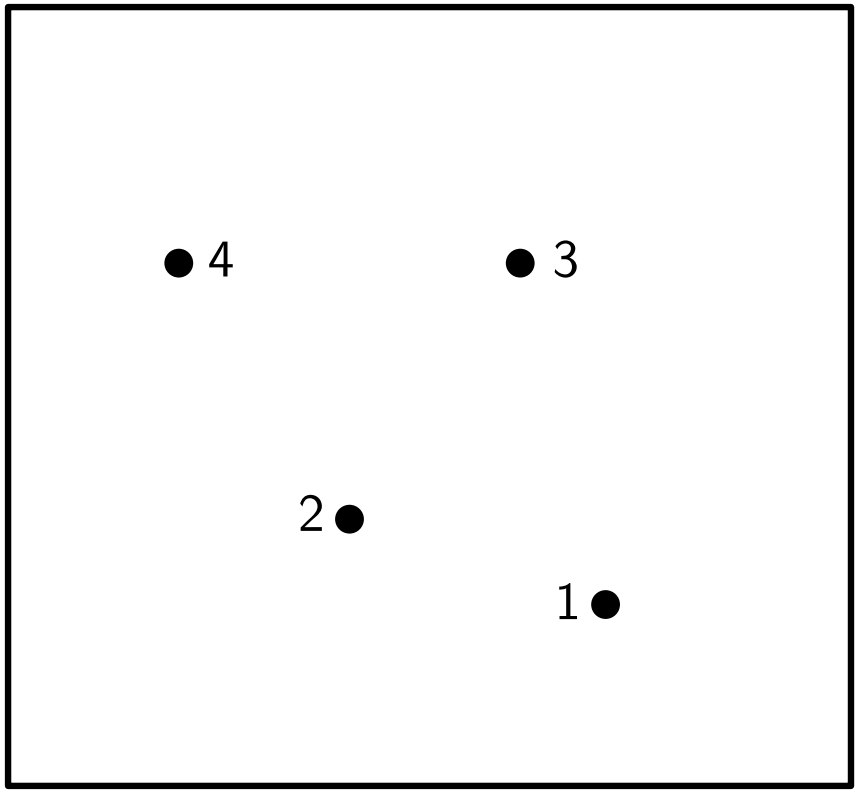
DCEL



Storing the Voronoi diagram

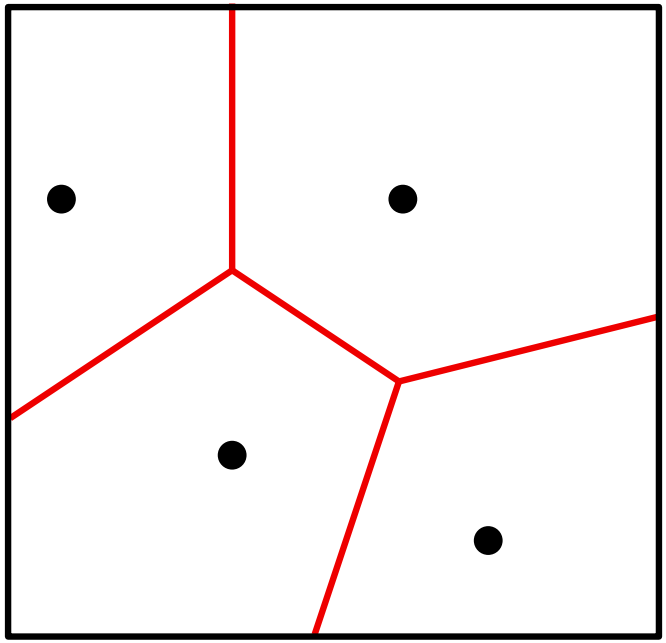
DCEL

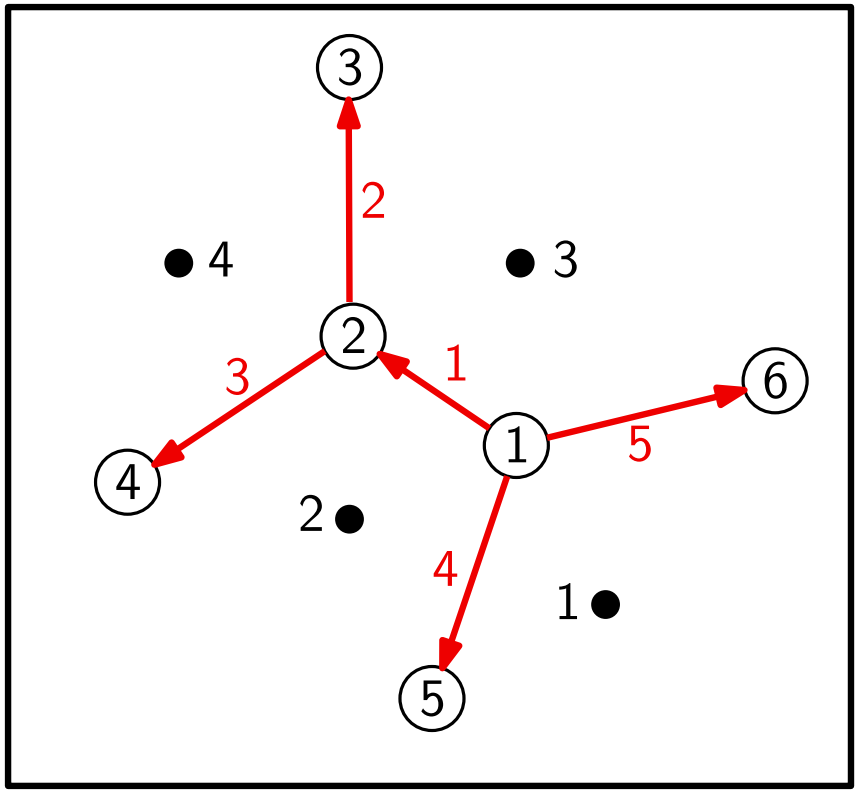




Storing the Voronoi diagram

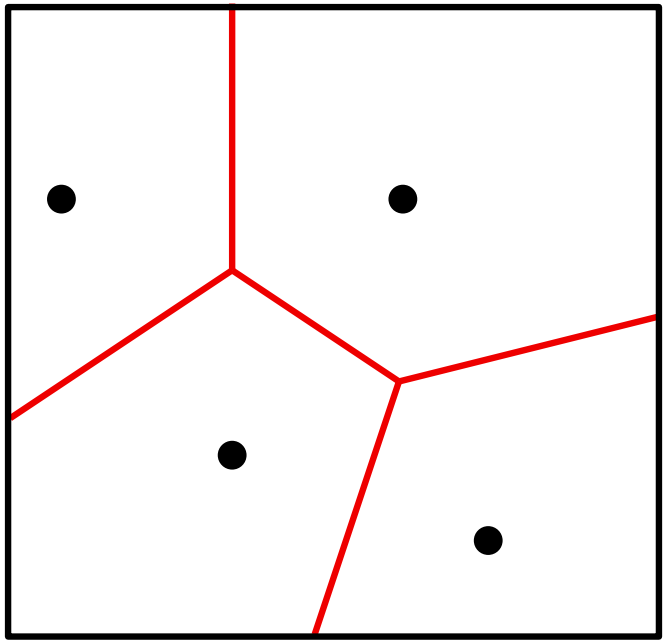
DCEL

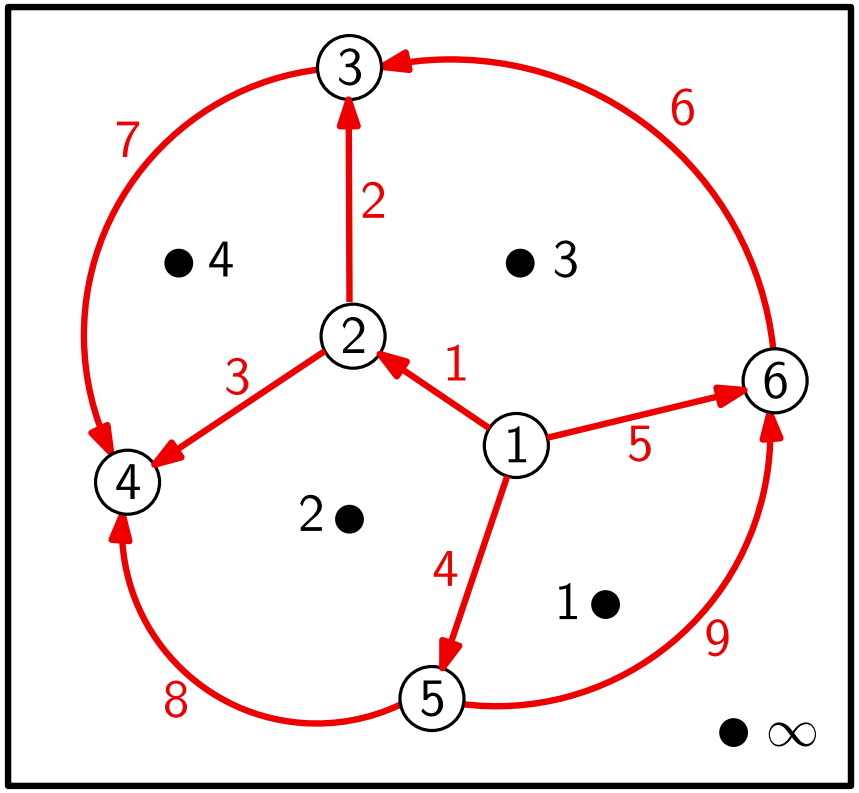




Storing the Voronoi diagram

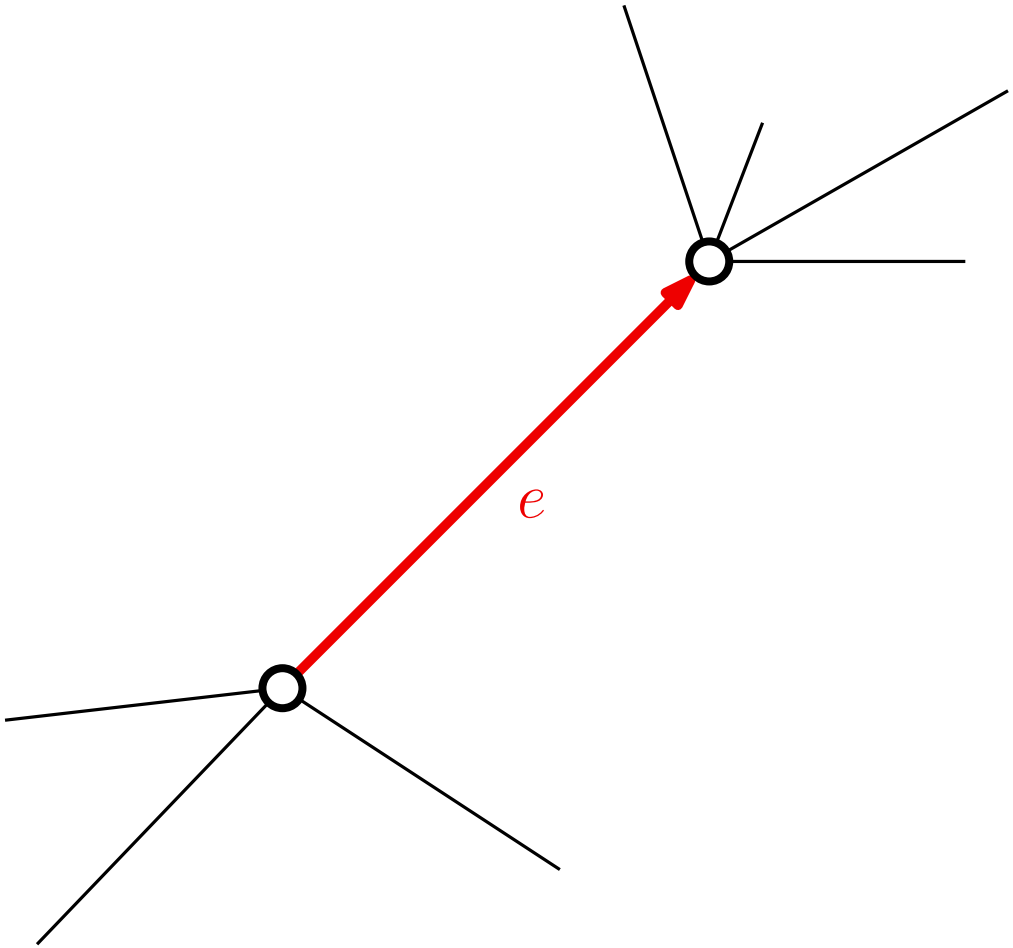
DCEL

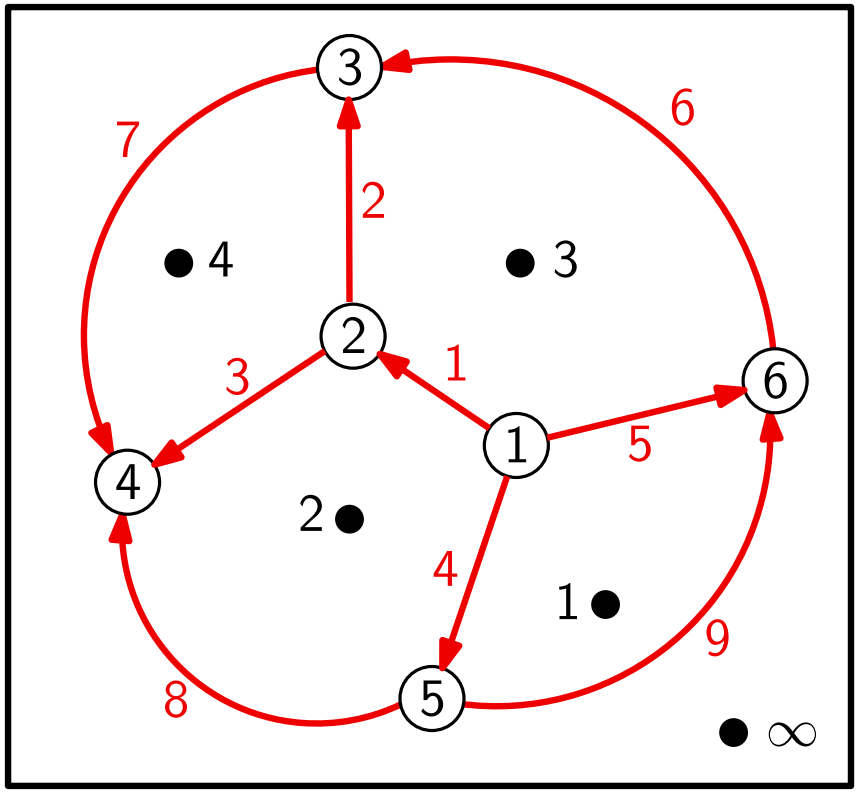




Storing the Voronoi diagram

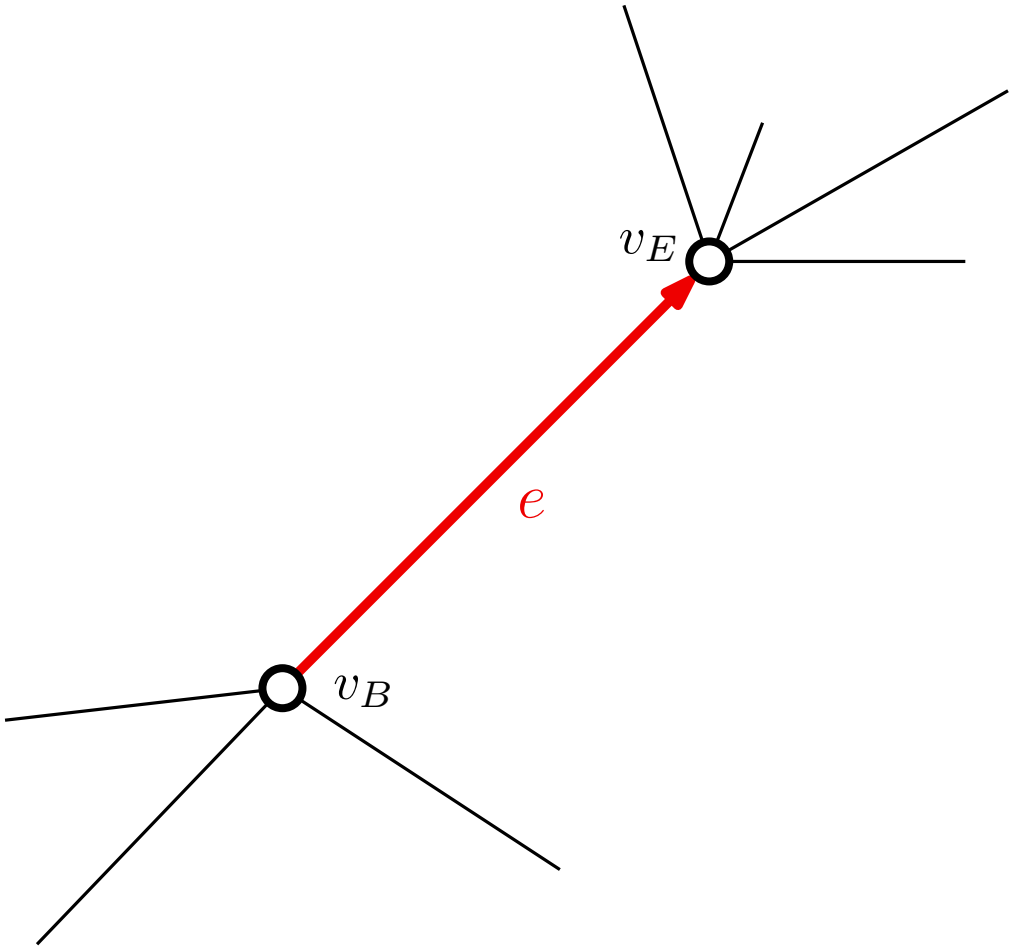
DCEL

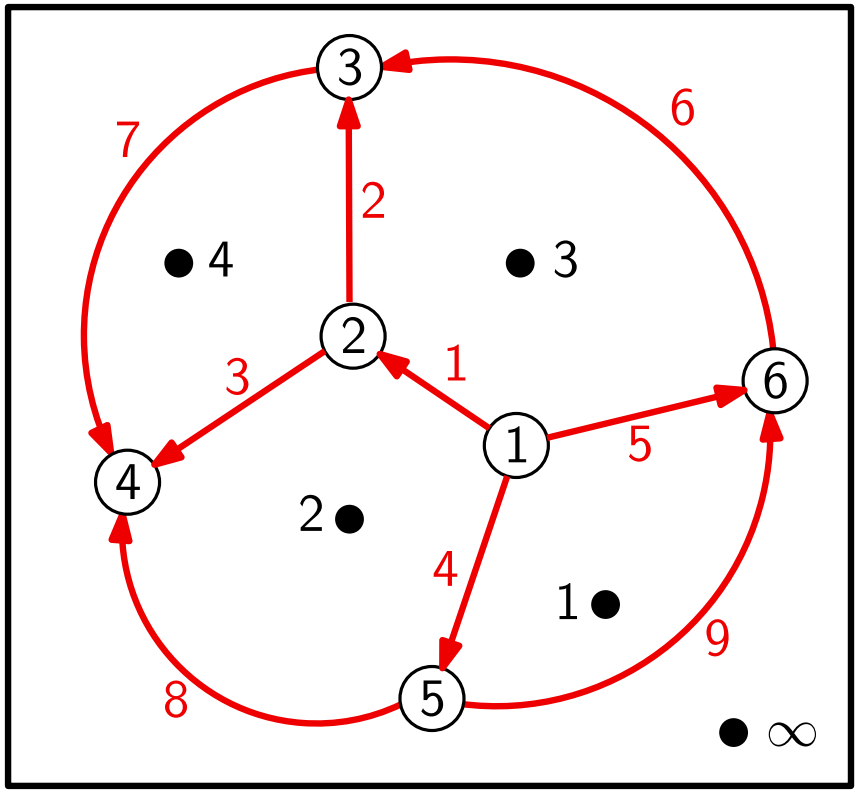




Storing the Voronoi diagram

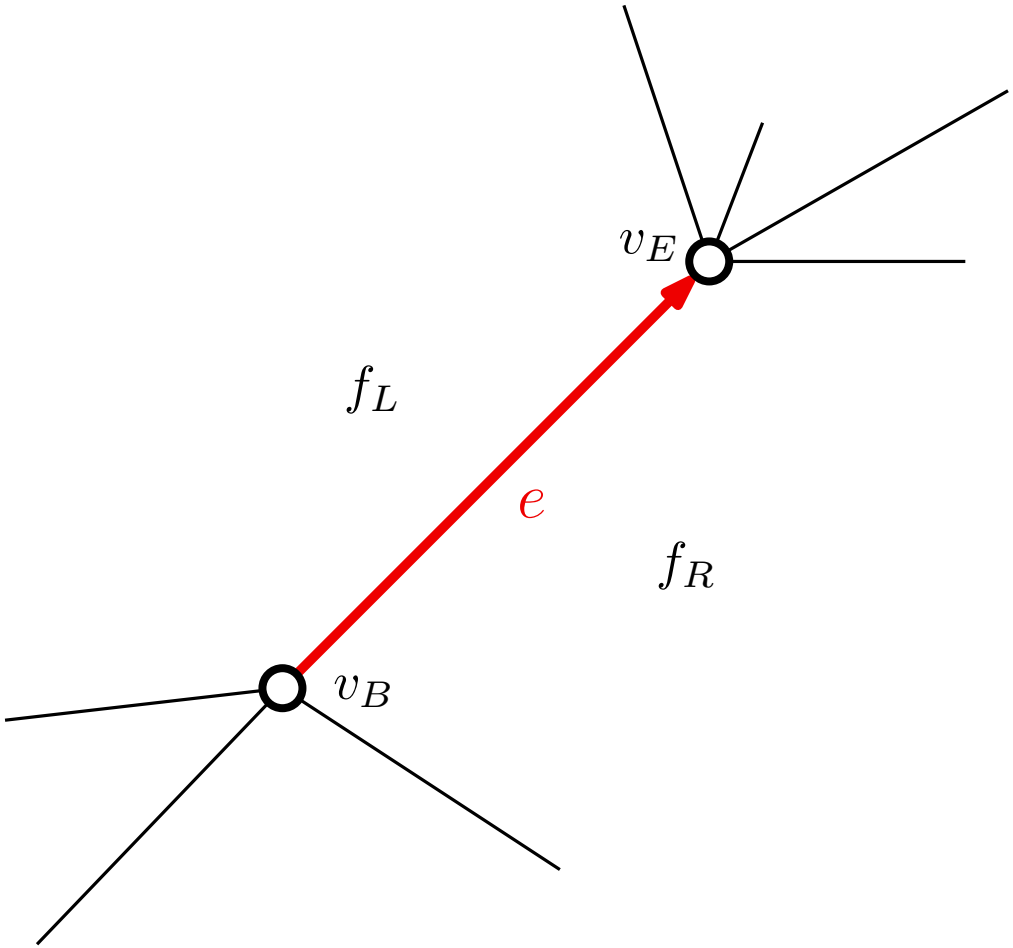
DCEL

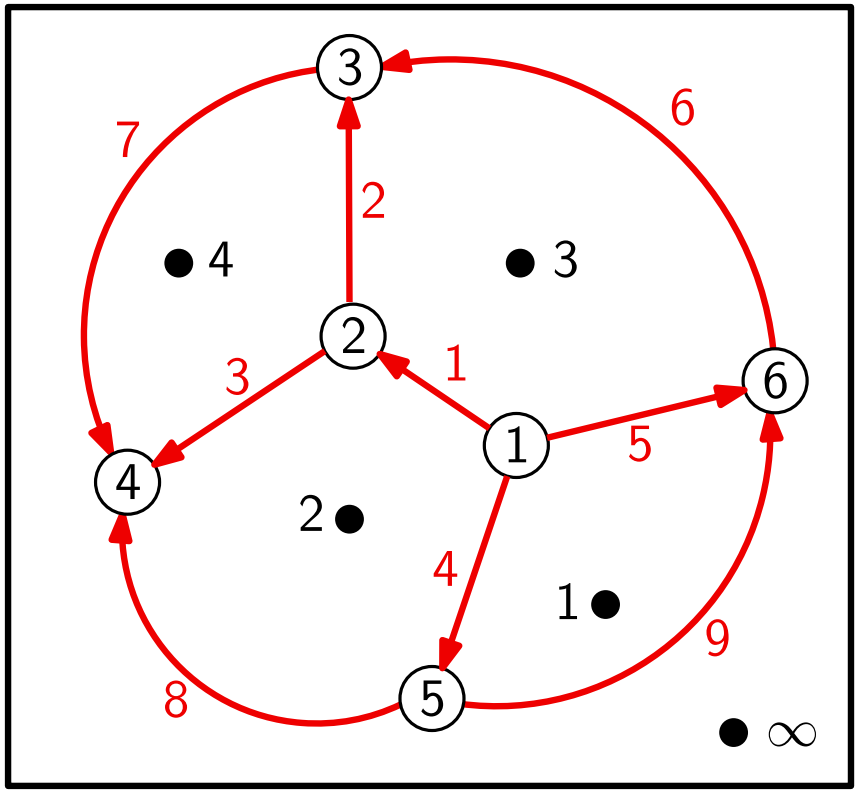




