

DCGI

DEPARTMENT OF COMPUTER GRAPHICS AND INTERACTION

ARRANGEMENTS (uspořádání)

PETR FELKEL

FEL CTU PRAGUE

felkel@fel.cvut.cz

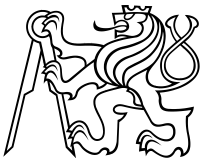
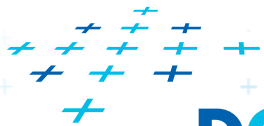
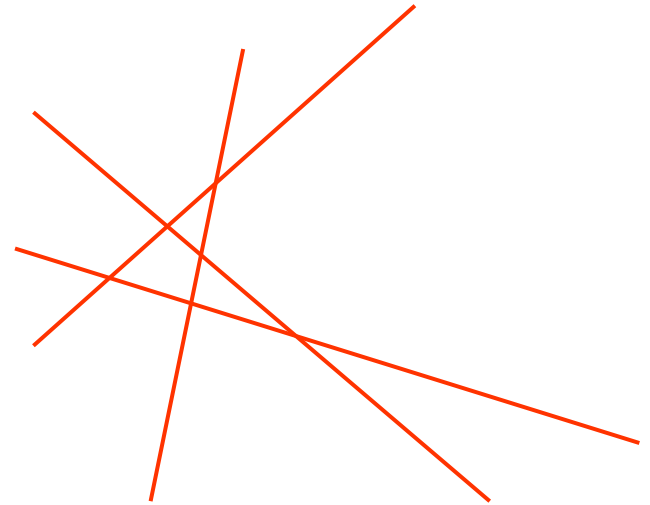
<https://cw.felk.cvut.cz/doku.php/courses/a4m39vg/start>

Based on [Berg], [Mount]

Version from 04.12.2024

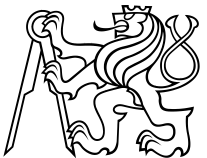
Talk overview

- Arrangements of lines
 - Incremental construction
 - Topological plane sweep
- Duality – next lesson



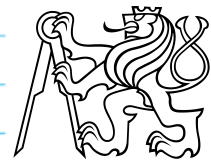
Arrangements

- The next most important structure in CG after CH, VD, and DT
- Possible in any dimension
arrangement of $(d-1)$ -dimensional hyperplanes
- We concentrate on arrangement of lines in plane
- Typical application:
problems of point sets in dual plane
(collinear points, point on circles, ...)



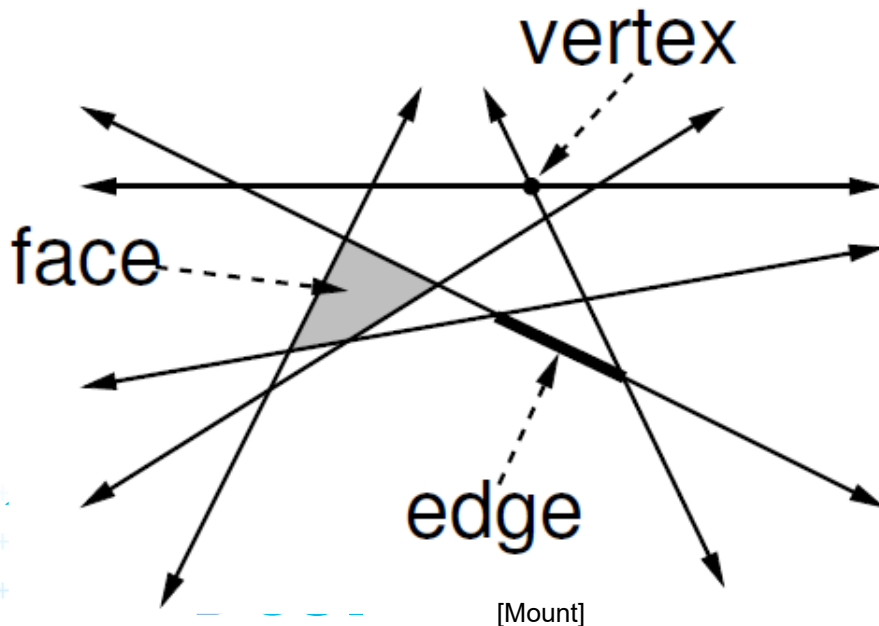
Some more applications (see CGAL)

- Finding the **minimum-area triangle** defined by a set of points,
- computation of the **sorted angular sequences** of points,
- finding the **ham-sandwich cut**,
- planning the **motion of a polygon** translating among polygons in the plane,
- computing the **offset polygon**,
- constructing the **farthest-point Voronoi diagram**,
- coordinating the **motion of two discs** moving among obstacles in the plane,
- performing **Boolean operations on curved polygons**.



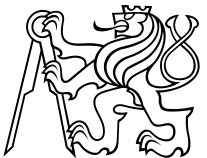
Line arrangement - definition

- A finite set L of lines subdivides the plane into a cell complex, called arrangement $A(L)$
- In plane, arrangement defines a planar graph
 - vertices – intersections of (2 or more) lines
 - edges – intersection free segments (or rays or lines)
 - faces – convex regions containing no line (possibly unbounded)



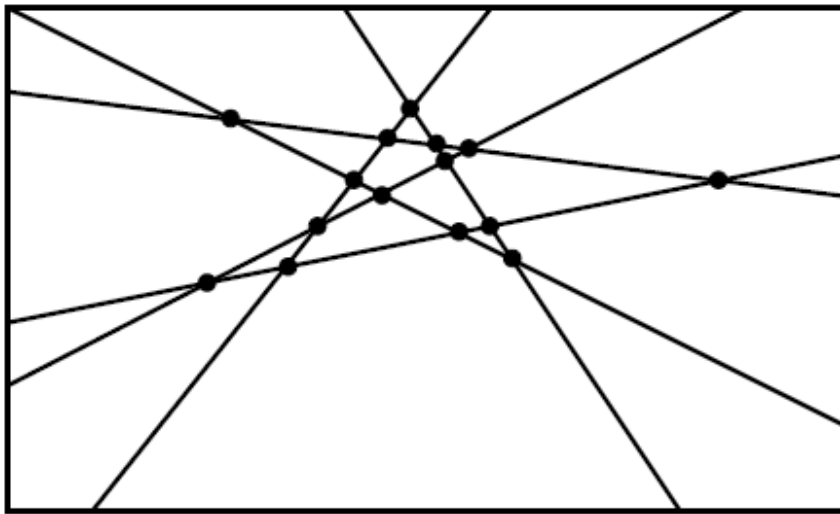
Line arrangement - assumption

- Simple arrangement assumption
 - = no three lines intersect in a single point
 - Can be solved by careful implementation or symbolic perturbation

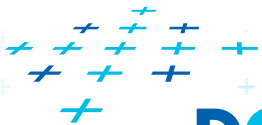


Line arrangement - graph

- Formal problem: graph must have bounded edges
 - Topological fix: add vertex in infinity
 - Geometrical fix: BBOX, often enough as abstract with corners $\{-\infty, -\infty\}, \{\infty, \infty\}$



bounding box [Mount]



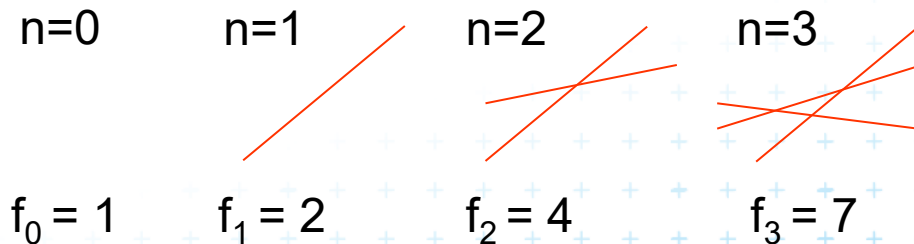
Combinatorial complexity of line arrangement

- $O(n^2)$
- Given n lines in general position, max numbers are
 - Vertices $\binom{n}{2} = \frac{n(n-1)}{2} \rightarrow$ each line intersect $n - 1$ others
 - Edges $n^2 \rightarrow n - 1$ intersections create n edges on each of n lines

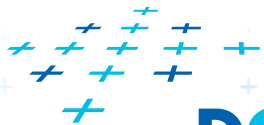
– Faces $\frac{n(n+1)}{2} + 1 = \binom{n}{2} + n + 1$

$f_0 = 1$ (whole plane)

$f_n = f_{n-1} + n$



$$f_n = f_0 + \sum_{i=1}^n i = \frac{n(n+1)}{2} + 1$$



Construction of line arrangement

(0. Plane sweep method)

- $O(n^2 \log n)$ time and $O(n)$ storage plus $O(n^2)$ storage for the arrangement (n^2 vertices, edges, faces. $\log n^2$ - heap & BVS access)

$$\begin{aligned} & n^2 \log n^2 \\ &= 2n^2 \log n \\ &= O(n^2 \log n) \end{aligned}$$

A. Incremental method


- $O(n^2)$ time and $O(n^2)$ storage
- Optimal method

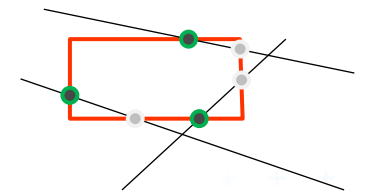
B. Topological plane sweep

- $O(n^2)$ time and $O(n)$ storage only
- Does not store the result arrangement
- Useful for applications that may throw out the arrangement after processing



A. Incremental construction of arrangement

- $O(n^2)$ time, $O(n^2)$ space
~size of arrangement => it is an **optimal algorithm**
- **Not randomized** – depends on n only, not on order
- Create BBOX  (4 edges)
- Add line l_i one by one ($i = 1 \dots n$)
 - Find the leftmost intersection with the BBOX among $2(i - 1) + 4$ edges already on the BBOX ... $O(i)$
 - Trace the line through the arrangement $A(L_{i-1})$ and split the intersected faces ... $O(i)$ – why? See later
 - Update the subdivision (cell split) ... $O(1)$

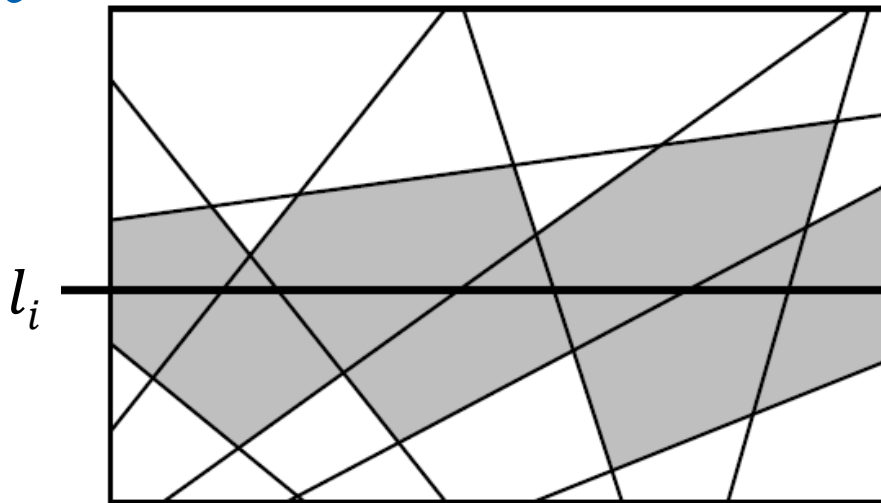


- Altogether $O(ni) = O(n^2)$



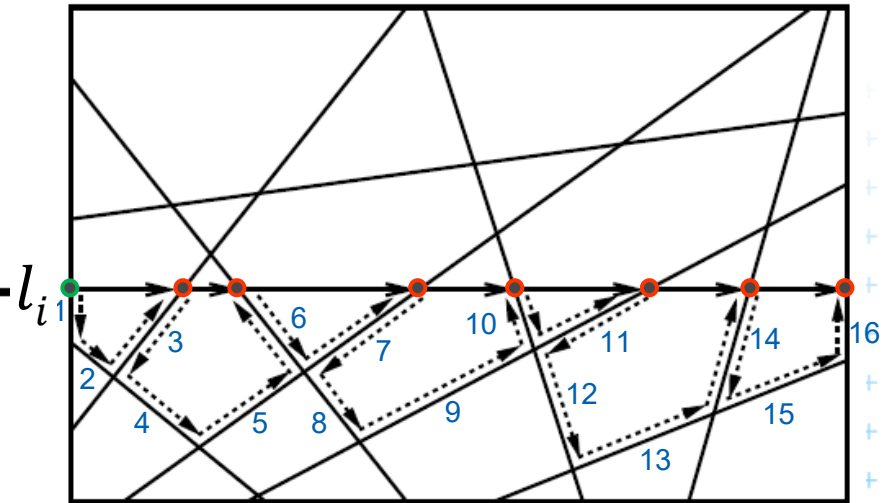
A. Tracing the line through the arrangement

- Walk around edges of current face (face walking)
- Determine if the line l_i intersects current edge e
- When intersection found, jump to the face on the other side of edge e

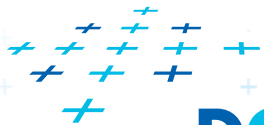


The zone of l_i

$n=8$ lines, 7 faces in the zone, 16 edges tested of max 48



Walking the lower part
of the zone



A. Incremental construction of arrangement

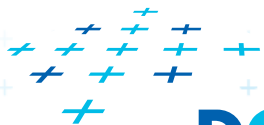
Arrangement(L)

Input: Set of lines L in general position (no 3 intersect in 1 common point)