Storing the Voronoi diagram

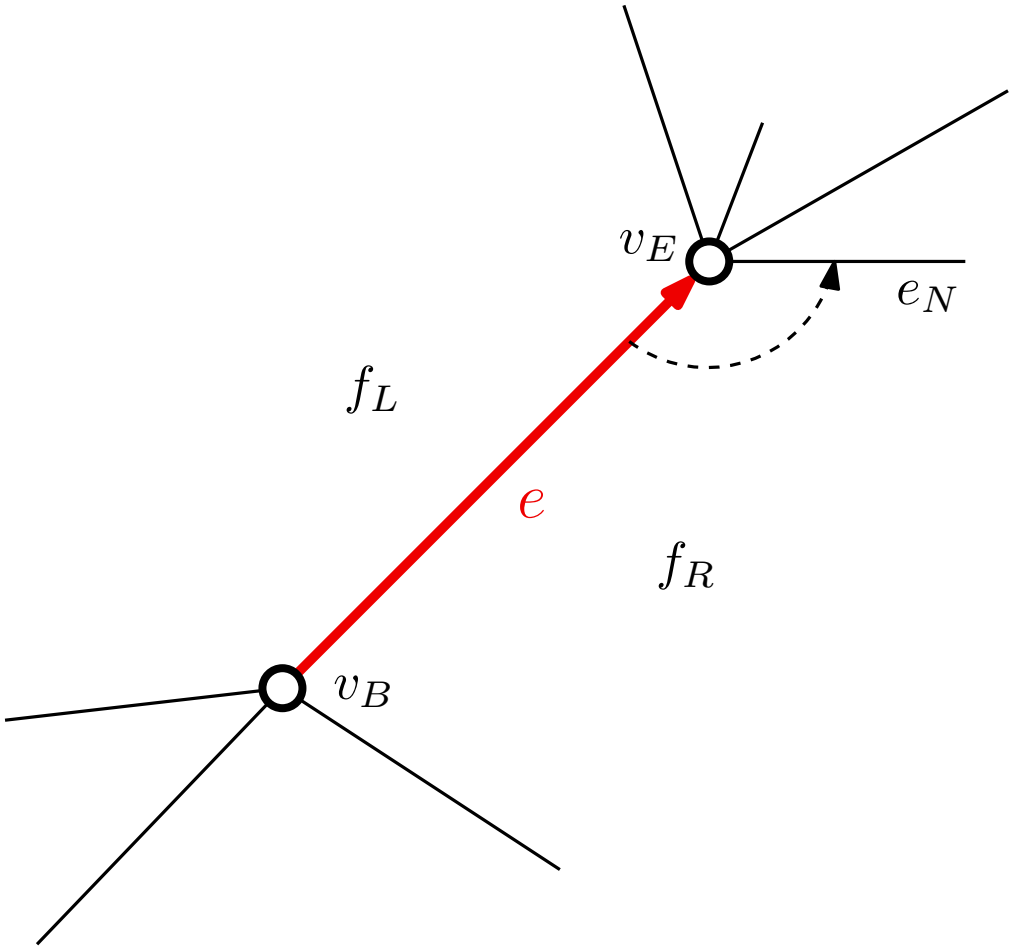
DCEL

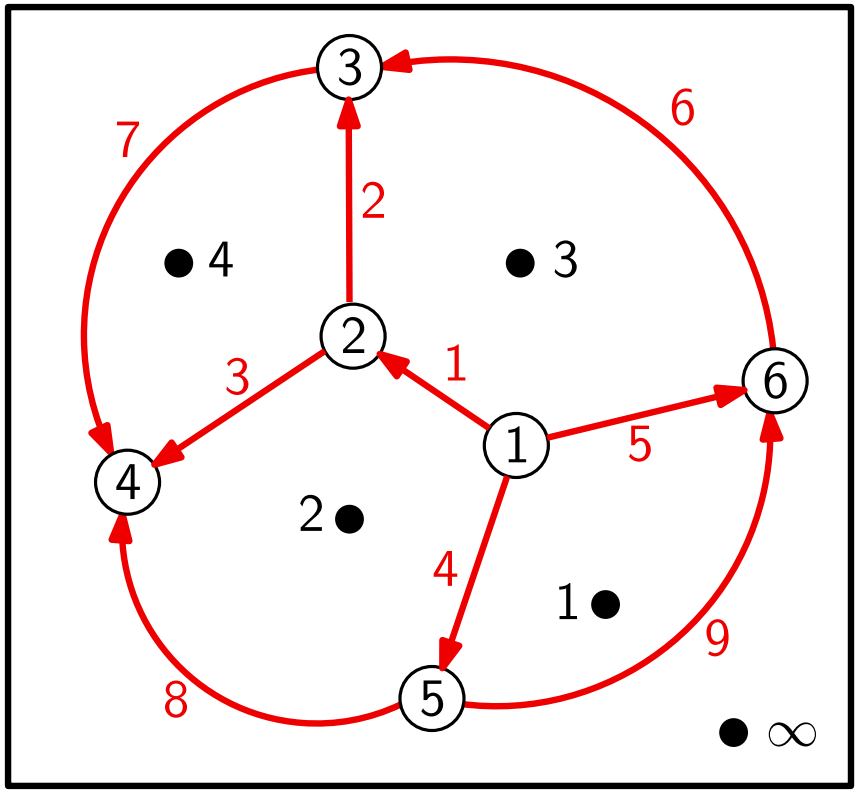




Storing the Voronoi diagram

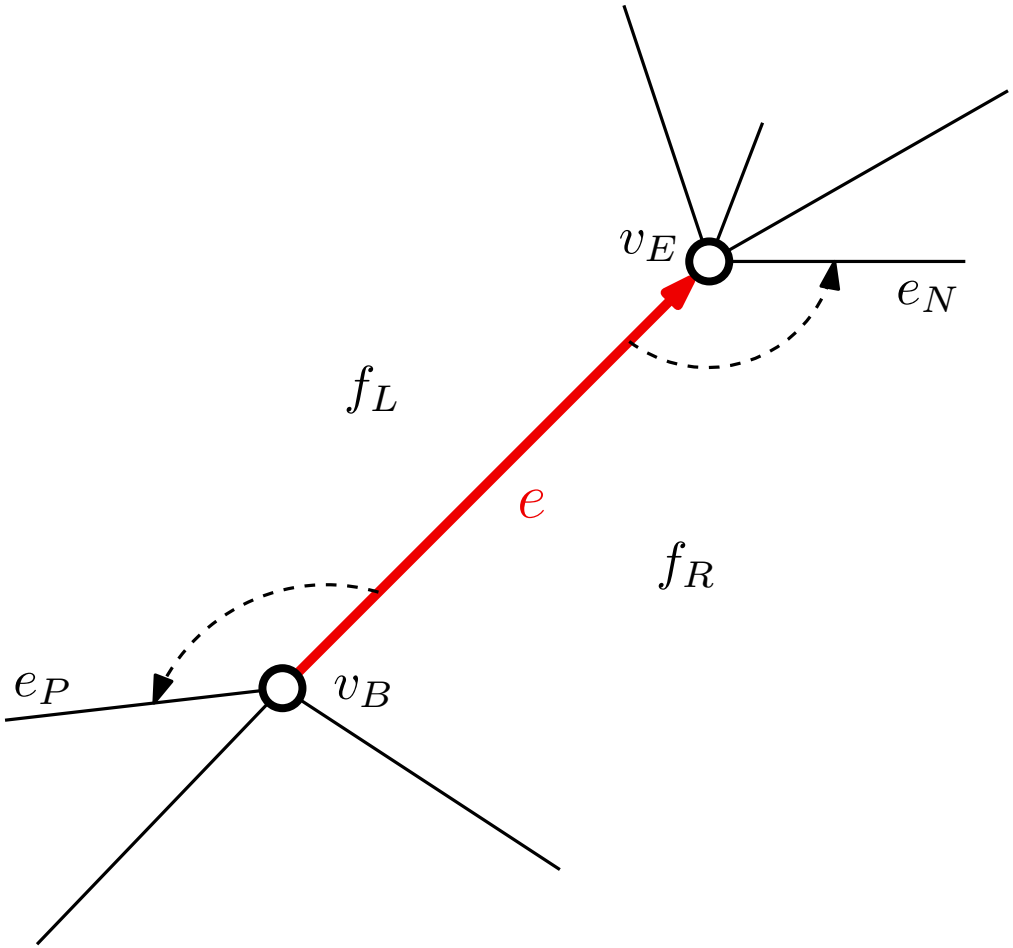
DCEL

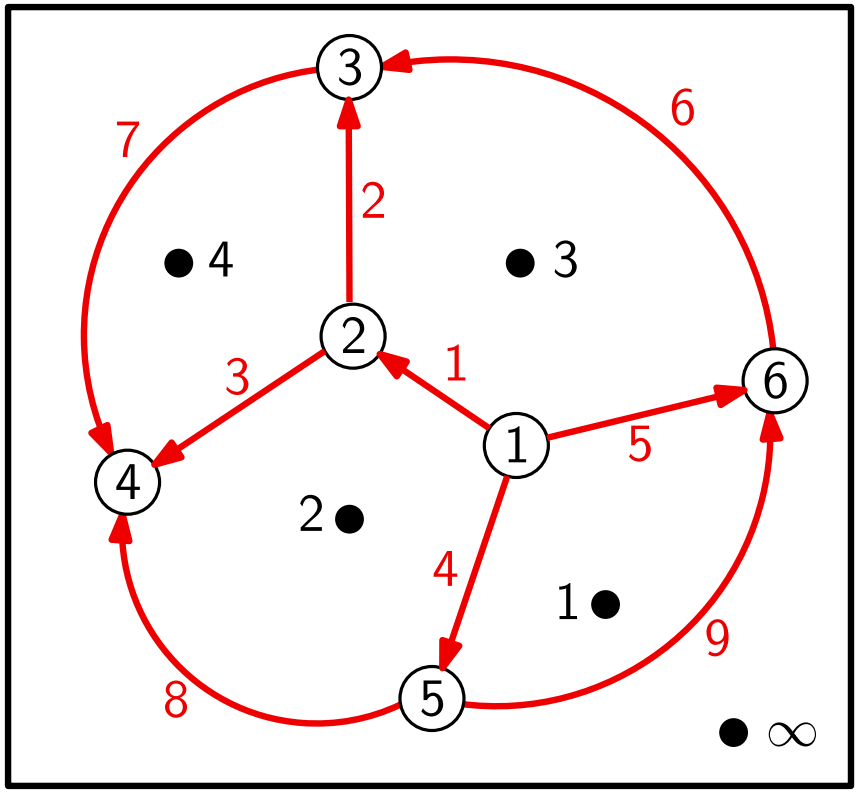




Storing the Voronoi diagram

DCEL



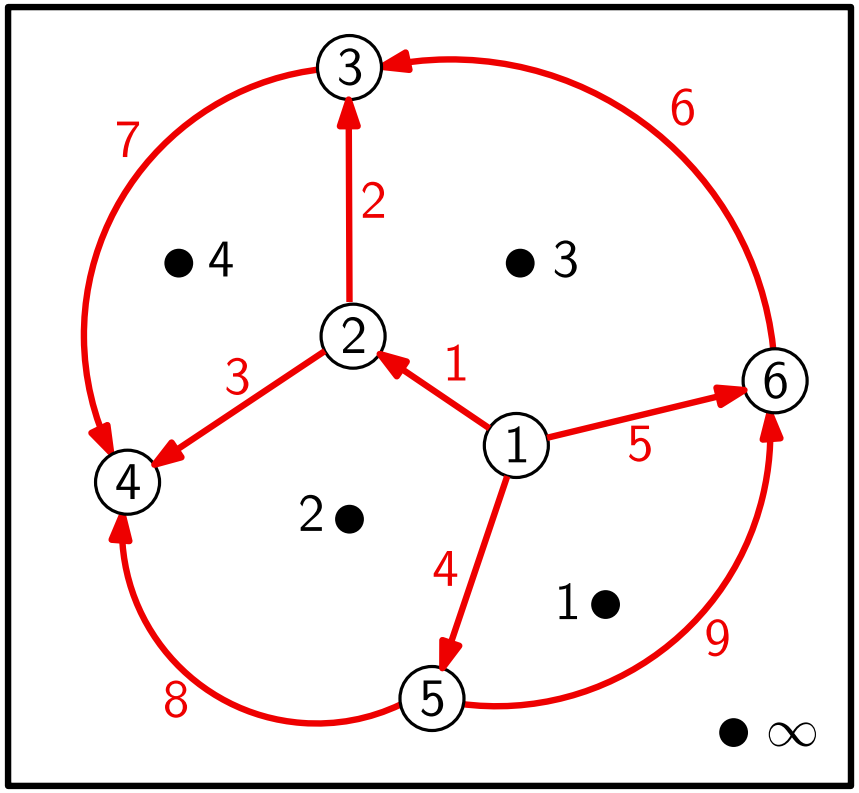


Storing the Voronoi diagram

Table of faces

p	x	y	e
1	x_1	y_1	4
2	x_2	y_2	4
3	x_3	y_3	1
4	x_4	y_4	3
∞	—	—	9

DCEL



Storing the Voronoi diagram

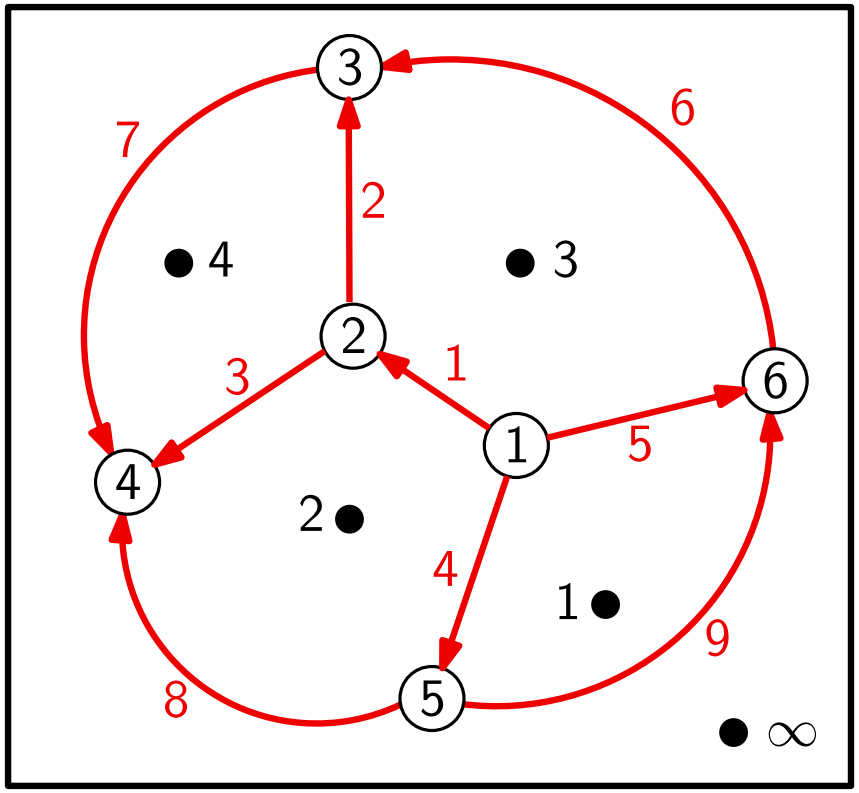
Table of faces

p	x	y	e
1	x_1	y_1	4
2	x_2	y_2	4
3	x_3	y_3	1
4	x_4	y_4	3
∞	—	—	9

Table of vertices

v	x	y	e	original?
1	x_1	y_1	1	1
2	x_2	y_2	1	1
3	x_3	y_3	2	0