Output: Line arrangement $A(L)$ (resp. part of the arrangement stored in BBOX $B(L)$ containing all the vertices of $A(L)$)

1. Compute the BBOX $B(L)$ containing all the vertices of $A(L)$ $\dots O(n^2)$
2. Construct DCEL for the subdivision induced by BBOX $B(L)$ $\dots O(1)$
3. **for** $i = 1$ **to** n **do** // *insert line* l_i
4. find edge e , where line l_i intersects the BBOX of $2(i - 1) + 4$ edges $\dots O(i)$
5. $f =$ bounded face incident to the edge e
6. **while** f is in BBOX $B(L)$ (bounded face $f \Leftrightarrow f$ is in the BBOX) $\dots O(i)$
7. split f and set f to be the next intersected face across the intersected edge
8. update the DCEL (split the cell of f)

$\dots O(1)$

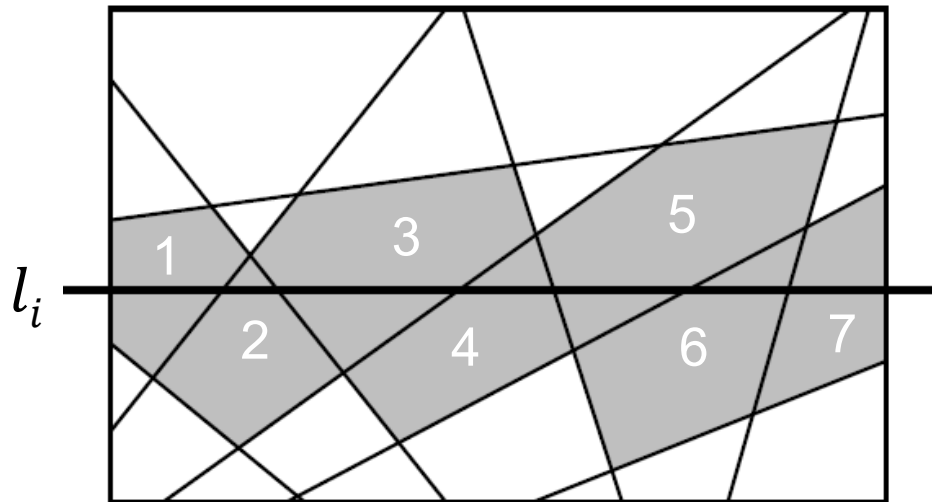


DCGI

See now...



The Zone of edge l_i – has max i faces



The zone of l_i for $i = 9$

$$i = 9$$

⇒ 8 lines there

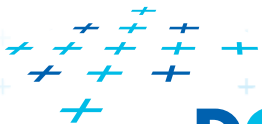
⇒ all may be crossed
(only parallel not)

⇒ max 9 faces in the zone

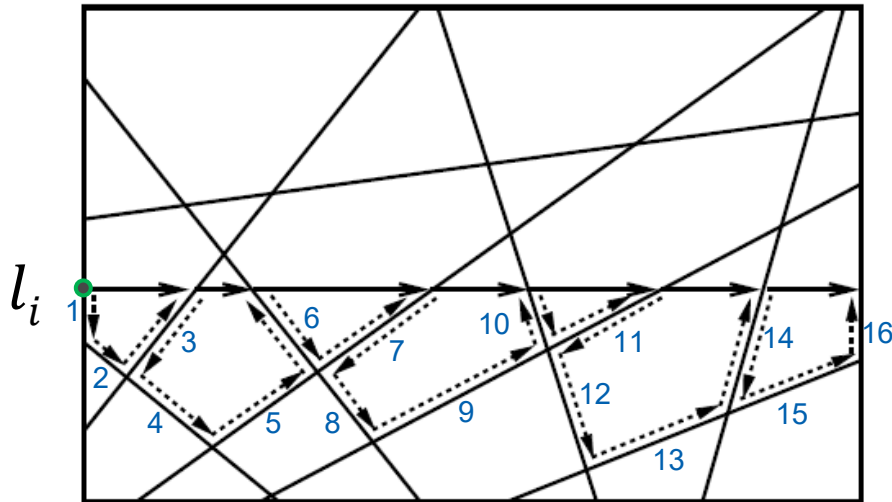
Zone $Z_A(l_i)$ = set of max i faces of $A(L)$ intersected by l_i

l_i crosses max $i - 1$ lines ⇒ can cross max i faces

Here l_9 : 7 of max 9 possible faces in the zone
(we intersect 6 lines only, not the left- and right-most
⇒ not a correct figure, BBOX should be enlarged)



Edges in the cells of the zone faces



Total number of edges in all zone faces

Naïve upper bound

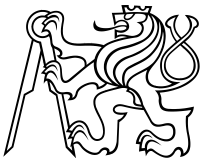
edge l_i passes max i faces ... $O(i)$

each face bounded by at most i lines

} $O(i^2)$????

Tight upper bound $6i = O(i)$

$n=8$ lines, 16 edges tested of max 48

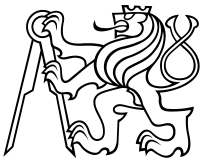


Tracing the line through the arrangement

- Number of traversed edges determines the insertion complexity
- Naïve estimation would be $O(i^2)$ traversed edges (i faces, i lines per face, i^2 edges)
- According to the Zone theorem, it is $O(i)$ edges only!

Zone theorem

= given an arrangement $A(L)$ of n lines in the plane and given any line l in the plane, the total number of edges in all the cells of the zone $Z_A(L)$ is at most $6n$.

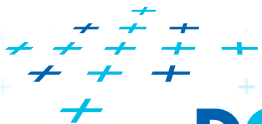
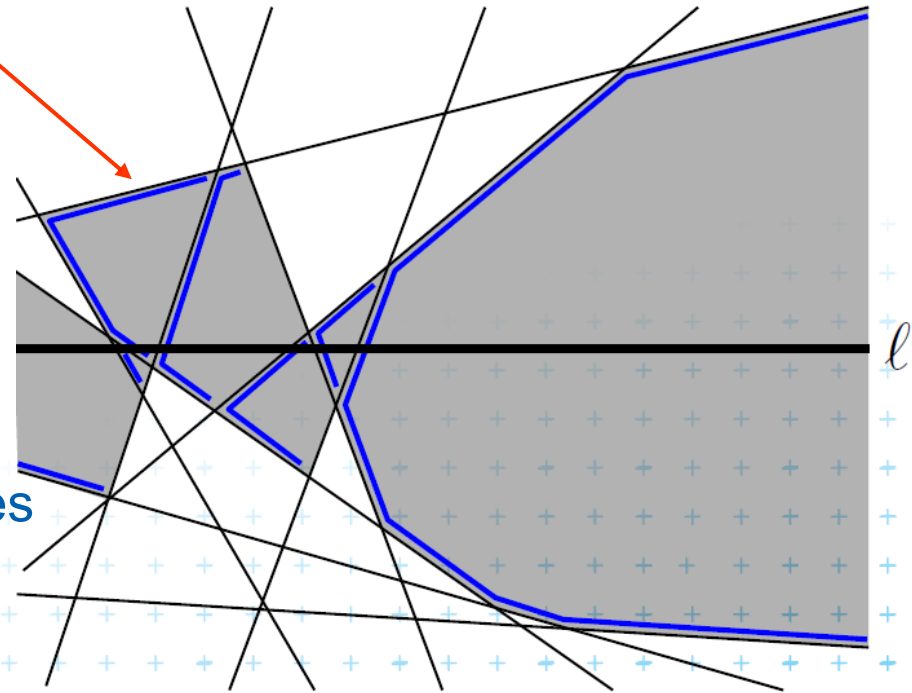


Key idea of a proof

[Mount 2014, page 75]

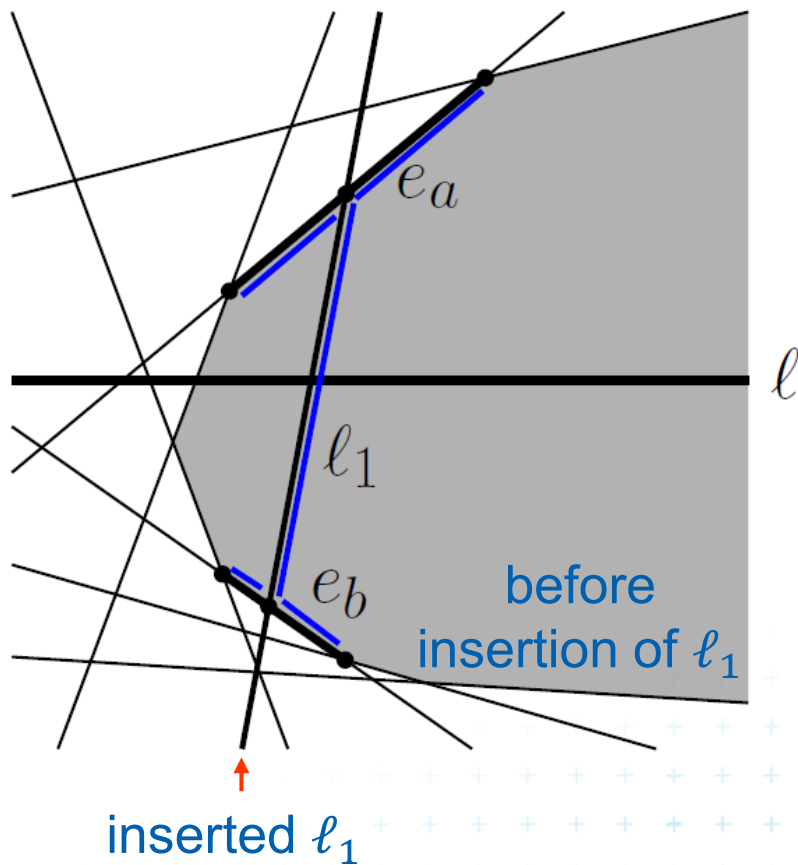
- Find a way to add up edges so that each line will induce a constant number of edges
- Split $6n$ edges of the zone into
 - $3n$ left bounding edges
 - $3n$ right bounding edges
 - $6n$ bounding edges total

⇒ the Zone has linear number of edges
Checking some of them takes $O(i)$



The proof ($3n$ left bounding edges in the zone) [Mount 2014, p. 75]

1) $n = 1$, one left bounding edge, $1 \leq 3 = 3n$ / ℓ



2) True for $n - 1$ lines
 \Rightarrow holds for n lines

ℓ_1 = rightmost line intersecting ℓ

Without line ℓ_1

$3(n - 1)$ left bounding edges

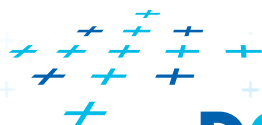
Insert ℓ_1

+1 left bounding edge ℓ_1

+2 split e_a and e_b

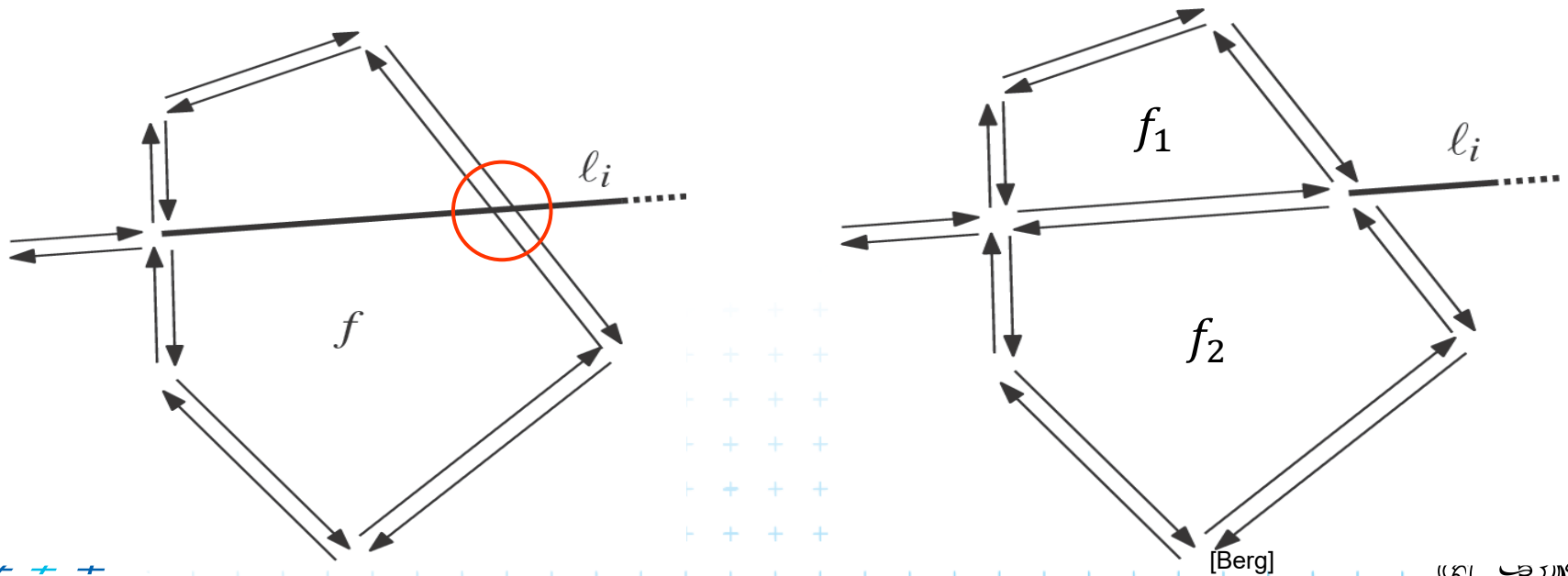
$3(n - 1) + 3 = 3n \Rightarrow$ hold

or less if right bounding edges



Cell split in $O(1)$

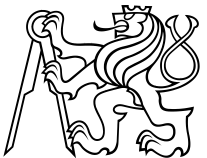
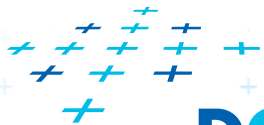
- 1 new vertex
- 2 new face records, 1 face record (f) destroyed
- 3x2 new half-edges, 2 half-edges destroyed
- update pointers ... $O(1)$



Complexity of incremental algorithm

- n insertions
- $O(i) = O(n)$ time for one line insertion instead of $O(i^2)$ (Zone theorem)

=> Complexity: $O(n^2)$ + $n O(i) = O(n^2)$
bbox edges walked



B. Topological plane sweep algorithm

- Complete arrangement needs $O(n^2)$ storage
- Often we need just to **process each arrangement element just once** – and we can throw it out then
- **Classical Sweep line algorithm** (for arrangement of lines)
 - needs $O(n)$ storage (SL status)
 - needs $\log n$ for **heap** manipulation in $O(n^2)$ event points
 - => $O(n^2 \log n)$ algorithm
- **Topological sweep line - TSL**
 - no $O(\log n)$ factor in time complexity in $O(n^2)$ event points
 - **array** of n neighbors and a **stack of ready vertices** $O(1)$
 - => $O(n^2)$ algorithm

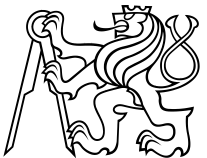
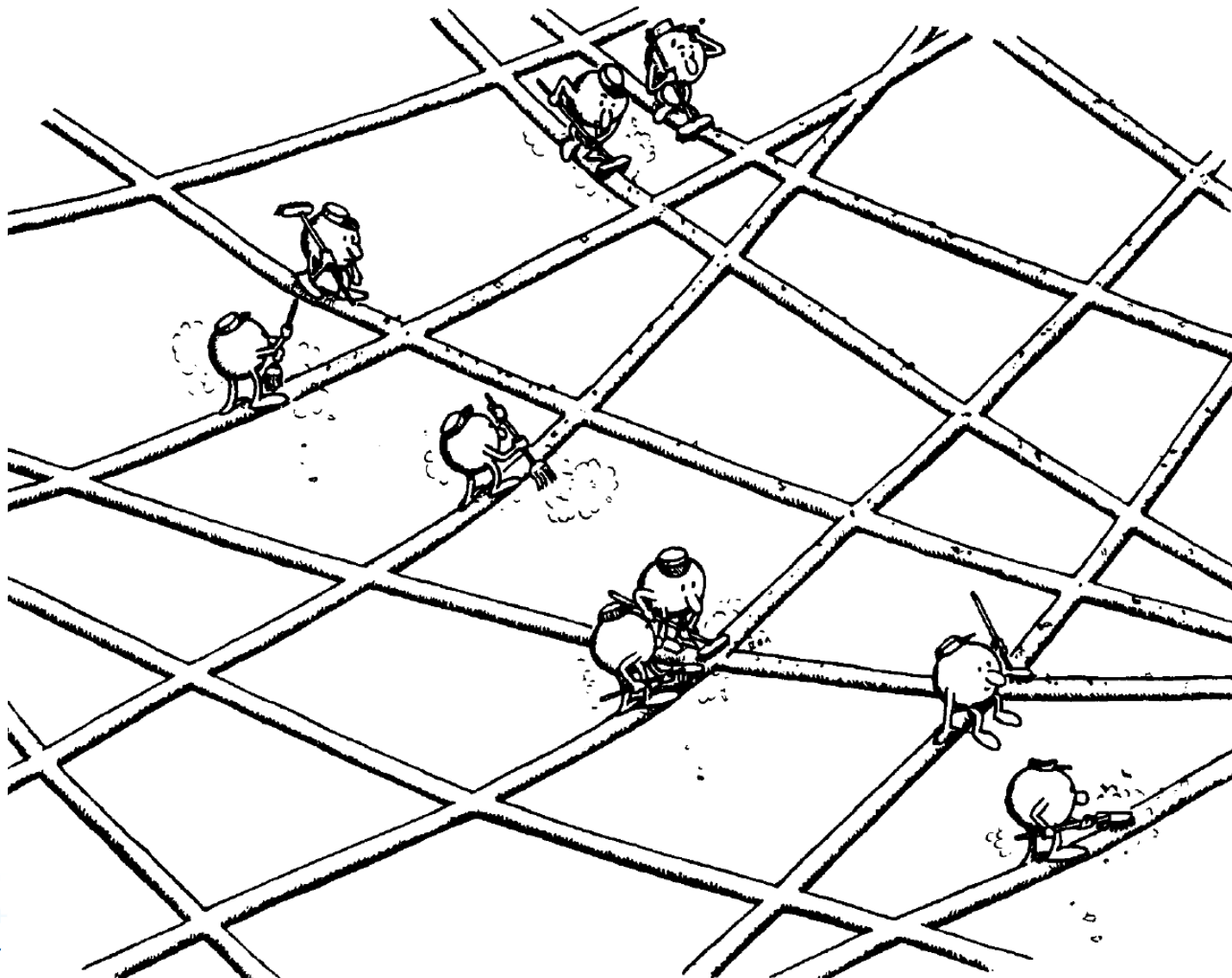


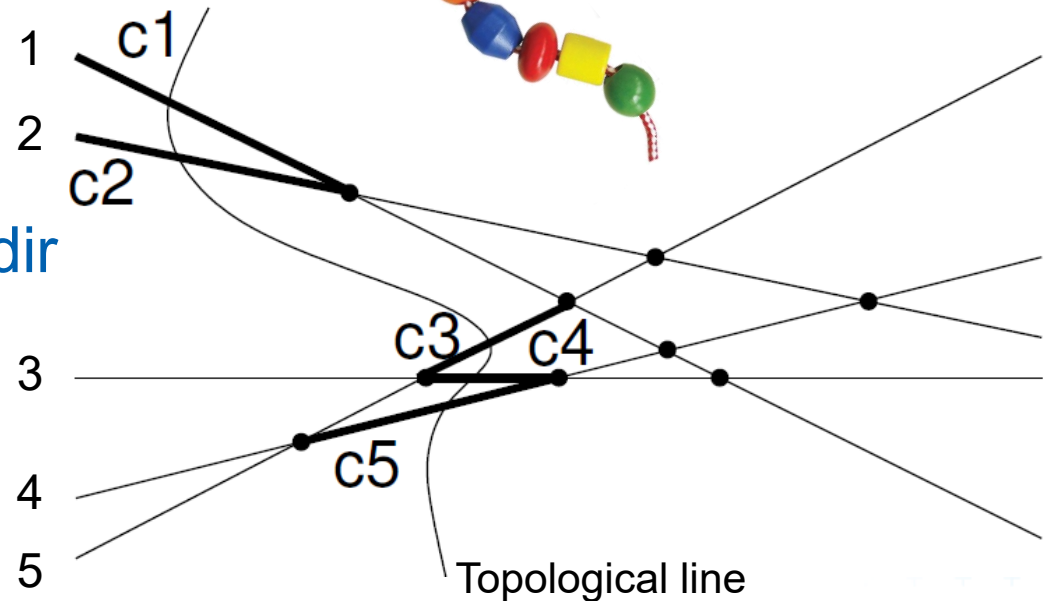
Illustration from Edelsbrunner & Guibas



Topological line and cut

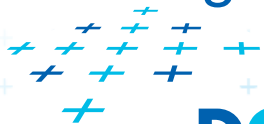
Topological line (curve) (an intuitive notion)

- Monotonic curve in y -dir
- intersects each line exactly once
(as a sweep line)



Cut in an arrangement A

- is an ordered sequence of edges c_1, c_2, \dots, c_n in A (one taken from each line), such that for $1 \leq i \leq n - 1$, c_i and c_{i+1} are incident to the same face of A and c_i is above and c_{i+1} below the face
- Edges in the cut are not necessarily connected (as c_2 and c_3)



Topological plane sweep algorithm

- Starts at the **leftmost cut**

- Consist of left-unbounded edges of A (ending at $-\infty$)
- Computed in $O(n \log n)$ time – order of slopes

- The sweep line is

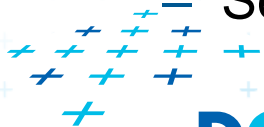
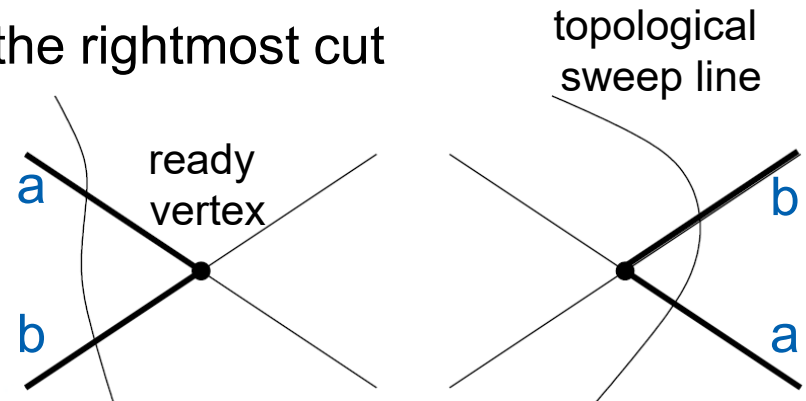
- pushed from the leftmost cut to the rightmost cut
- Advances in elementary steps

- **Elementary step**

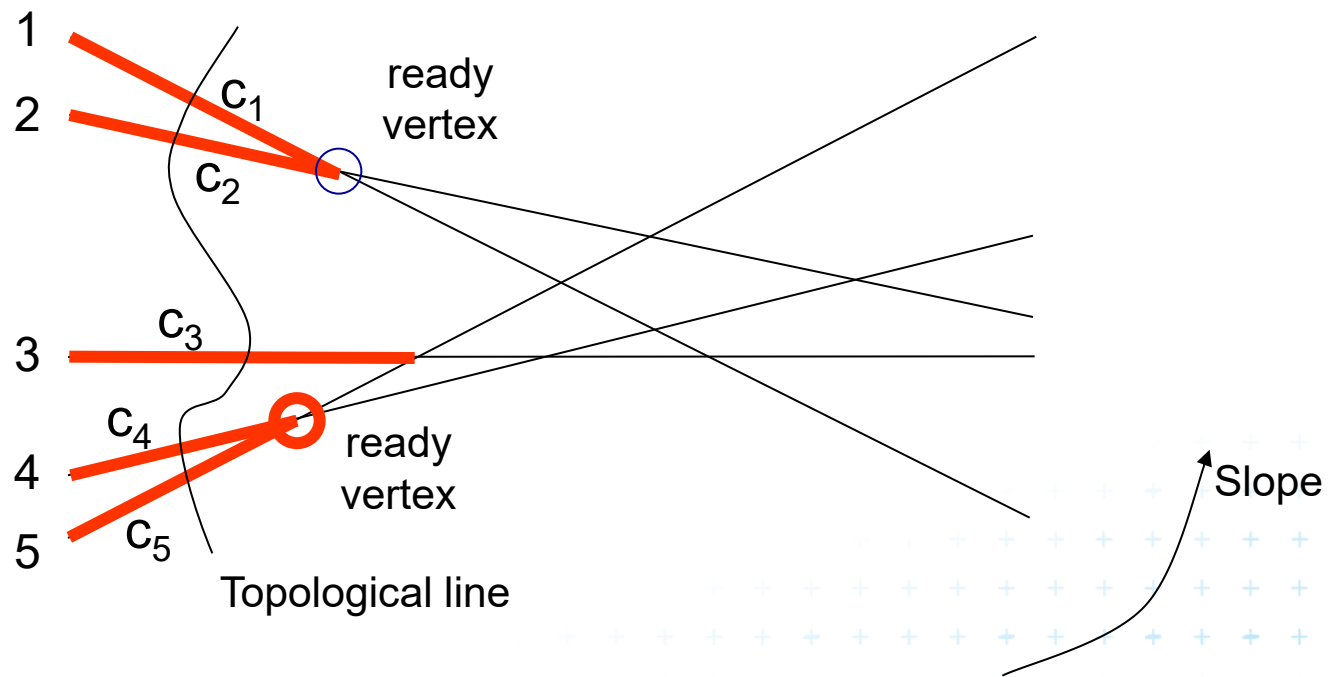
= Processing of any *ready vertex*

(intersection of consecutive edges at their right-point)

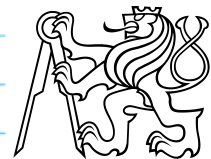
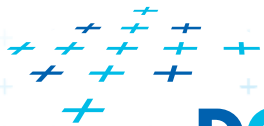
- Swaps the order of lines along the sweep line
- Is always possible (e.g., the point with smallest x)
- Searching of smallest x would need $O(\log n)$ time ...