4	x_4	y_4	8	0
5	x_5	y_5	4	0
6	x_6	y_6	9	0

DCEL



Storing the Voronoi diagram

Table of faces

p	x	y	e
1	x_1	y_1	4
2	x_2	y_2	4
3	x_3	y_3	1
4	x_4	y_4	3
∞	—	—	9

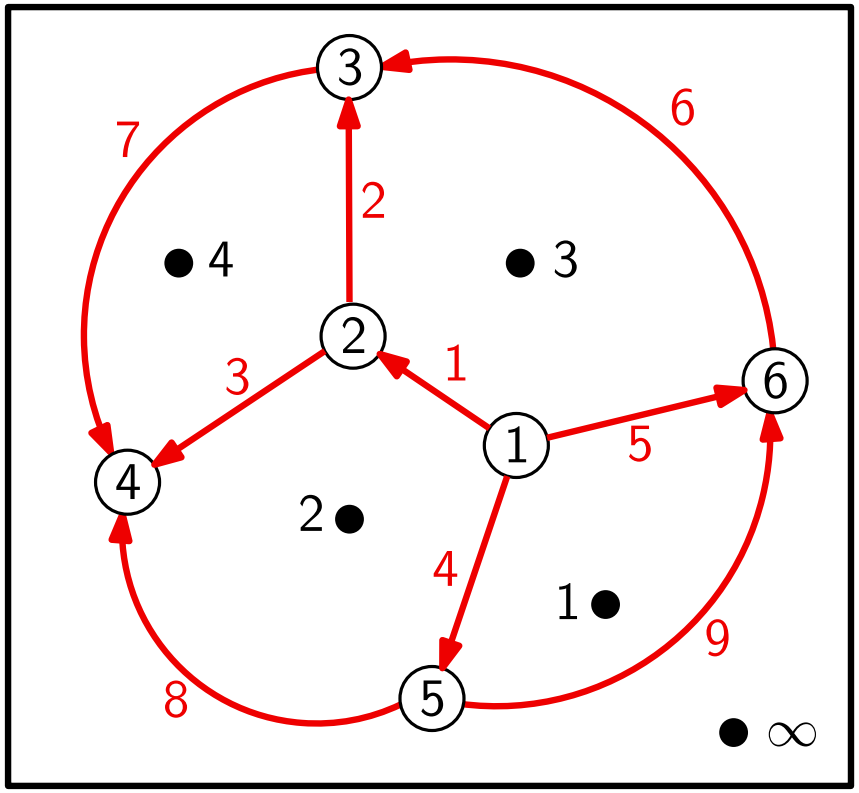
Table of vertices

v	x	y	e	original?
1	x_1	y_1	1	1
2	x_2	y_2	1	1
3	x_3	y_3	2	0
4	x_4	y_4	8	0
5	x_5	y_5	4	0
6	x_6	y_6	9	0

DCEL

DCEL

e	v_B	v_E	f_L	f_R	e_P	e_N
1	1	2	2	3	4	2
2	2	3	4	3	3	6
3	2	4	2	4	1	7
4	1	5	1	2	5	8
5	1	6	3	1	1	9
6	6	3	3	∞	5	7
7	3	4	4	∞	2	8
8	5	4	∞	2	9	3
9	5	6	1	∞	4	6



Storing the Voronoi diagram

Table of faces

p	x	y	e
1	x_1	y_1	4
2	x_2	y_2	4
3	x_3	y_3	1
4	x_4	y_4	3
∞	—	—	9

Table of vertices

v	x	y	e	original?
1	x_1	y_1	1	1
2	x_2	y_2	1	1