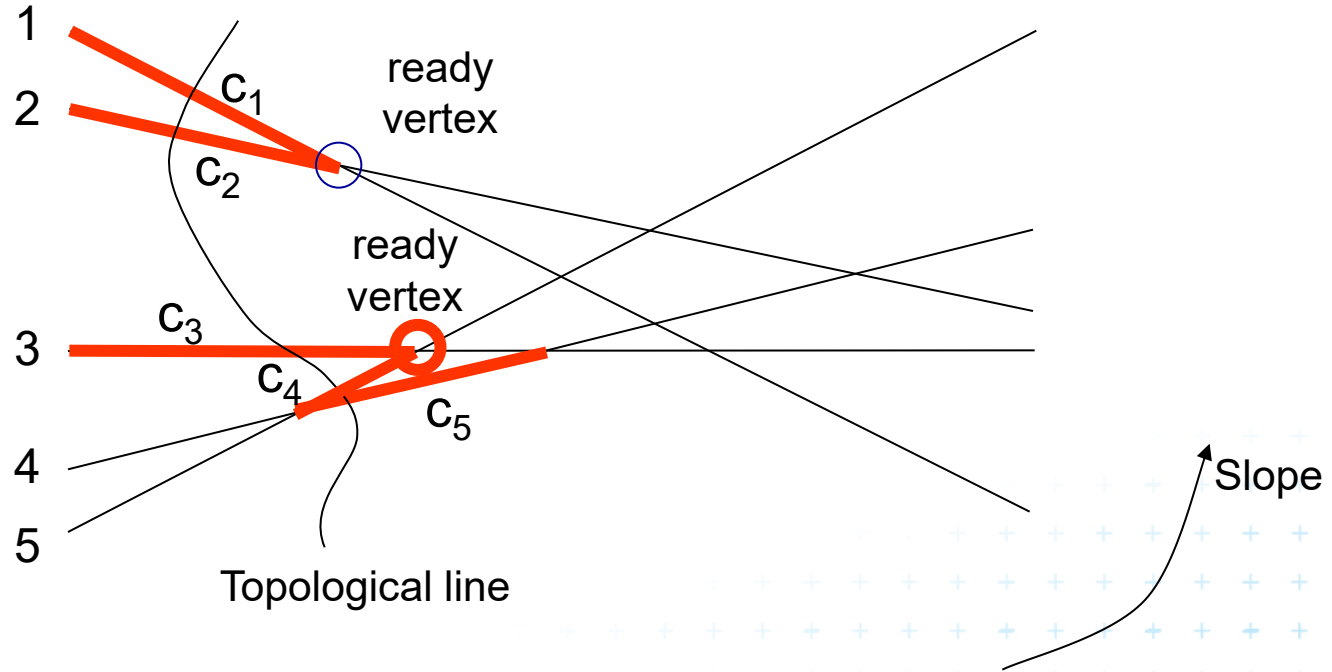
Step 0 – the leftmost cut



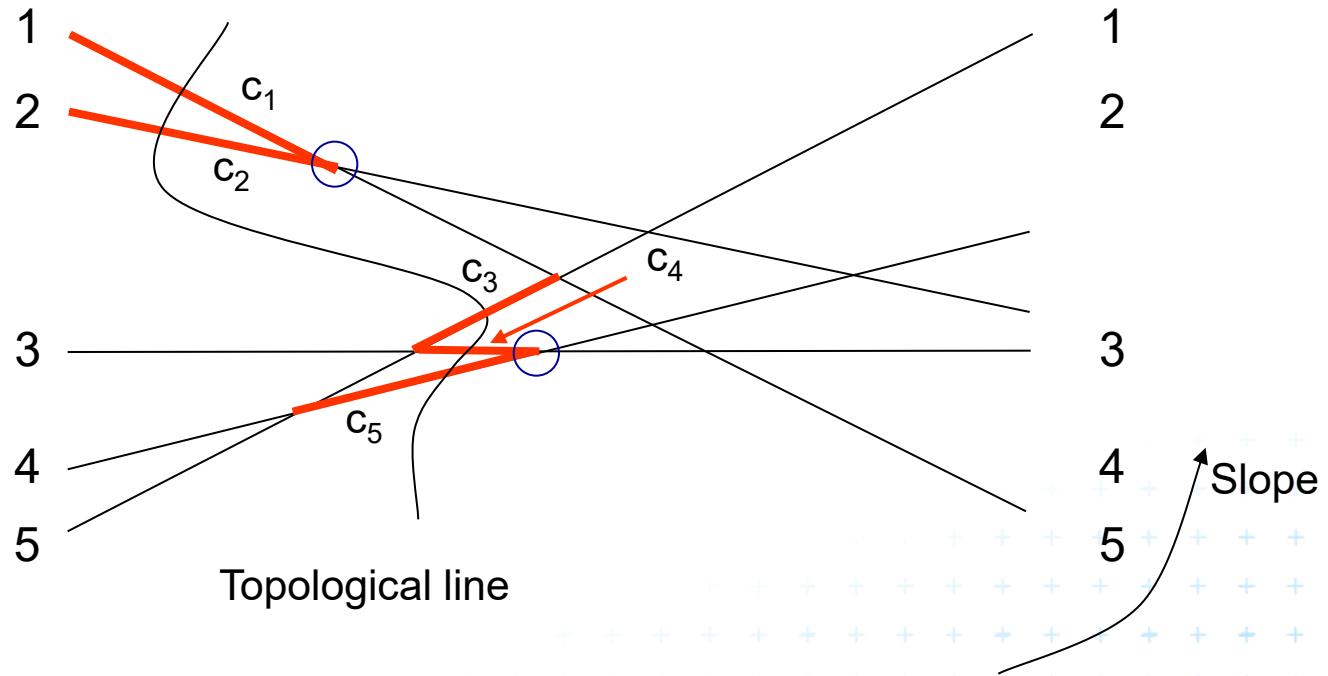
c_i = ordered sequence of edges along the topological sweep line



Step 1 – after processing of $c_4 \times c_5$



Step 2 – after processing of c3 x c4



How to determine the next right point?

- **Elementary step** (intersection at edges right-point)
 - Is always possible (e.g., the point with smallest x)
 - But searching the smallest x would need $O(\log n)$ time
 - We need $O(1)$ time

- **Right endpoint** of the edge in the cut results from ^{UHT} a line of *smaller slope* intersecting it *from above* (traced from L to R) or

^{LHT} line of *larger slope* intersecting it *from below*.



- **Use Upper and Lower Horizon Trees (UHT, LHT)**

- Common segments of UHT and LHT belong to the cut
- Intersect the trees, find pairs of consecutive edges

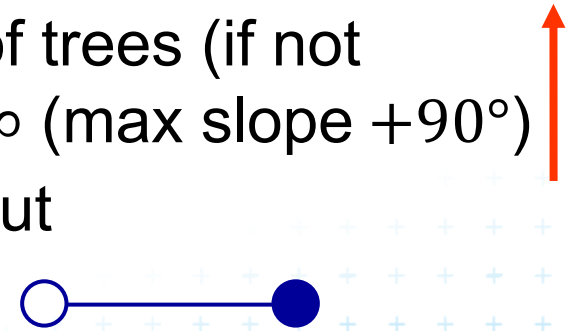
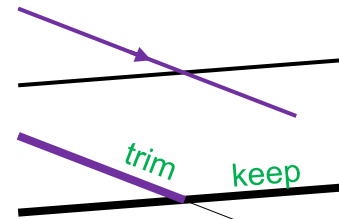
use the right points as legal steps (push to stack)



Upper and lower horizon tree

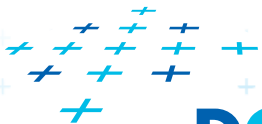
■ Upper horizon tree (UHT)

- Insert lines in order of **decreasing** slope (cw)
- When two edges meet, **keep the edge with higher slope and trim the inserted edge (with lower slope)**
- To get one tree and not the forest of trees (if not connected) add a vertical line in $+\infty$ (max slope $+90^\circ$)
- **Left endpoints** of the edges in the cut do not belong to the tree



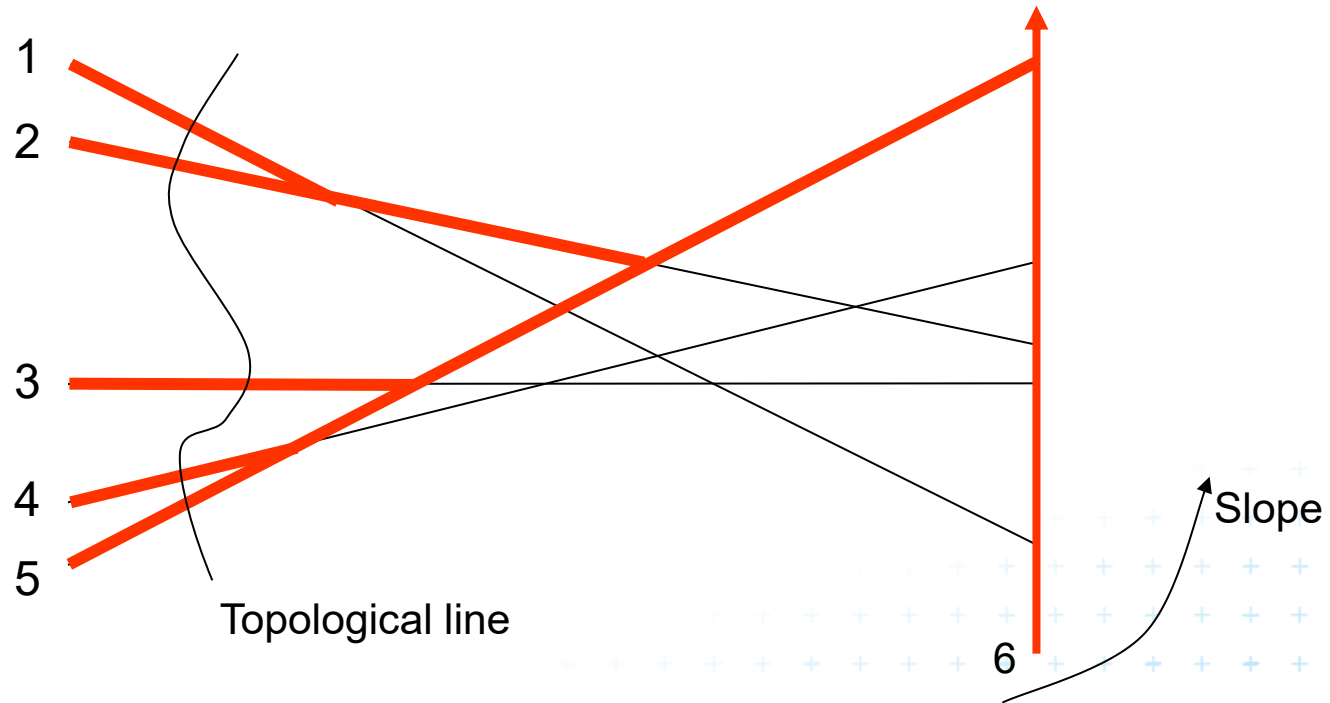
■ Lower horizon tree (LHT) construction is symmetrical

■ UHT and LHT serve for right endpoints determination

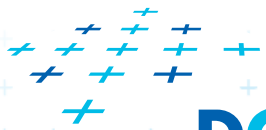
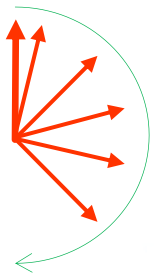


Upper horizon tree (UHT) – initial tree

Insert lines in order of **decreasing slope** (“cw”) 



Insertion order: 6, 5, 4, 3, 2, 1

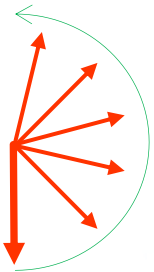
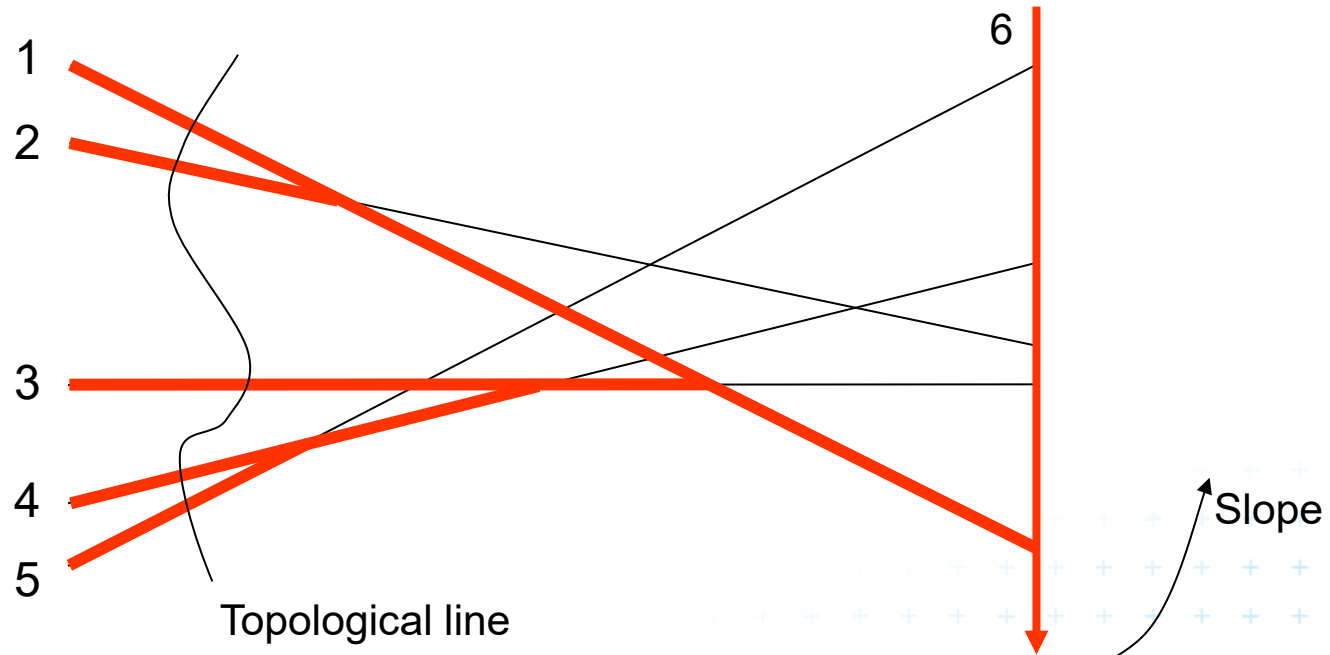


DCGI

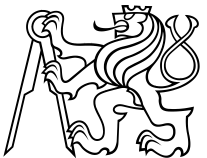
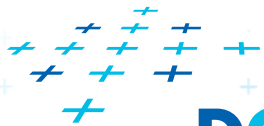


Lower horizon tree (LHT) – initial tree

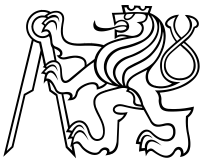
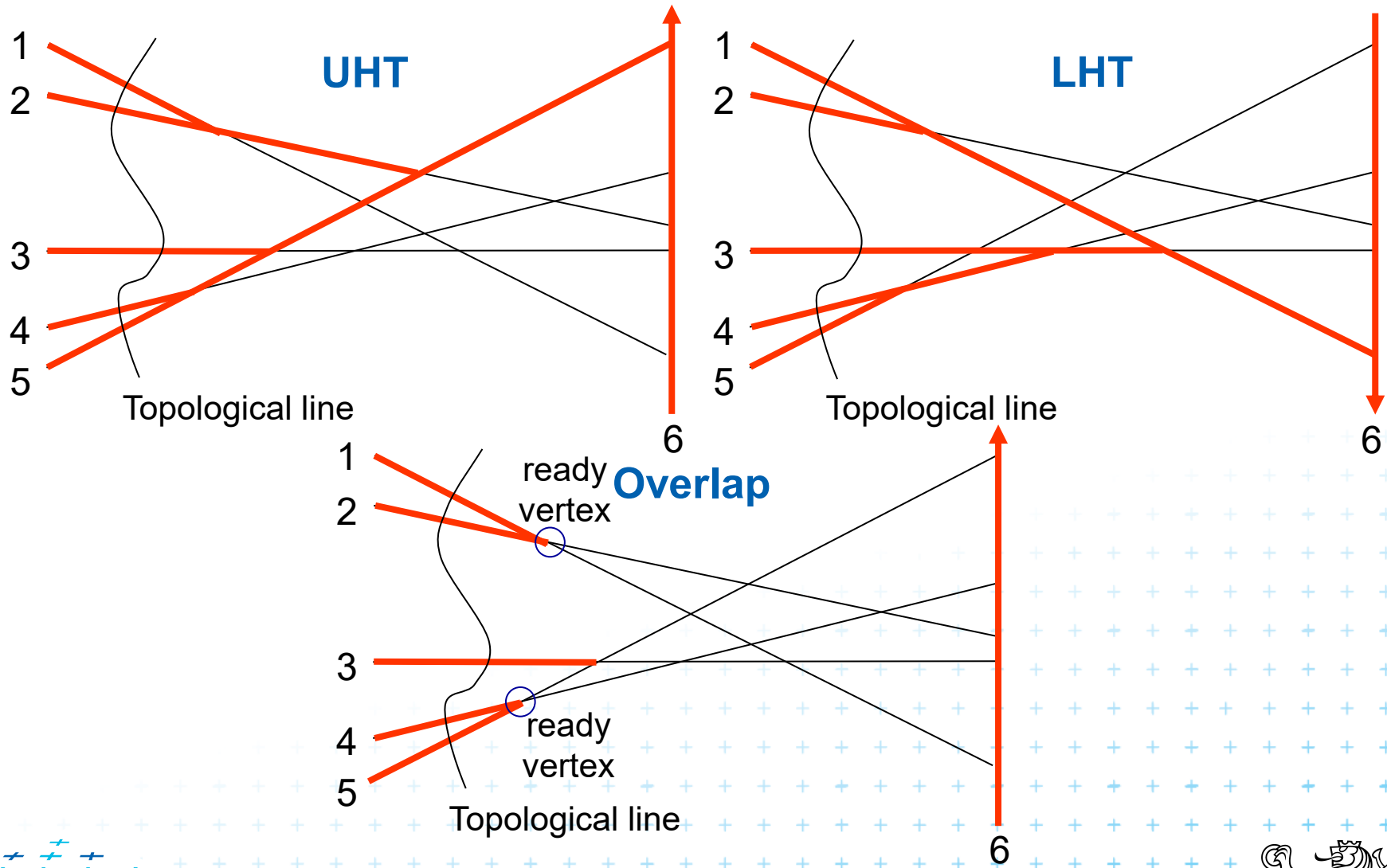
Insert lines in order of **increasing slope** (“ccw”) 



Insertion order: 6, 1, 2, 3, 4, 5

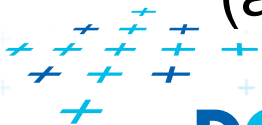
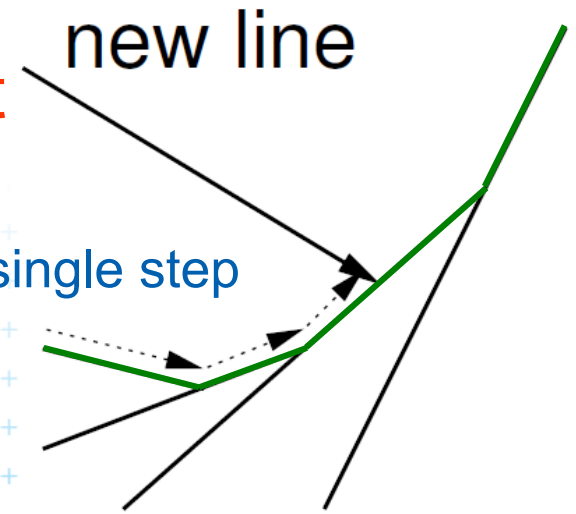


Overlap UHT and LHT – detect ready vertices



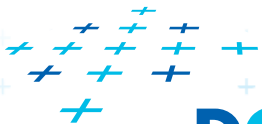
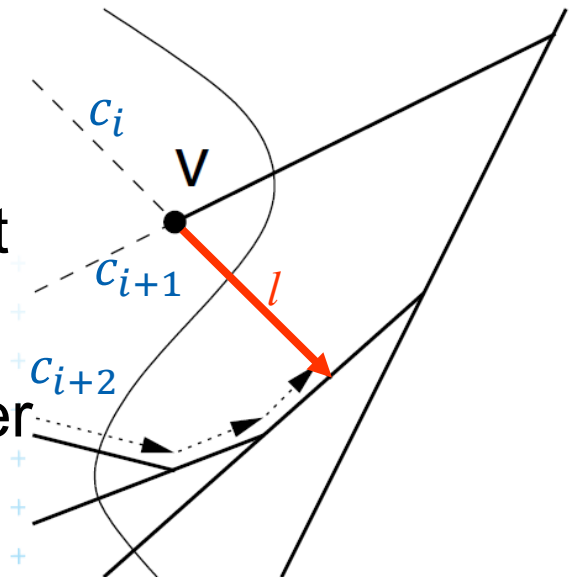
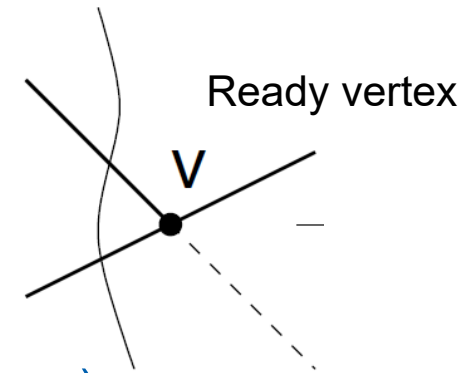
Upper horizon tree (UHT) – init. construction

- Insert lines in order of **decreasing slope** (cw)
- Each new line starts above all the current lines
- The uppermost face = convex polygonal chain
- Walk left to right along the chain to determine the intersection
- **Never walk twice over a segment**
 - Such segment is no longer part of the upper chain $O(1)$ amortized for a single step
 - $O(n)$ segments in UHT $\Rightarrow O(n)$ initial construction
(after $n \log n$ sorting of the lines \sim slope)



Upper horizon tree (UHT) – update

- After the elementary step
- Two edges swap position along the sweep line
- Lower edge l (lower slope, comes from above)
 - Reenter to UHT
 - Terminate at nearest edge of UHT
 - Start in edge below in the current cut
 - Traverse the face in CCW order
 - Intersection must exist, as l has lower slope than the other edge from v and both belong to the same face

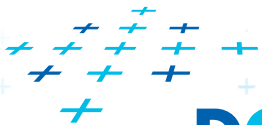


Data structures for topological sweep alg.

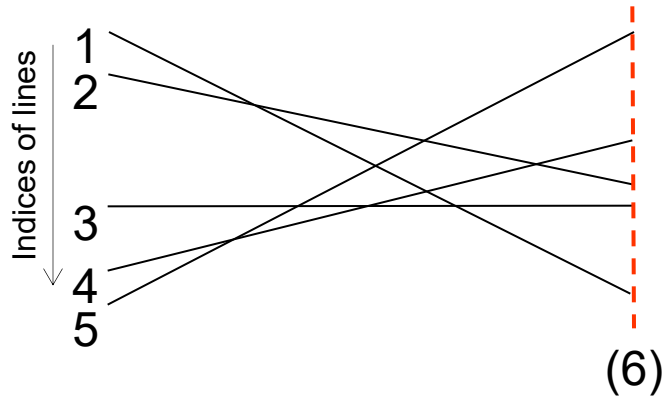
Topological sweep line algorithm uses 5 arrays:

- 1) Line equation coefficients – $E [1:n]$
- 2) Upper horizon tree – UHT $[1:n]$
- 3) Lower horizon tree – LHT $[1:n]$
- 4) Order of lines cut by the sweep line – $C [1:n]$
- 5) Edges along the sweep line – $N [1:n]$
- 6) Stack for ready vertices (events) – S

(n number of lines)



1) Line equation coefficients $E [1:n]$

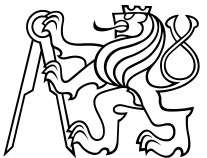
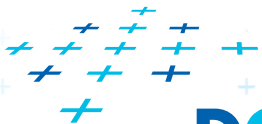
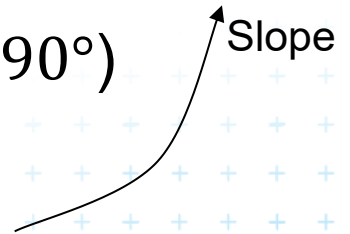


Array of line equations E
 $y = a_i x + b_i$

1	a_1	b_1
2	a_2	b_2
3	a_3	b_3
4	a_4	b_4
5	a_5	b_5

Array of line equation coeffs. E

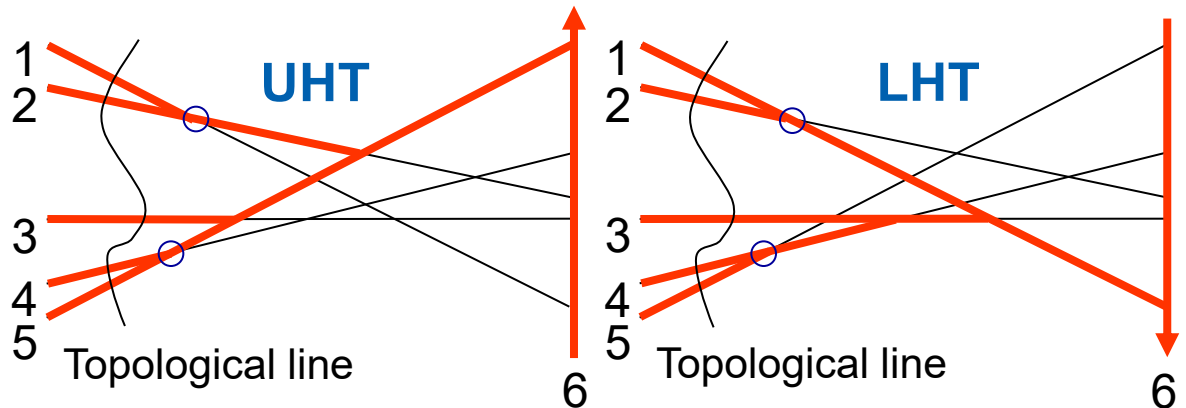
- Contains coefficients a_i and b_i of line equations $y = a_i x + b_i$
- E is indexed by the **line index**
- **Lines are ordered** according to their slope a_i (angle from -90° to 90°)



2) and 3) – Horizon trees UHT and LHT

Their intersection is used for searching of legal steps (right points)

- the shorter edge wins



UHT array
Delimiting lines indices

1	$-\infty$	2
2	$-\infty$	5
3	$-\infty$	5
4	$-\infty$	5
5	$-\infty$	6
6	$-\infty$	$+\infty$

LHT array
Delimiting lines indices

1	$-\infty$	6
2	$-\infty$	1
3	$-\infty$	1
4	$-\infty$	3
5	$-\infty$	4
6	$+\infty$	$-\infty$

■ Store pairs of line indices in E that delimit segment l_i to the left and to the right