3	x_3	y_3	2	0
4	x_4	y_4	8	0
5	x_5	y_5	4	0
6	x_6	y_6	9	0

DCEL

DCEL

e	v_B	v_E	f_L	f_R	e_P	e_N
1	1	2	2	3	4	2
2	2	3	4	3	3	6
3	2	4	2	4	1	7
4	1	5	1	2	5	8
5	1	6	3	1	1	9
6	6	3	3	∞	5	7
7	3	4	4	∞	2	8
8	5	4	∞	2	9	3
9	5	6	1	∞	4	6

Storage space

- For each face:

  2 coordinates + 1 pointer.

- For each vertex:

  2 coordinates + 1 pointer + 1 bit.

- For each edge:

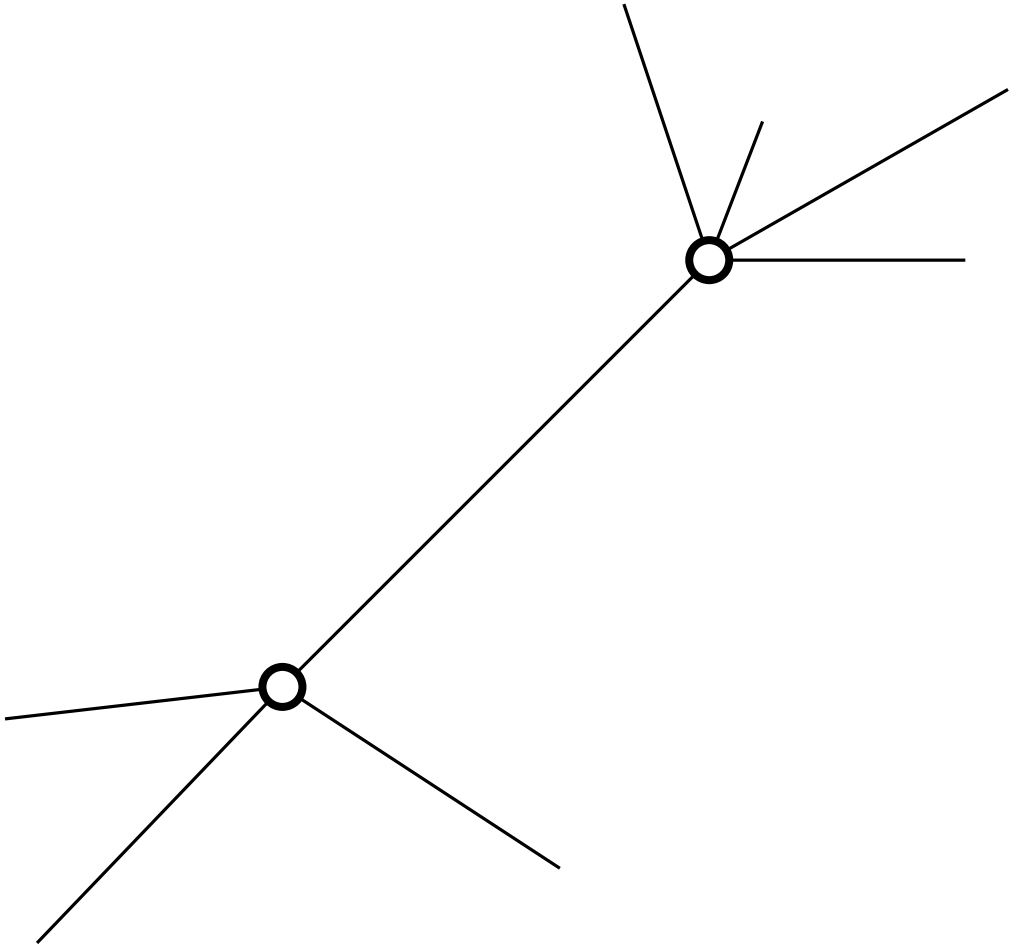
  6 pointers.

In total, the storage space is $O(n)$.

Storing the Voronoi diagram

DCEL

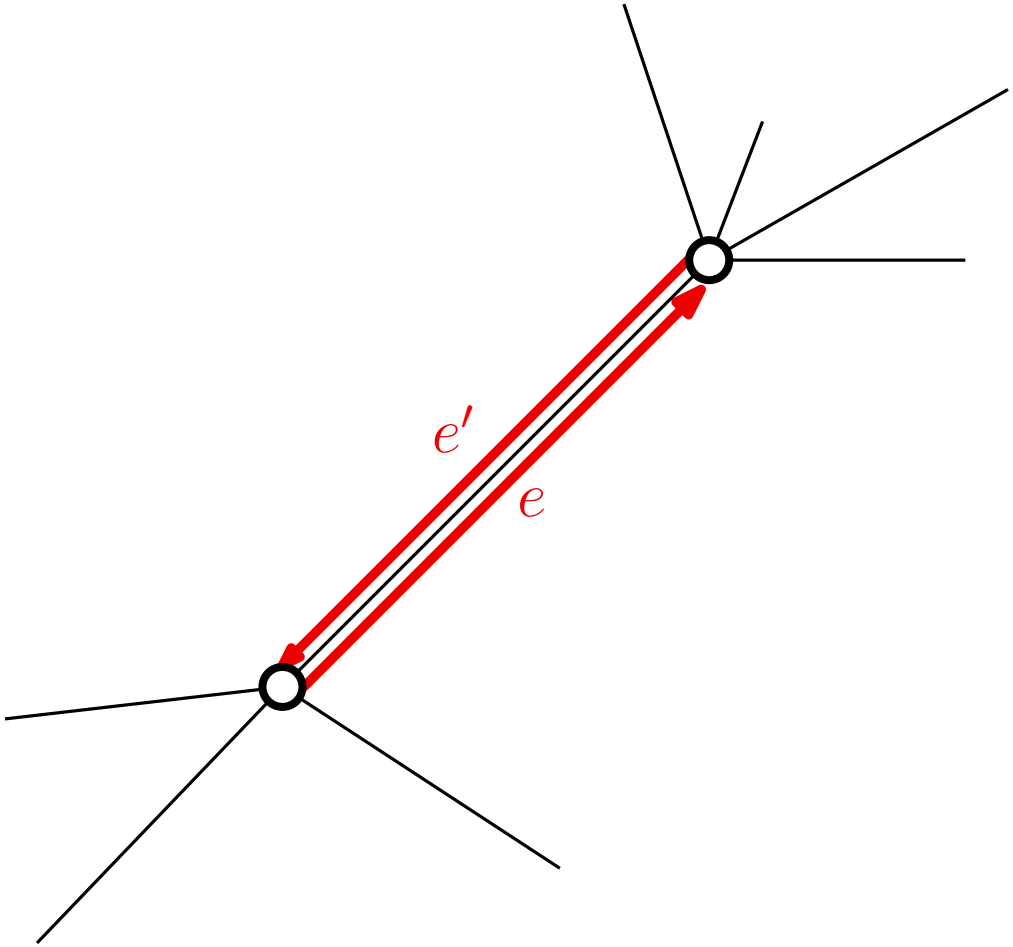
There are other DCEL variants, as for example:



Storing the Voronoi diagram

DCEL

There are other DCEL variants, as for example:

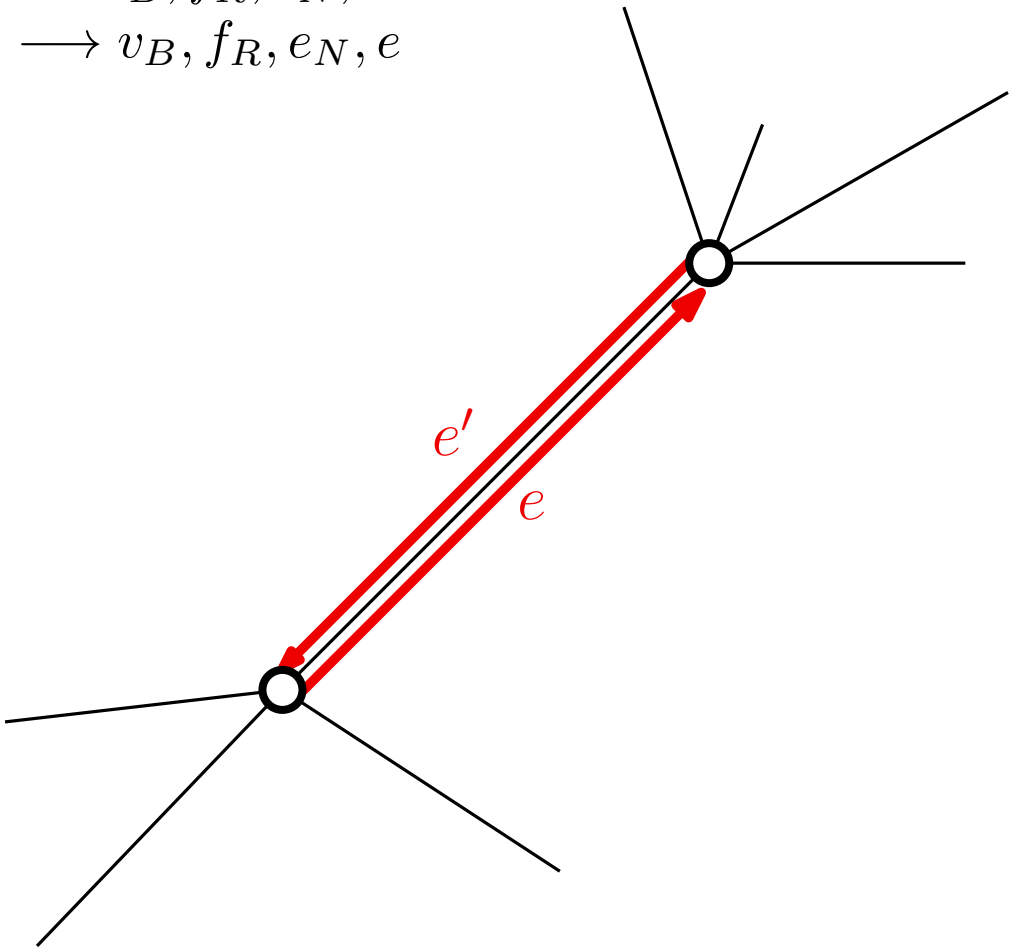


Storing the Voronoi diagram

DCEL

There are other DCEL variants, as for example:

$e \longrightarrow v_B, f_R, e_N, e'$
$e' \longrightarrow v_B, f_R, e_N, e$

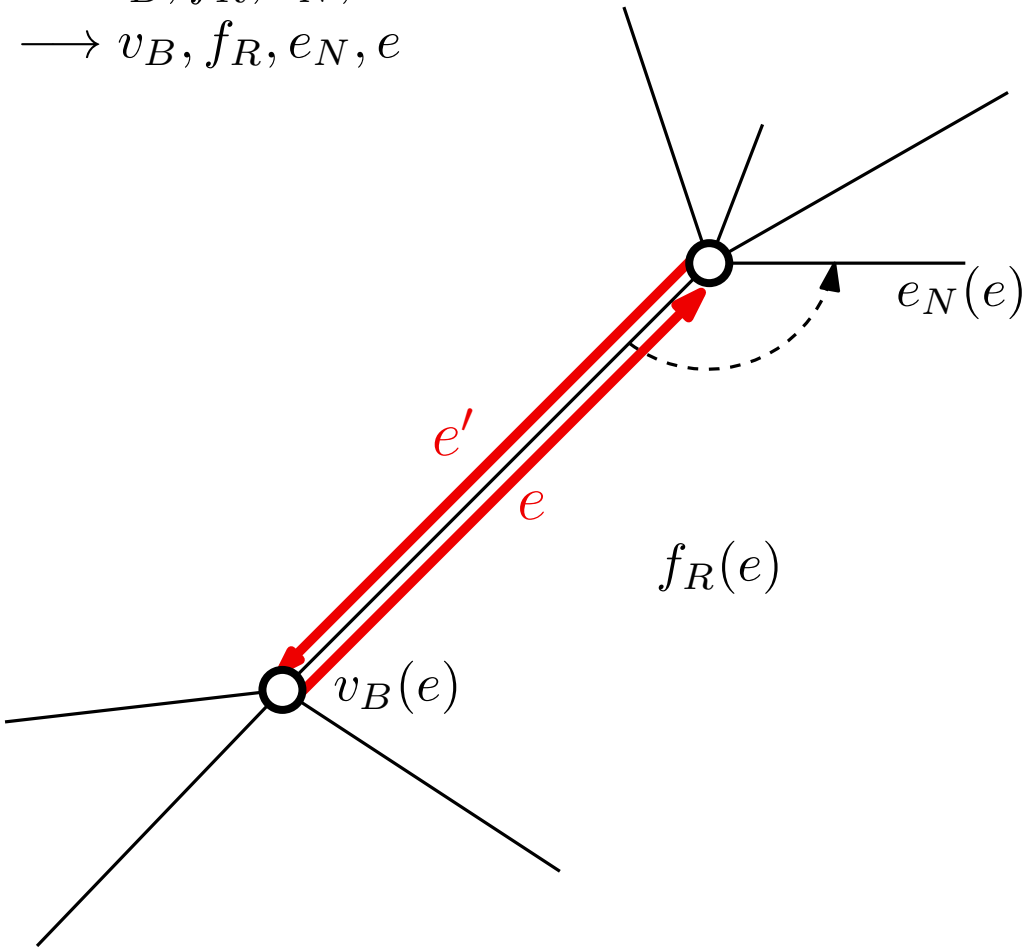


Storing the Voronoi diagram

DCEL

There are other DCEL variants, as for example:

$e \longrightarrow v_B, f_R, e_N, e'$
$e' \longrightarrow v_B, f_R, e_N, e$

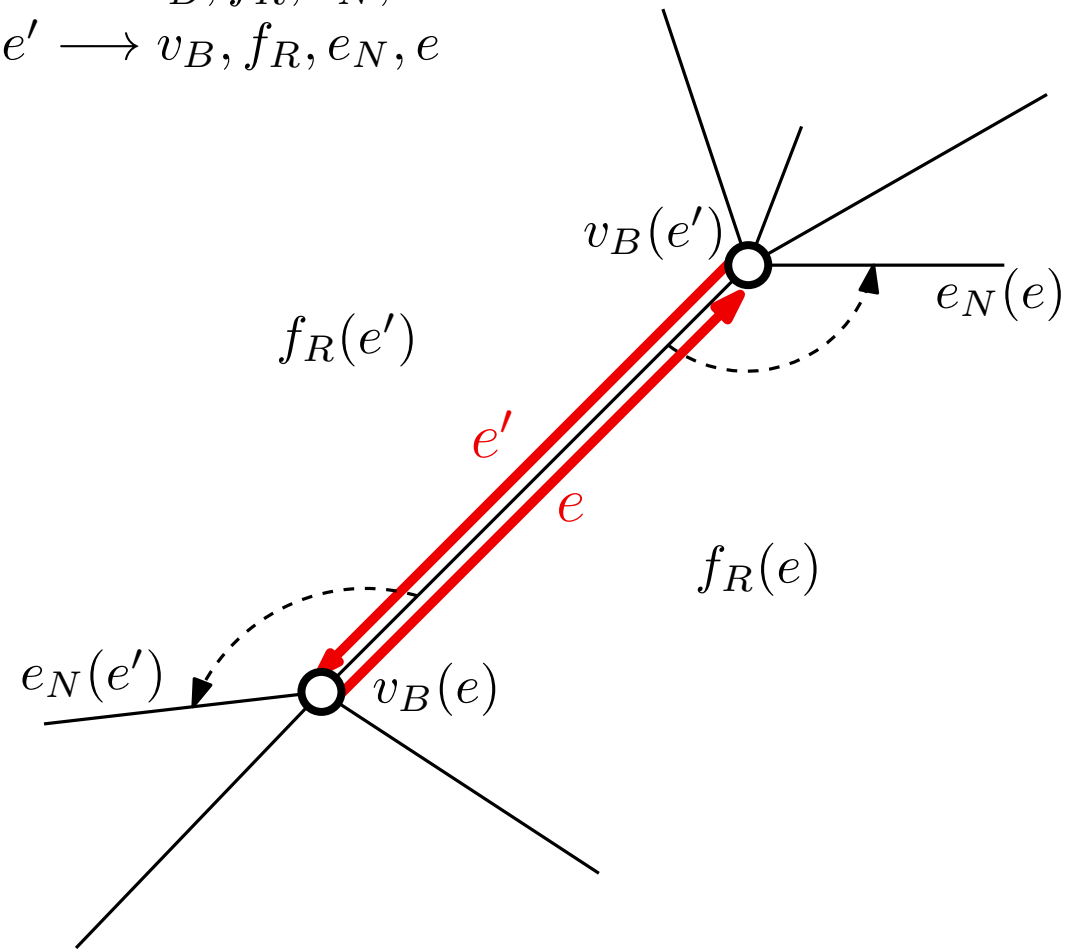


Storing the Voronoi diagram

DCEL

There are other DCEL variants, as for example:

$e \longrightarrow v_B, f_R, e_N, e'$
$e' \longrightarrow v_B, f_R, e_N, e$

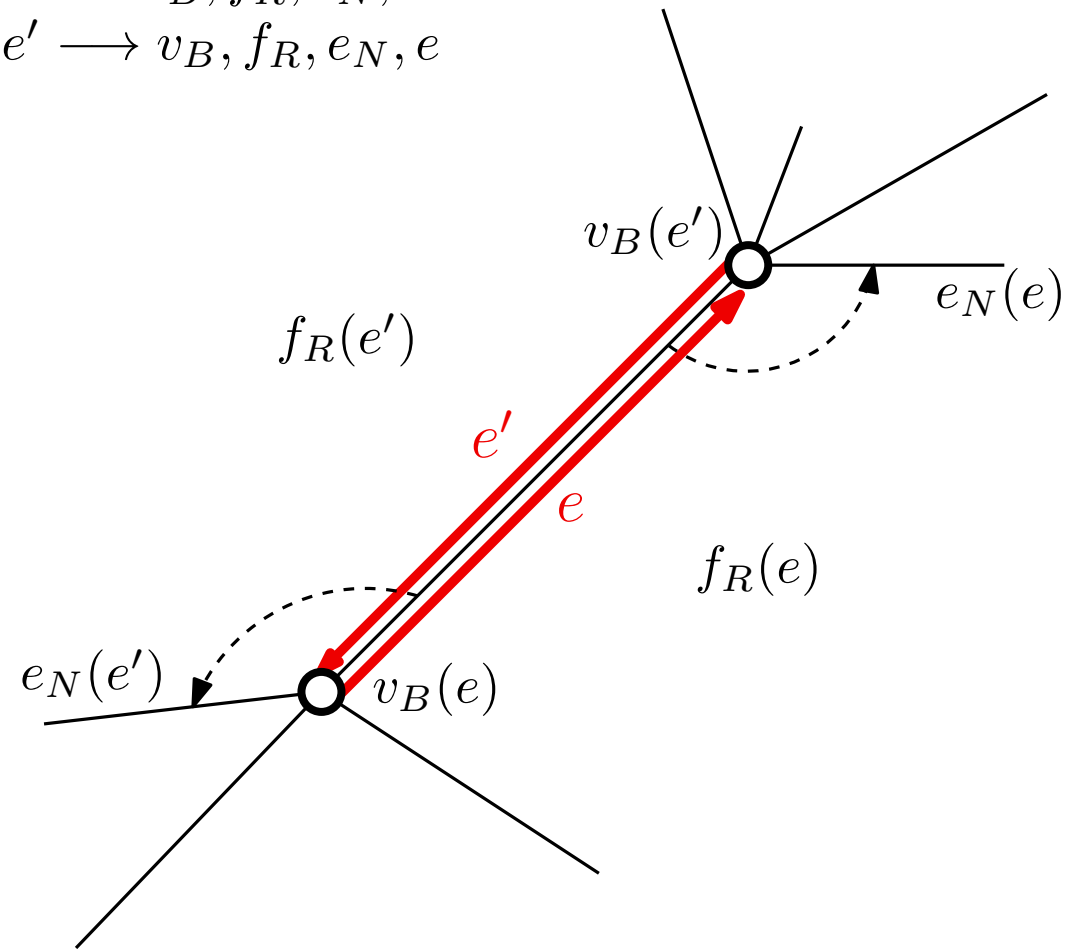


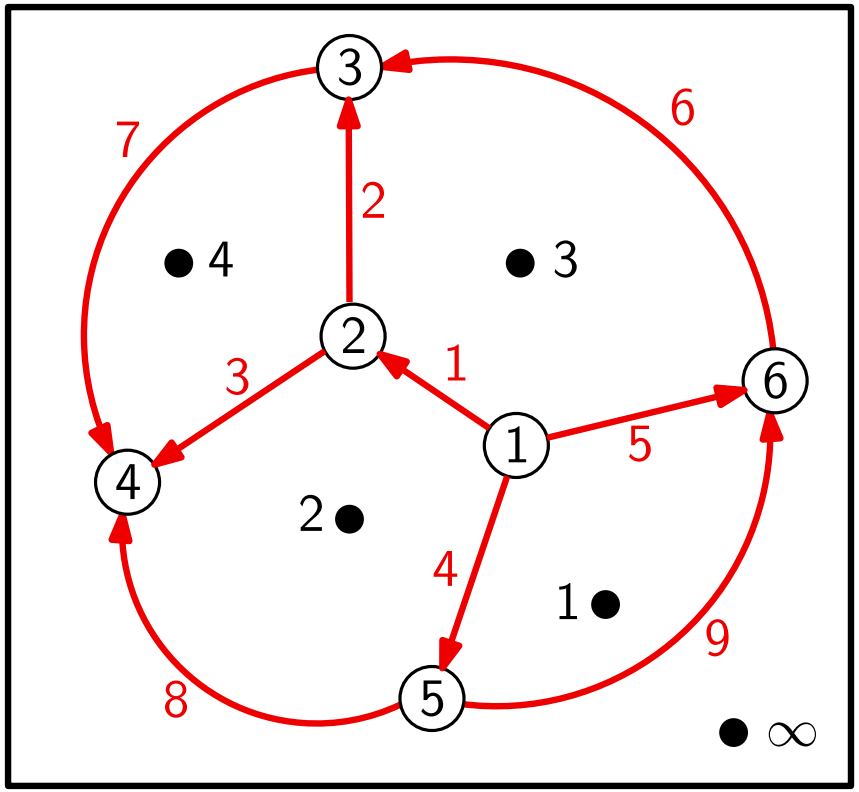
Storing the Voronoi diagram

DCEL

There are other DCEL variants, as for example:

$e \longrightarrow v_B, f_R, e_N, e'$
$e' \longrightarrow v_B, f_R, e_N, e$

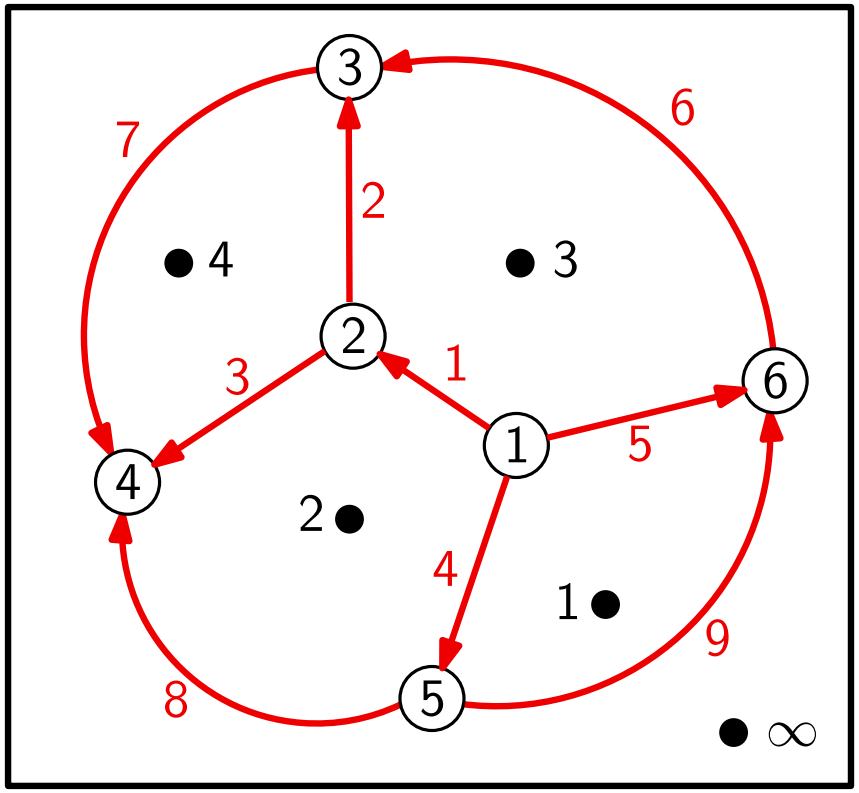




Storing the Voronoi diagram

DCEL

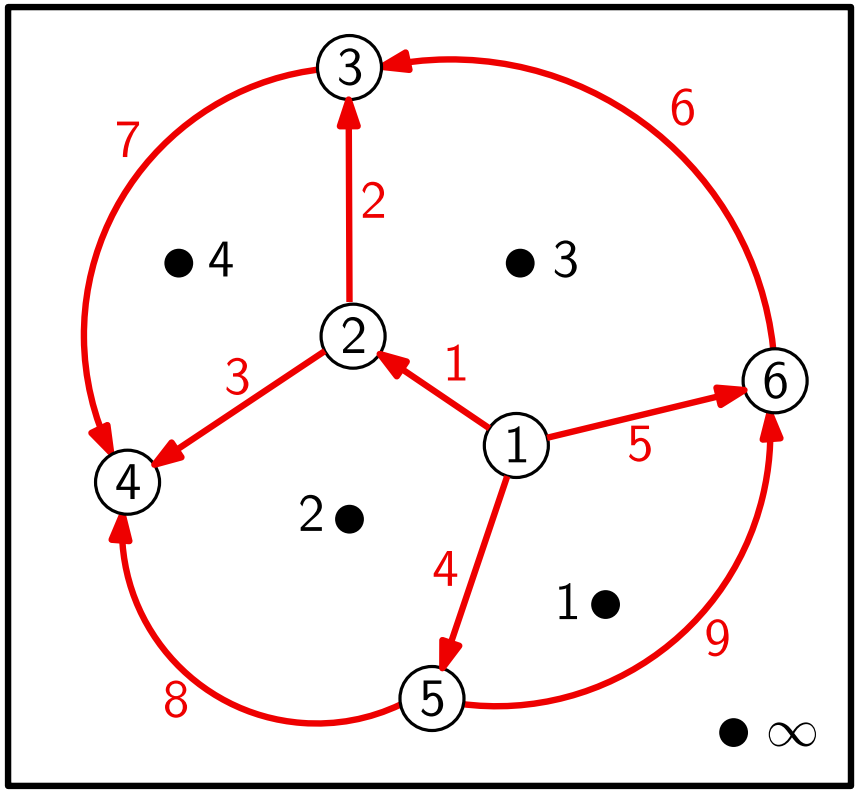
e	v_B	v_E	f_L	f_R	e_P	e_N
1	1	2	2	3	4	2
2	2	3	4	3	3	6
3	2	4	2	4	1	7
4	1	5	1	2	5	8
5	1	6	3	1	1	9
6	6	3	3	∞	5	7
7	3	4	4	∞	2	8
8	5	4	∞	2	9	3
9	5	6	1	∞	4	6



Storing the Voronoi diagram

DCEL

e	v_B	f_R	e_N	e'
1	1	3	2	1'
2	2	3	6	2'
3	2	4	7	3'
4	1	2	8	4'
5	1	1	9	5'
6	6	∞	7	6'
7	3	∞	8	7'
8	5	2	3	8'
9	5	∞	6	9'
1'	2	2	4	1
2'	3	4	3	2
3'	4	2	1	3
4'	5	1	5	4
5'	6	3	1	5
6'	3	3	5	6
7'	4	4	2	7
8'	4	∞	9	8
9'	6	1	4	9



Voronoi diagram storage

How to obtain information from the DCEL

Voronoi diagram storage

How to obtain information from the DCEL

Sorted list of the edges and faces incident to a given Voronoi vertex

Voronoi diagram storage

How to obtain information from the DCEL

Sorted list of the edges and faces incident to a given Voronoi vertex

Input: v_i, a Voronoi vertex
Output: $listE$ and $listF$, sorted in counterclockwise order

Voronoi diagram storage

How to obtain information from the DCEL

Sorted list of the edges and faces incident to a given Voronoi vertex

Input: v_i, a Voronoi vertex
Output: $listE$ and $listF$, sorted in counterclockwise order

Procedure:

Initialization

$listE = \{\ \}$, $listF = \{\ \}$, $e = e(v_i)$

Advance

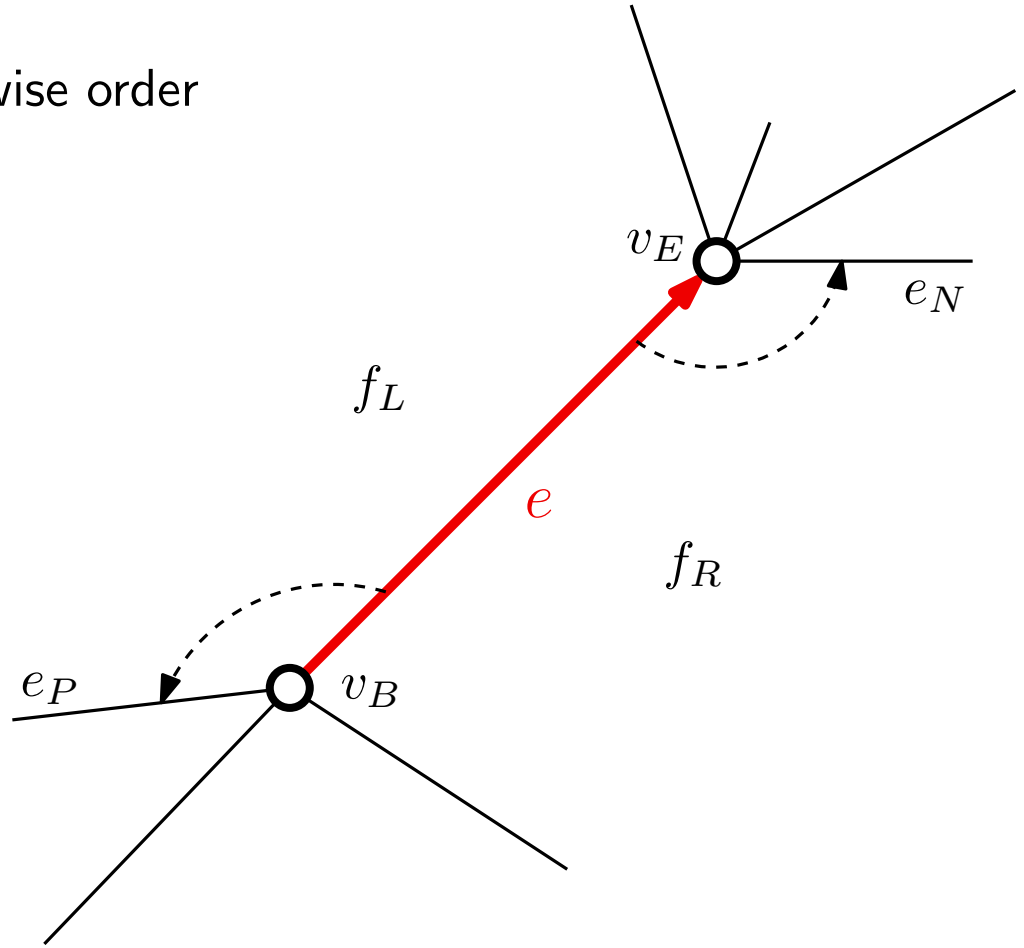
Add e to $listE$
If $i = v_B(e)$, then

add $f_L(e)$ to $listF$
$e = e_P(e)$

else

add $f_R(e)$ to $listF$
$e = e_N(e)$

Repeat until e coincides again with $e(v_i)$



Voronoi diagram storage

How to obtain information from the DCEL

Sorted list of the edges and faces incident to a given Voronoi vertex

Input: v_i, a Voronoi vertex
Output: $listE$ and $listF$, sorted in counterclockwise order

Procedure:

Initialization

$listE = \{\ \}$, $listF = \{\ \}$, $e = e(v_i)$

Advance

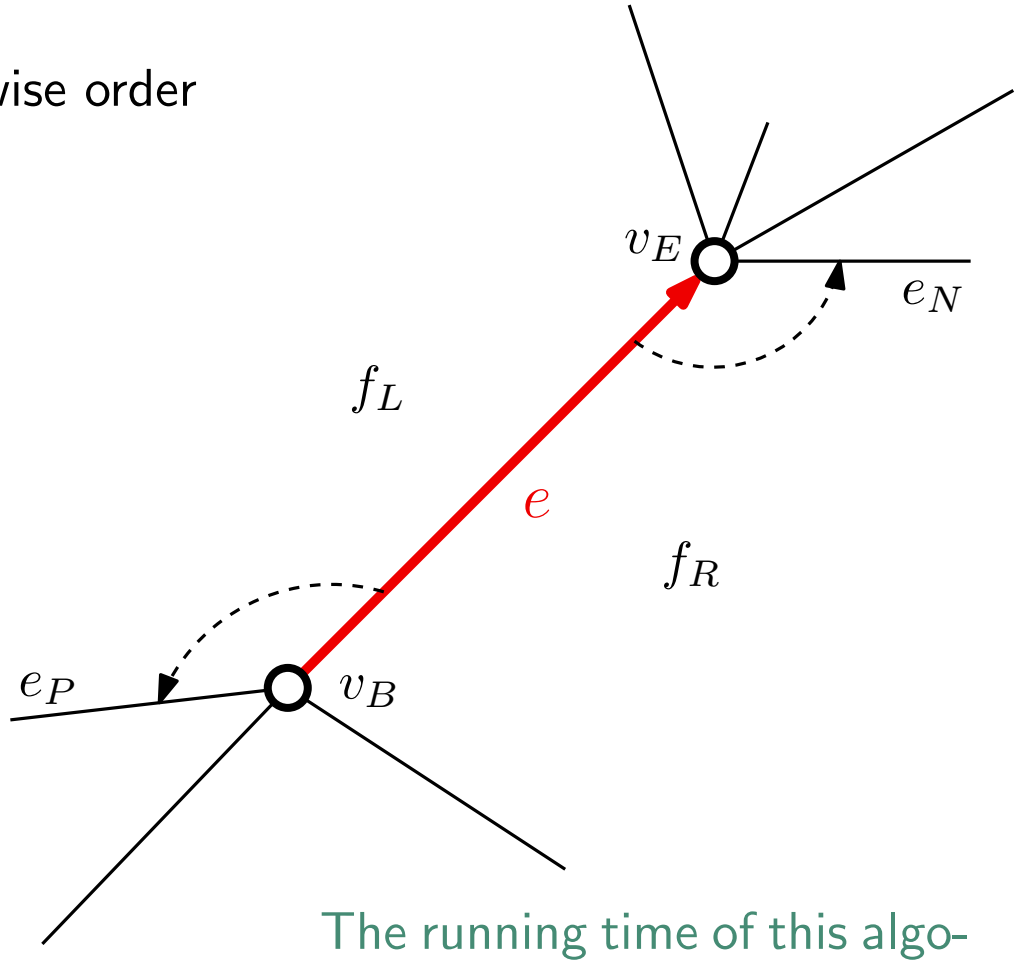
Add e to $listE$
If $i = v_B(e)$, then

add $f_L(e)$ to $listF$
$e = e_P(e)$

else

add $f_R(e)$ to $listF$
$e = e_N(e)$

Repeat until e coincides again with $e(v_i)$



The running time of this algorithm is linear in the number of edges (faces) incident to v_i

Voronoi diagram storage

How to obtain information from the DCEL

Sorted list of vertices and edges of a Voronoi region

Voronoi diagram storage

How to obtain information from the DCEL

Sorted list of vertices and edges of a Voronoi region

Input: p_i, a site
Output: $listE$ and $listV$, sorted in clockwise order

Voronoi diagram storage

How to obtain information from the DCEL

Sorted list of vertices and edges of a Voronoi region

Input: p_i, a site
Output: $listE$ and $listV$, sorted in clockwise order

Procedure:

Initialization

$listE = \{\ \}$, $listV = \{\ \}$, $e = e(p_i)$

Advance

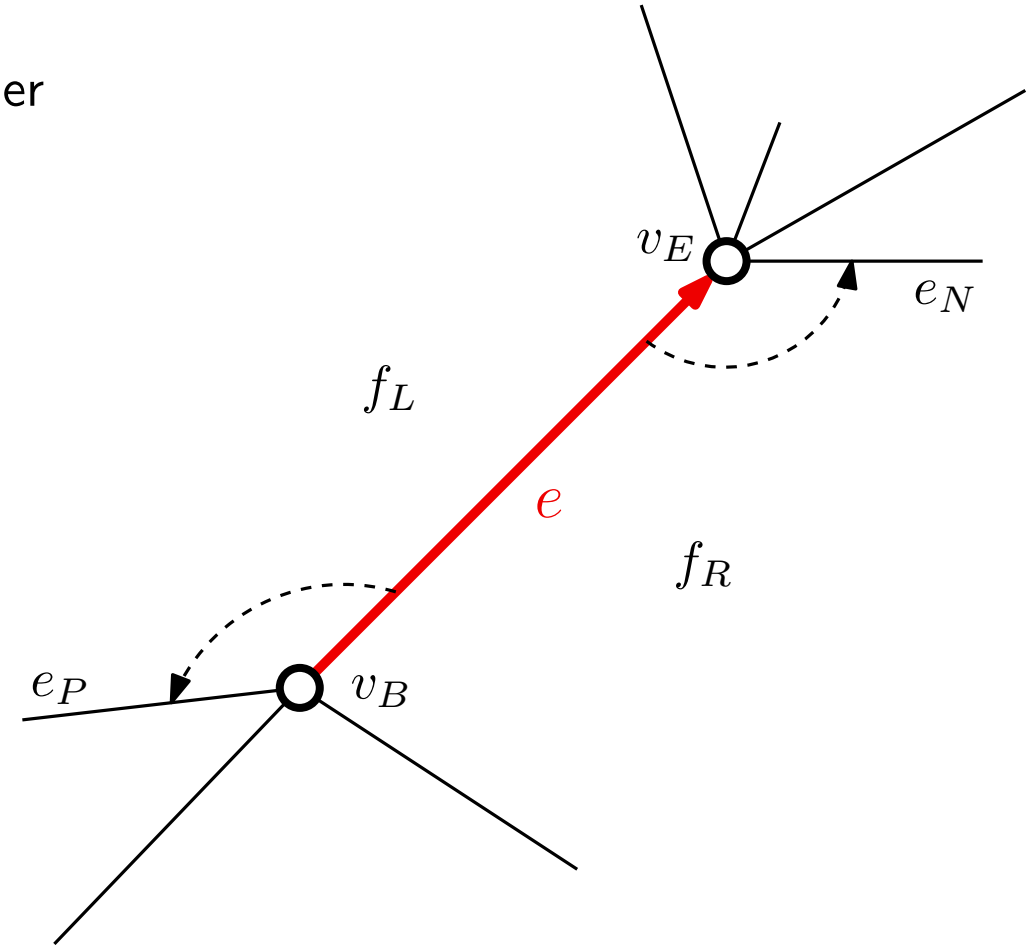
Add e to $listE$
If $i = f_L(e)$, then

add $v_B(e)$ to $listV$
$e = e_P(e)$

else

addr $v_E(e)$ to $listV$
$e = e_N(e)$

Repeat until e coincides again with $e(v_i)$



Voronoi diagram storage

How to obtain information from the DCEL

Sorted list of vertices and edges of a Voronoi region

Input: p_i, a site
Output: $listE$ and $listV$, sorted in clockwise order

Procedure:

Initialization

$listE = \{\ \}$, $listV = \{\ \}$, $e = e(p_i)$

Advance

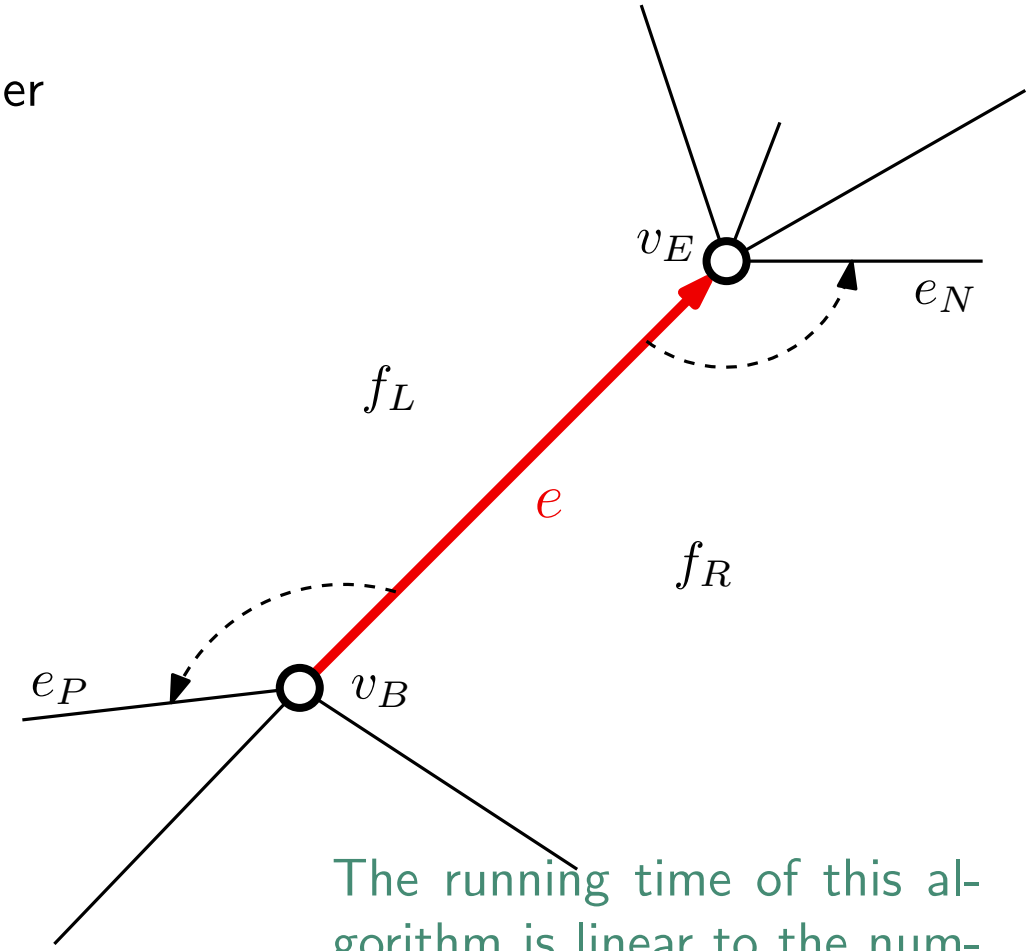
Add e to $listE$
If $i = f_L(e)$, then

add $v_B(e)$ to $listV$
$e = e_P(e)$

else

addr $v_E(e)$ to $listV$
$e = e_N(e)$

Repeat until e coincides again with $e(v_i)$

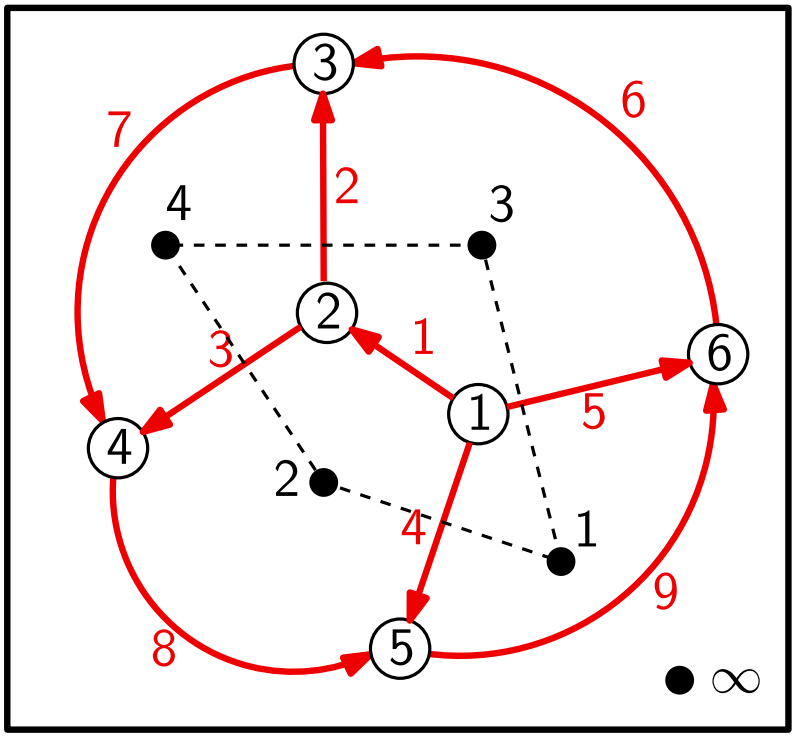


The running time of this algorithm is linear to the number of edges (vertices) of the Voronoi region of p_i