■ Segments are half open

■ Unlimited line has “indices”

$(-\infty, +\infty]$ $(+\infty, -\infty]$

■ One additional vertical line

– prevents the tree from splitting into forest of trees

– is inserted first and never trimmed

– is $(-\infty, +\infty]$ for UHT

– is $(+\infty, -\infty]$ for LHT

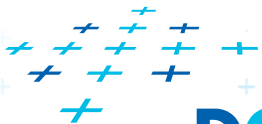


4) Order of lines cut by sweep line – $C [1:n]$

- The topological sweep line cuts each line once
- Order of the cuts (along the topological sweep line) is stored in array C as a sequence of line indices
- Array C “points” to the array E of line equations
- For the initial leftmost cut, the order is the same as in E
- Index c_i addresses i^{th} line from top along the sweep line

CUT Lines C
Indices of supporting lines

c_1	1
c_2	2
c_3	3
c_4	4
c_5	5



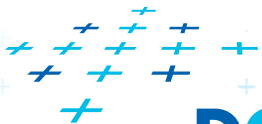
5) Edges along the sweep line – $N [1:n]$

- Edges intersected by the topological sweep line are stored here (edges along the sweep line)
- Instead of endpoints themselves, we store the **indices of lines whose intersections delimit the edge**
- Order of these edges is the same as in \mathcal{C} (both use the index c_i)
- Index c_i stores the index of i -th edge from top along the sweep line

CUT edges N
Pairs of line indices
delimiting the edge

c_1	$-\infty$	2
c_2	$-\infty$	1
c_3	$-\infty$	5
c_4	$-\infty$	5
c_5	$-\infty$	4

The first edge along the sweep line:
- lies on the line index $\mathcal{C}[c_1]$ (line 1 on previous slide)
- Comes from infinity
- is delimited by edge $E[2]$



6) Stack S

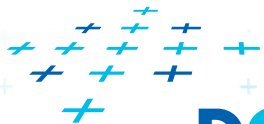
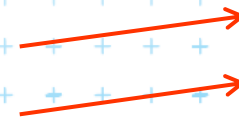
- The exact order of events is not important
(event = intersection in ready vertex)
- Alg. can process any “ready vertex”
- **Event queue** is therefore **replaced by a stack**
(faster: $O(1)$ instead of $O(\log n)$ for queue)
- The stack stores just the **upper edge c_i**
from the pair intersecting in ready vertex
- Intersection in the ready vertex
is computed between stored c_i and c_{i+1}

Stack S
Ready vertex
first edge idx

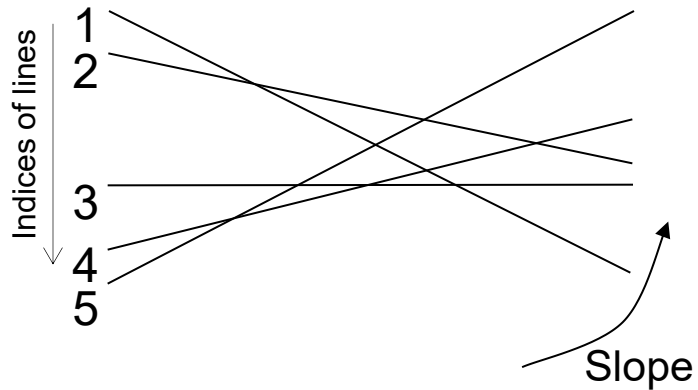


$c4 \times c5$

$c1 \times c2$



Topological sweep line demo



Input

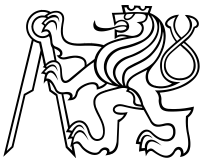
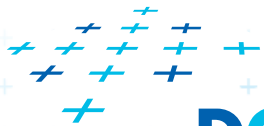
- set of lines L in the plane
- ordered in increasing slope ($\angle -90^\circ$ to 90°), simple, not vertical
- line parameters in array E

Array of line equations E

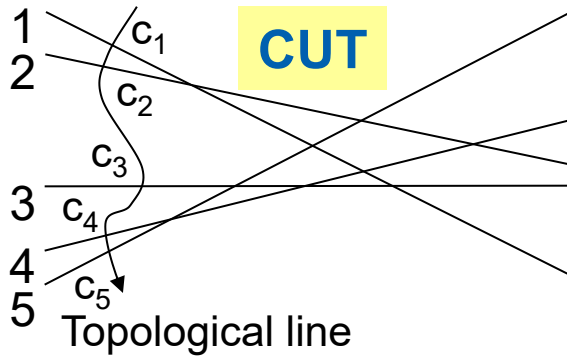
$$y = a_i x + b$$

1	a_1	b_1
2	a_2	b_2
3	a_3	b_3
4	a_4	b_4
5	a_5	b_5

Indices of lines



1) Initial leftmost cut - C



- Store the indices of lines in E into the **Cut lines array C** in increasing slope order

Array of line equations E
 $y = a_i x + b$

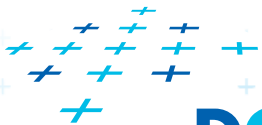
1	a_1	b_1
2	a_2	b_2
3	a_3	b_3
4	a_4	b_4
5	a_5	b_5

Indices of lines

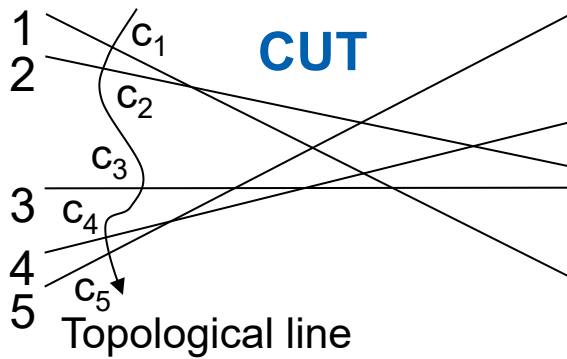
CUT Lines C
 Indices of supporting lines

C_1	1
C_2	2
C_3	3
C_4	4
C_5	5

Line indices along the cut



1) Initial leftmost cut - N



- Prepare **array N** for endpoints of the cut edges (resp. for line indices delimiting these edges)
- Init it by line “ends” $-\infty, +\infty$

Array of line equations E
 $y = a_i x + b$

1	a_1	b_1
2	a_2	b_2
3	a_3	b_3
4	a_4	b_4
5	a_5	b_5

indices of lines

CUT edges N
 Pairs of line indices delimiting the edge

c_1	$-\infty$	∞
c_2	$-\infty$	∞
c_3	$-\infty$	∞
c_4	$-\infty$	∞
c_5	$-\infty$	∞

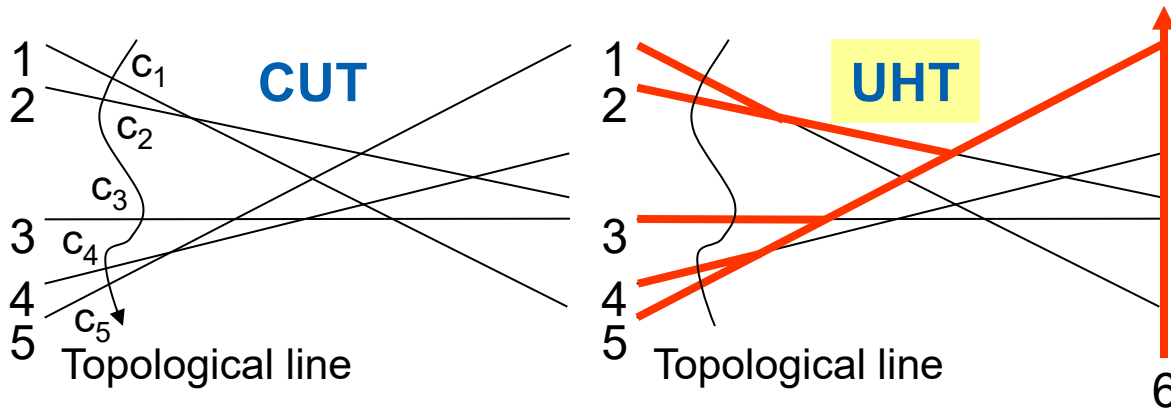
Index of delimiter edge in $-\infty$

CUT Lines C
 Indices of supporting lines

c_1	1
c_2	2
c_3	3
c_4	4
c_5	5



2a) Compute Upper Horizon Tree - UHT



Array of line equations E
 $y = a_i x + b$

1	a_1	b_1
2	a_2	b_2
3	a_3	b_3
4	a_4	b_4
5	a_5	b_5

UHT array
 Delimiting lines indices

1	$-\infty$	2
2	$-\infty$	5
3	$-\infty$	5
4	$-\infty$	5
5	$-\infty$	6
6	$-\infty$	$+\infty$

Order of insertion into UHT

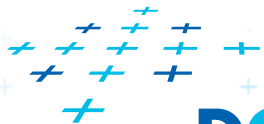
CUT edges N
 Pairs of line indices delimiting the edge

C_1	$-\infty$	∞
C_2	$-\infty$	∞
C_3	$-\infty$	∞
C_4	$-\infty$	∞
C_5	$-\infty$	∞

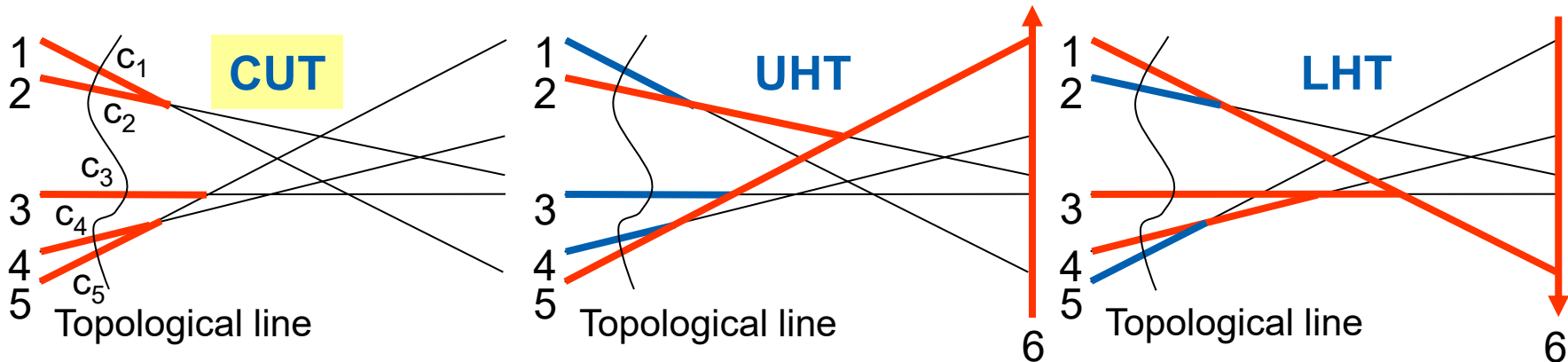
CUT Lines C
 Indices of supporting lines

C_1	1
C_2	2
C_3	3
C_4	4
C_5	5

Additional "help edge"
 Unlimited, bottom-up
 Inserted first, never changed



3a) Determine right delimiters of edges - N



Array of line equations E
 $y = a_i x + b_i$

1	a_1	b_1
2	a_2	b_2
3	a_3	b_3
4	a_4	b_4
5	a_5	b_5

UHT array
 Delimiting lines indices

1	$-\infty$	2
2	$-\infty$	5
3	$-\infty$	5
4	$-\infty$	5
5	$-\infty$	6
6	$-\infty$	$+\infty$

LHT array
 Delimiting lines indices

1	$-\infty$	6
2	$-\infty$	1
3	$-\infty$	1
4	$-\infty$	3
5	$-\infty$	4
6	$+\infty$	$-\infty$

CUT edges N
 Pairs of line indices delimiting the edge

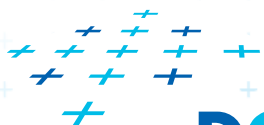
c_1	$-\infty$	2
c_2	$-\infty$	1
c_3	$-\infty$	5
c_4	$-\infty$	5
c_5	$-\infty$	4

CUT Lines C
 Indices of supporting lines

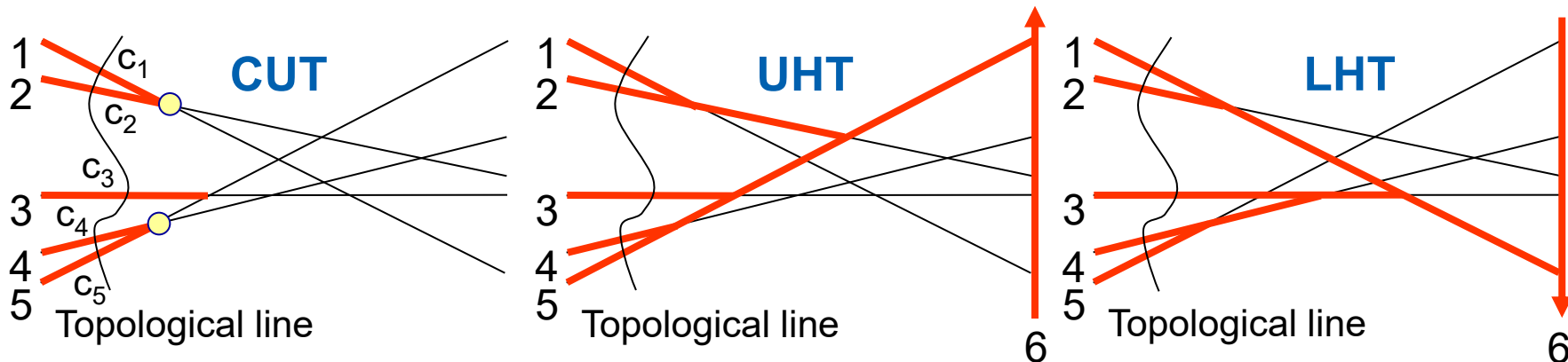
c_1	1
c_2	2
c_3	3
c_4	4
c_5	5

Stack S
 Ready vertex first edge idx

Intersect the trees – take the shorter edge
 = edge with smaller x coordinate of R endpoint