Voronoi diagram storage

How to obtain information from the DCEL

Convex hull of P

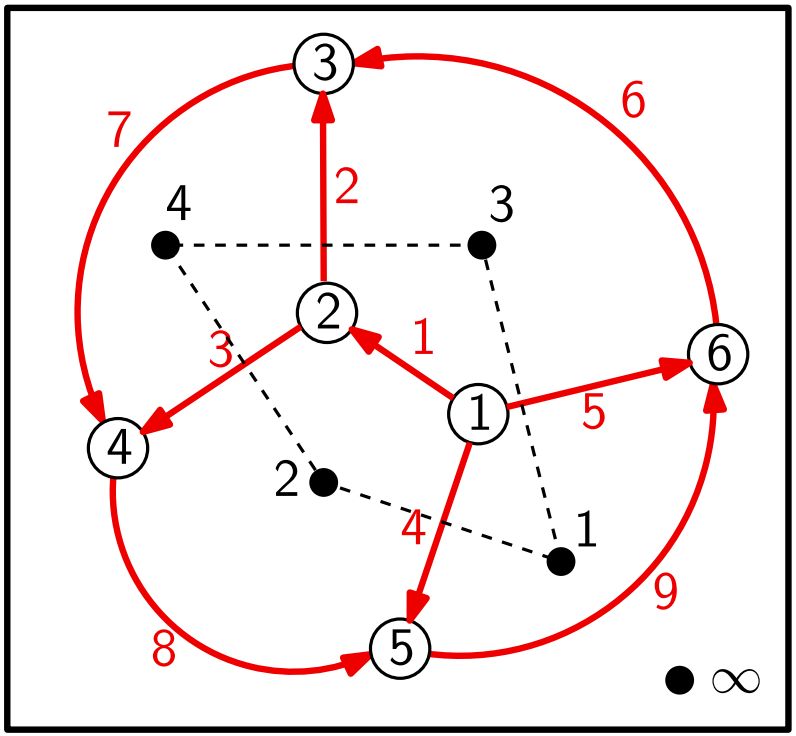


Voronoi diagram storage

How to obtain information from the DCEL

Convex hull of P

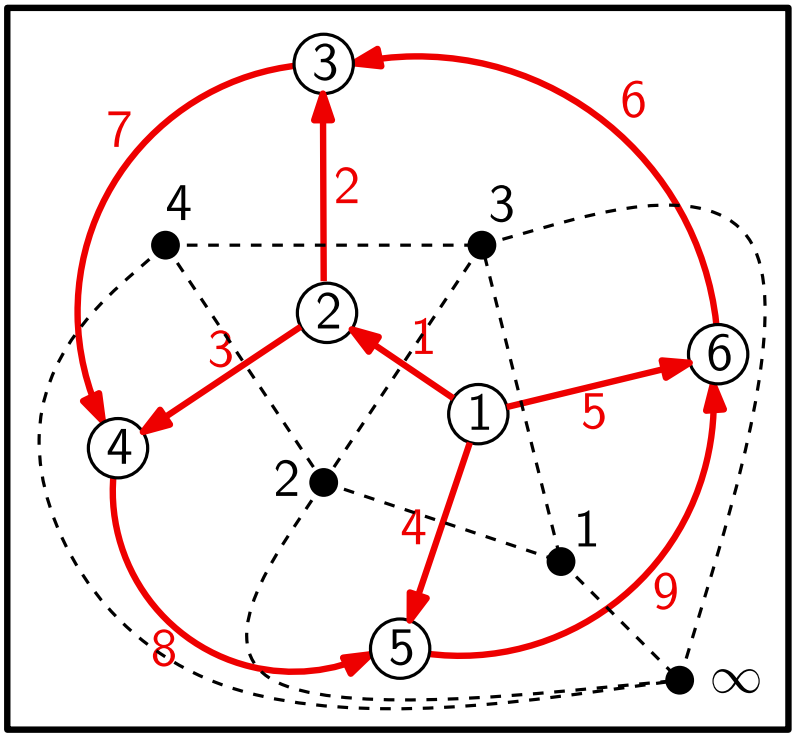
When $p_i = \infty$, the previous algorithm returns, in counterclockwise order, the (fictitious) edges of the Voronoi region of this (fictitious) point. For each obtained edge, reporting its other adjacent Voronoi face will produce the sorted list of the convex hull vertices of P, in time proportional to its size.



Voronoi diagram storage

How to obtain information from the DCEL

Delaunay diagram

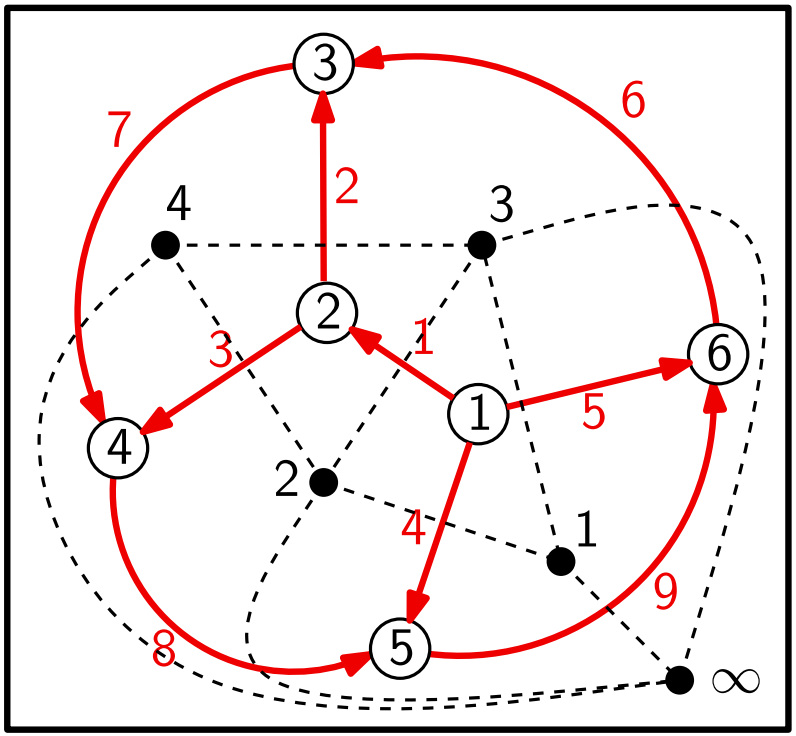


Voronoi diagram storage

How to obtain information from the DCEL

Delaunay diagram

The DCEL storing the Voronoi diagram information and the DCEL storing the Delaunay triangulation information are the same, we just need to do some "dual reading":



Voronoi diagram storage

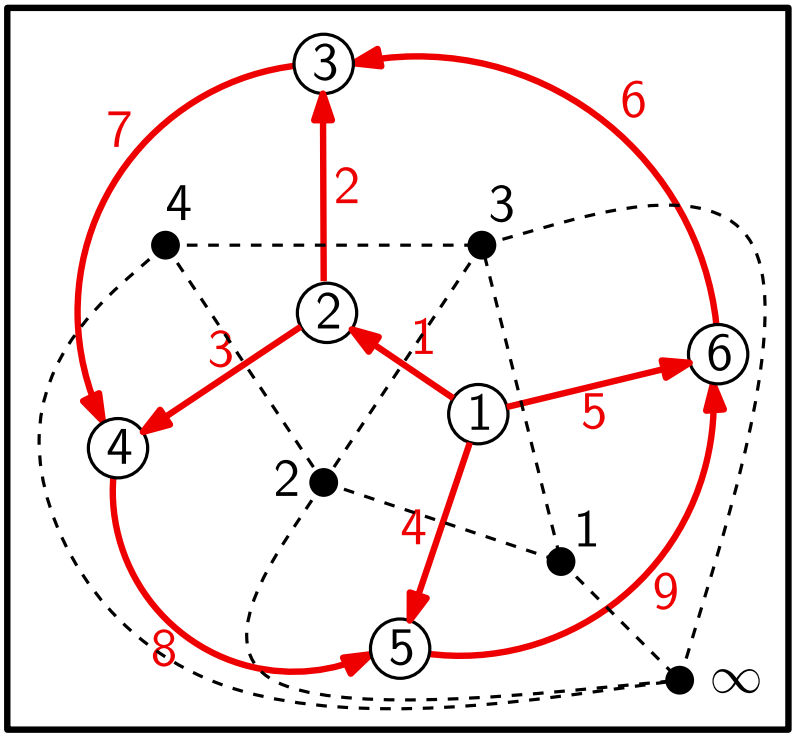
How to obtain information from the DCEL

Delaunay diagram

The DCEL storing the Voronoi diagram information and the DCEL storing the Delaunay triangulation information are the same, we just need to do some "dual reading":

Voronoi:

Delaunay:



Voronoi diagram storage

How to obtain information from the DCEL

Delaunay diagram

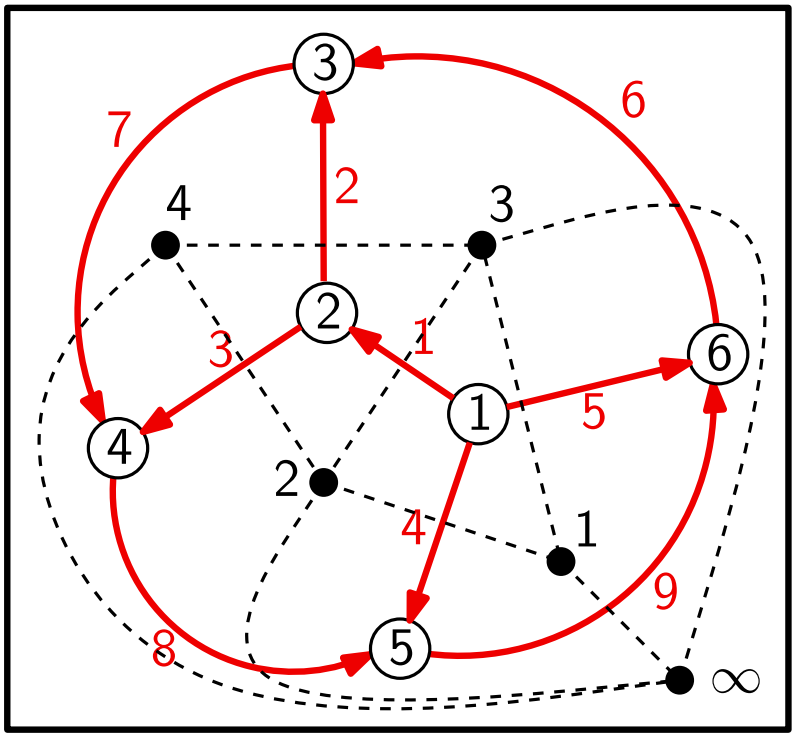
The DCEL storing the Voronoi diagram information and the DCEL storing the Delaunay triangulation information are the same, we just need to do some "dual reading":

Voronoi: faces

p	x	y	e

Delaunay: vertices

p	x	y	e incident



Voronoi diagram storage

How to obtain information from the DCEL

Delaunay diagram

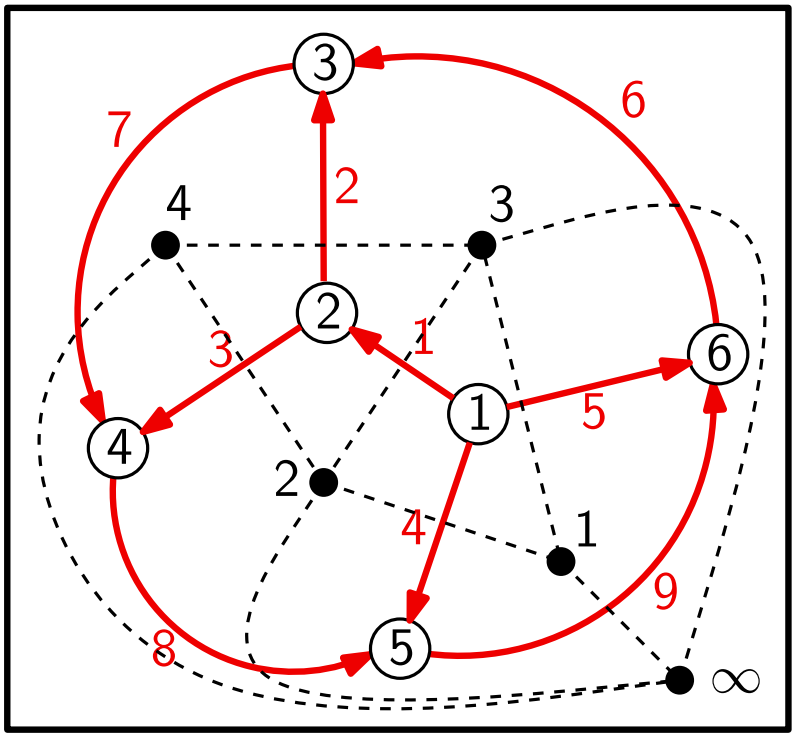
The DCEL storing the Voronoi diagram information and the DCEL storing the Delaunay triangulation information are the same, we just need to do some "dual reading":

Voronoi: vertices

v	x	y	e	original?

Delaunay: triangles

v	x	y	e	original?
	circumcenter		edge	



Voronoi diagram storage

How to obtain information from the DCEL

Delaunay diagram

The DCEL storing the Voronoi diagram information and the DCEL storing the Delaunay triangulation information are the same, we just need to do some "dual reading":

Voronoi: edges

e	v_B	v_E	f_L	f_R	e_P	e_N

Delaunay: edges

e dual	f_L	f_R	v_E	v_B	e_N	e_P clockwise