3b) Ready vertices = inters. of neighbors – S



Array of line equations E
 $y = a_i x + b$

1	a_1	b_1
2	a_2	b_2
3	a_3	b_3
4	a_4	b_4
5	a_5	b_5

UHT array
Delimiting lines indices

1	$-\infty$	2
2	$-\infty$	5
3	$-\infty$	5
4	$-\infty$	5
5	$-\infty$	6
6	$-\infty$	$+\infty$

LHT array
Delimiting lines indices

1	$-\infty$	6
2	$-\infty$	1
3	$-\infty$	1
4	$-\infty$	3
5	$-\infty$	4
6	$+\infty$	$-\infty$

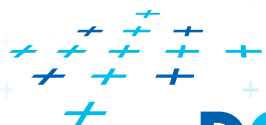
CUT edges N
Pairs of line indices delimiting the edge

c_1	$-\infty$	2
c_2	$-\infty$	1
c_3	$-\infty$	5
c_4	$-\infty$	5
c_5	$-\infty$	4

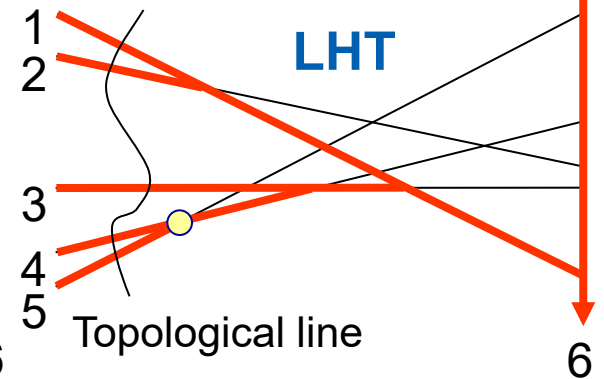
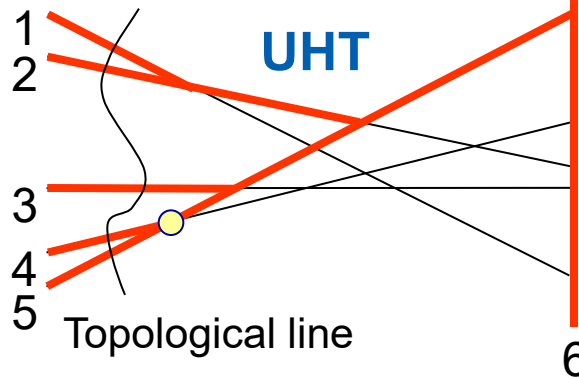
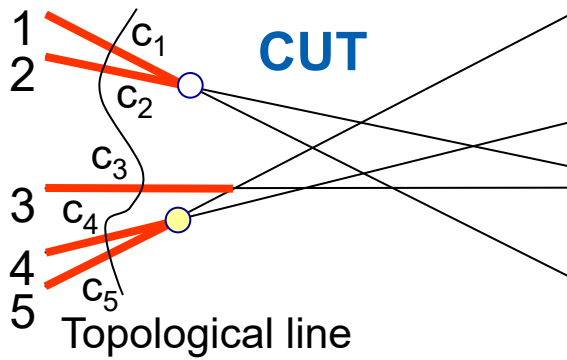
CUT Lines C
Indices of supporting lines

c_1	1
c_2	2
c_3	3
c_4	4
c_5	5

Stack S
Ready vertex first edge idx



4a) Pop ready vertex from S – process c4



Array of line equations E
 $y = a_i x + b$

UHT array
 Delimiting lines indices

LHT array
 Delimiting lines indices

CUT edges N
 Pairs of line indices delimiting the edge

CUT Lines C
 Indices of supporting lines

Stack S
 Ready vertex first edge idx

1	a_1	b_1
2	a_2	b_2
3	a_3	b_3
4	a_4	b_4
5	a_5	b_5

1	$-\infty$	2
2	$-\infty$	5
3	$-\infty$	5
4	$-\infty$	5
5	$-\infty$	6
6	$-\infty$	$+\infty$

1	$-\infty$	6
2	$-\infty$	1
3	$-\infty$	1
4	$-\infty$	3
5	$-\infty$	4
6	$+\infty$	$-\infty$

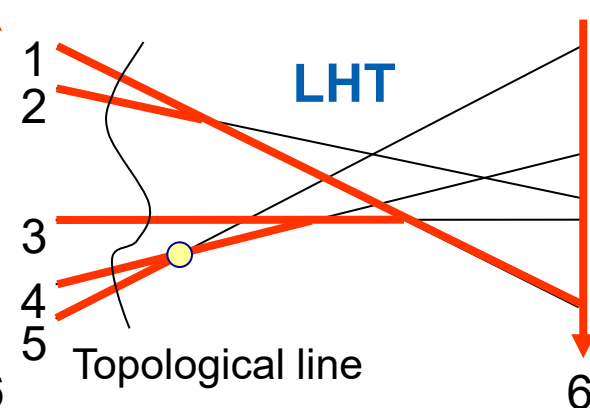
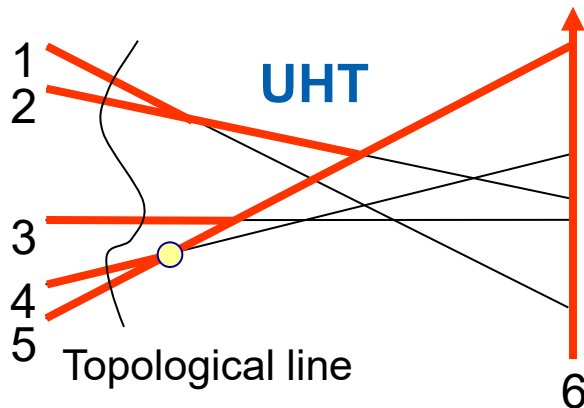
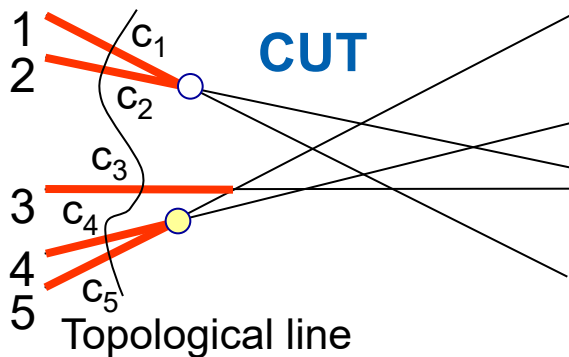
c_1	$-\infty$	2
c_2	$-\infty$	1
c_3	$-\infty$	5
c_4	$-\infty$	5
c_5	$-\infty$	4

c_1	1
c_2	2
c_3	3
c_4	4
c_5	5

c_4
c_1



4b) Swap lines c4 and c5 – swap 4 and 5



Array of line equations E
 $y = a_i x + b$

1	a_1	b_1
2	a_2	b_2
3	a_3	b_3
4	a_4	b_4
5	a_5	b_5

UHT array
Delimiting lines indices

1	$-\infty$	2
2	$-\infty$	5
3	$-\infty$	5
4	$-\infty$	5
5	$-\infty$	6
6	$-\infty$	$+\infty$

LHT array
Delimiting lines indices

1	$-\infty$	6
2	$-\infty$	1
3	$-\infty$	1
4	$-\infty$	3
5	$-\infty$	4
6	$+\infty$	$-\infty$

CUT edges N
Pairs of line indices delimiting the edge

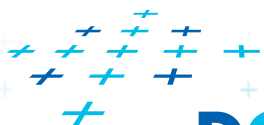
c_1	$-\infty$	2
c_2	$-\infty$	1
c_3	$-\infty$	5
c_4	$-\infty$	4
c_5	$-\infty$	5

CUT Lines C
Indices of supporting lines

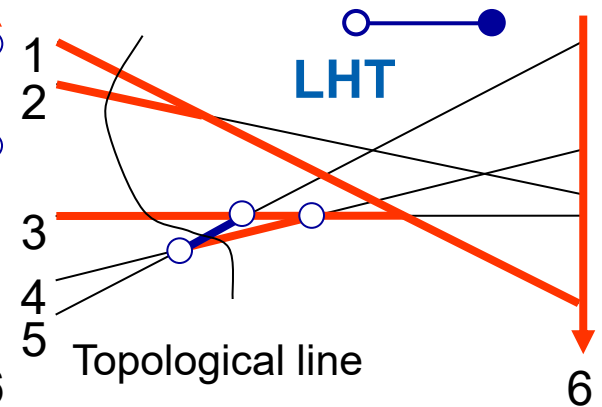
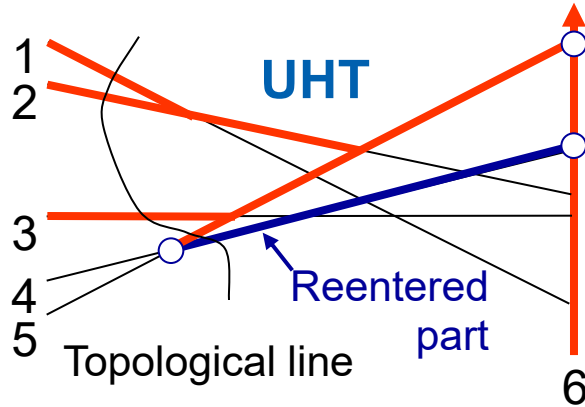
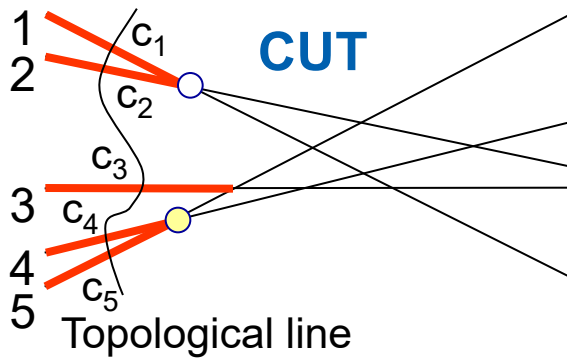
c_1	1
c_2	2
c_3	3
c_4	5
c_5	4

Stack S
Ready vertex first edge idx

c1



4c) Update the horizon trees – UHT and LHT



Array of line equations E
 $y = a_i x + b$

1	a_1	b_1
2	a_2	b_2
3	a_3	b_3
4	a_4	b_4
5	a_5	b_5

UHT array
 Delimiting lines indices

1	$-\infty$	2
2	$-\infty$	5
3	$-\infty$	5
4	5	6
5	4	6
6	$-\infty$	$+\infty$

LHT array
 Delimiting lines indices

1	$-\infty$	6
2	$-\infty$	1
3	$-\infty$	1
4	5	3
5	4	3
6	$+\infty$	$-\infty$

CUT edges N
 Pairs of line indices delimiting the edge

c_1	$-\infty$	2
c_2	$-\infty$	1
c_3	$-\infty$	5
c_4	$-\infty$	4
c_5	$-\infty$	5

CUT Lines C
 Indices of supporting lines

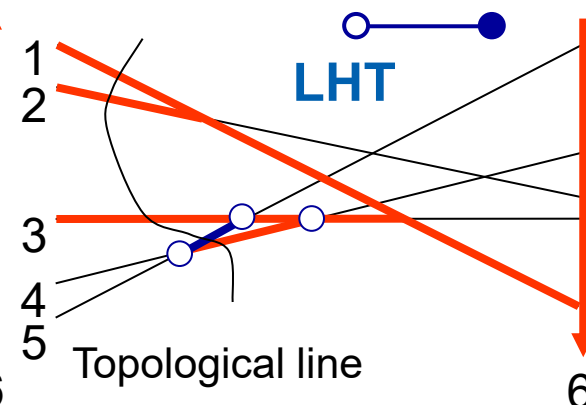
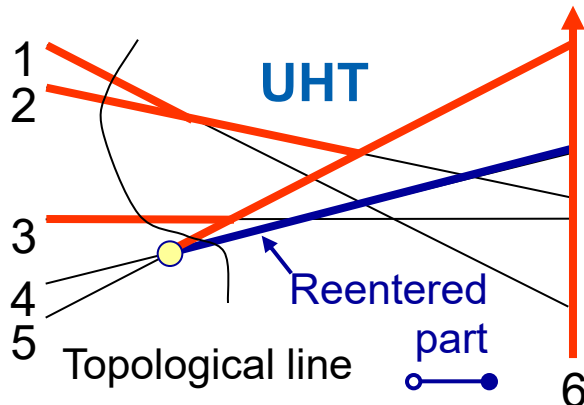
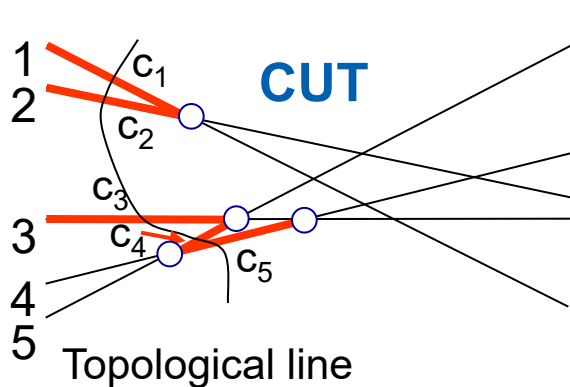
c_1	1
c_2	2
c_3	3
c_4	5
c_5	4

Stack S
 Ready vertex upper edge id

c1

Note: Edges are half open to avoid breaking the tree after reinsertion (edges starting in the ready vertex cannot have a common left endpoint)

4d) Determine new cut edges endpoints – N



Array of line equations E
 $y = a_i x + b$

1	a_1	b_1
2	a_2	b_2
3	a_3	b_3
4	a_4	b_4
5	a_5	b_5

UHT array
 Delimiting lines indices

1	$-\infty$	2
2	$-\infty$	5
3	$-\infty$	5
4	5	6
5	4	6
6	$-\infty$	$+\infty$

LHT array
 Delimiting lines indices

1	$-\infty$	6
2	$-\infty$	1
3	$-\infty$	1
4	5	3
5	4	3
6	$+\infty$	$-\infty$

CUT edges N
 Pairs of line indices delimiting the edge

c_1	$-\infty$	2
c_2	$-\infty$	1
c_3	$-\infty$	5
c_4	4	3
c_5	5	3

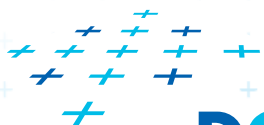
CUT Lines C
 Indices of supporting lines

c_1	1
c_2	2
c_3	3
c_4	5
c_5	4

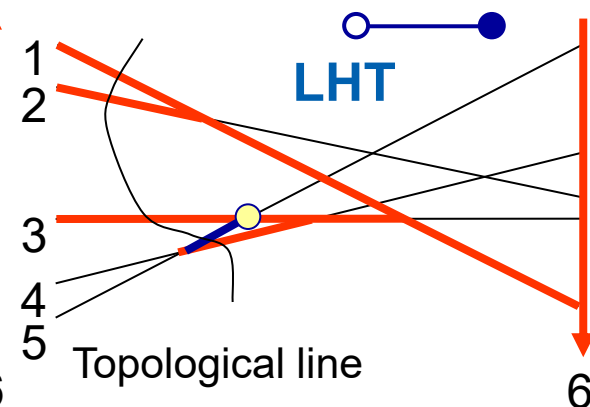
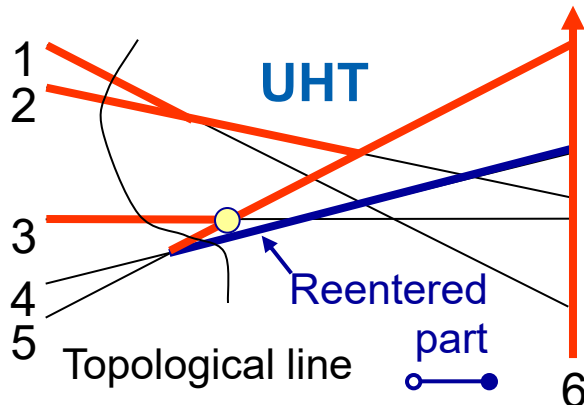
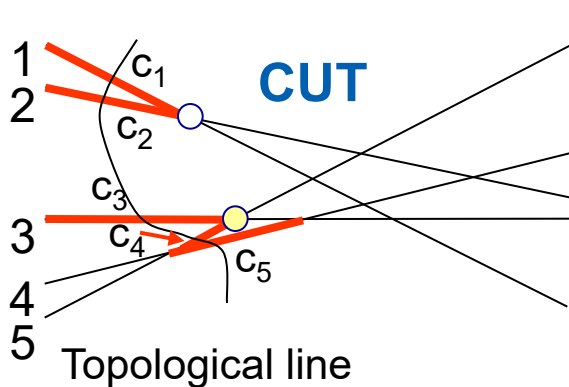
Stack S
 Ready vertex upper edge id

c_1

Intersect the trees – take the shorter edge



4e) Intersect with neighbors – push into S



Array of line equations E
 $y = a_i x + b$

1	a_1	b_1
2	a_2	b_2
3	a_3	b_3
4	a_4	b_4
5	a_5	b_5

UHT array
 Delimiting lines indices

1	$-\infty$	2
2	$-\infty$	5
3	$-\infty$	5
4	5	6
5	4	6
6	$-\infty$	$+\infty$

LHT array
 Delimiting lines indices

1	$-\infty$	6
2	$-\infty$	1
3	$-\infty$	1
4	5	3
5	4	3
6	$+\infty$	$-\infty$

CUT edges N
 Pairs of line indices delimiting the edge

c_1	$-\infty$	2
c_2	$-\infty$	1
c_3	$-\infty$	5
c_4	4	3
c_5	5	3

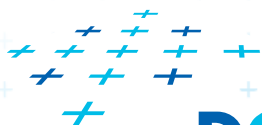
CUT Lines C
 Indices of supporting lines

c_1	1
c_2	2
c_3	3
c_4	5
c_5	4

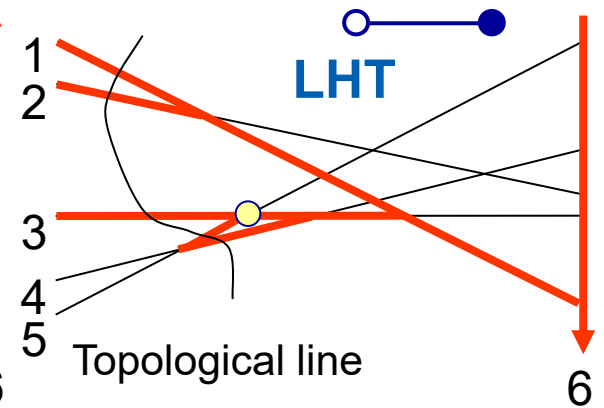
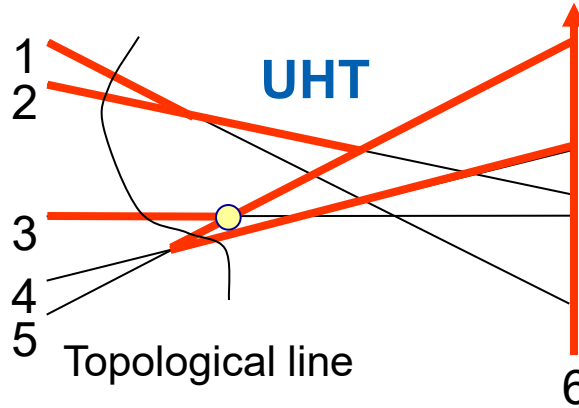
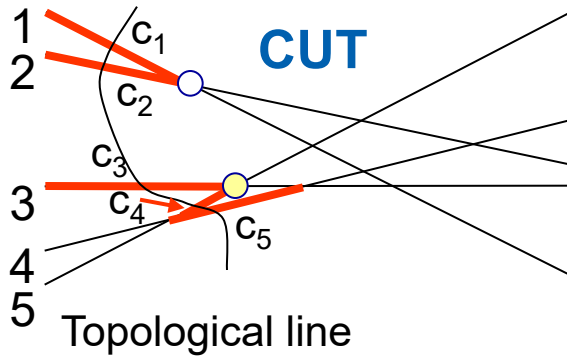
Stack S
 Ready vertex upper edge id



Intersections of neighbors - into stack



4a) Pop ready vertex from S – process c3



Array of line equations E
 $y = a_i x + b$

1	a_1	b_1
2	a_2	b_2
3	a_3	b_3
4	a_4	b_4
5	a_5	b_5

UHT array
 Delimiting lines indices

1	$-\infty$	2
2	$-\infty$	5
3	$-\infty$	5
4	5	6
5	4	6
6	$-\infty$	$+\infty$

LHT array
 Delimiting lines indices

1	$-\infty$	6
2	$-\infty$	1
3	$-\infty$	1
4	5	3
5	4	3
6	$+\infty$	$-\infty$

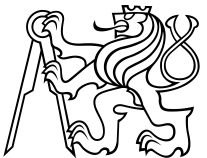
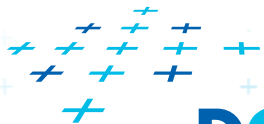
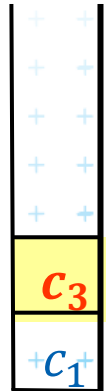
CUT edges N
 Pairs of line indices delimiting the edge

c_1	$-\infty$	2
c_2	$-\infty$	1
c_3	$-\infty$	5
c_4	4	3
c_5	5	3

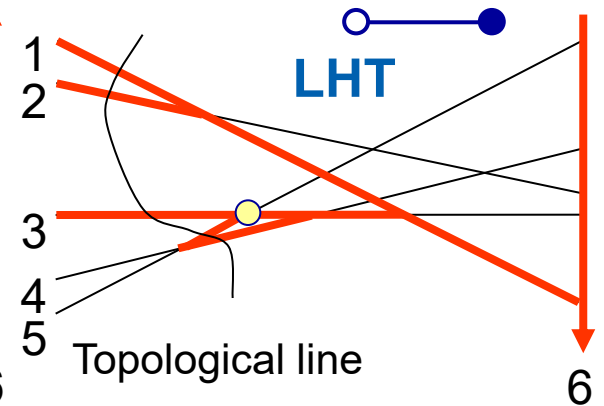
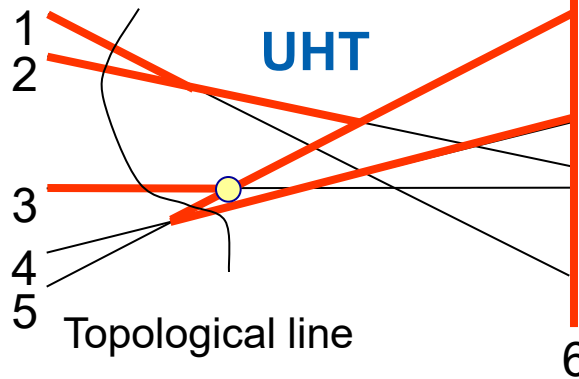
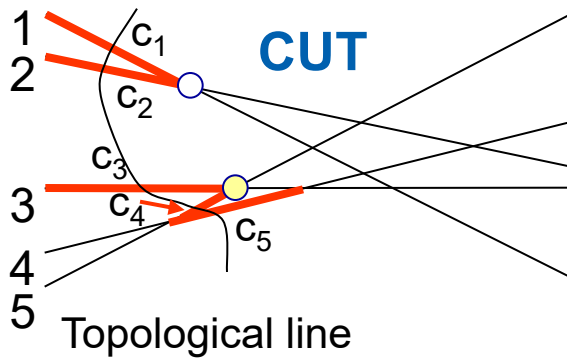
CUT Lines C
 Indices of supporting lines

c_1	1
c_2	2
c_3	3
c_4	5
c_5	4

Stack S
 Ready vertex first edge idx



4b) Swap lines c4 and c5 – swap 4 and 5



Array of line equations E
 $y = a_i x + b$

1	a_1	b_1
2	a_2	b_2
3	a_3	b_3
4	a_4	b_4
5	a_5	b_5

UHT array
 Delimiting lines indices

1	$-\infty$	2
2	$-\infty$	5
3	$-\infty$	5
4	5	6
5	4	6
6	$-\infty$	$+\infty$

LHT array
 Delimiting lines indices

1	$-\infty$	6
2	$-\infty$	1
3	$-\infty$	1
4	5	3
5	4	3
6	$+\infty$	$-\infty$

CUT edges N
 Pairs of line indices delimiting the edge

c_1	$-\infty$	2
c_2	$-\infty$	1
c_3	4	3
c_4	$-\infty$	5
c_5	5	3

Swapped invalidated

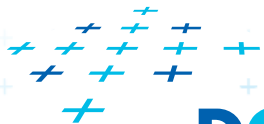
CUT Lines C
 Indices of supporting lines

c_1	1
c_2	2
c_3	5
c_4	3
c_5	4

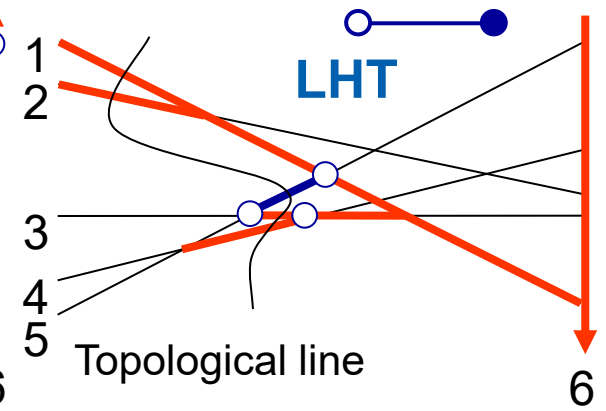
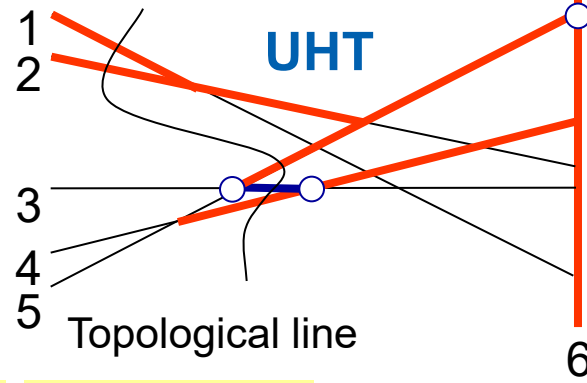
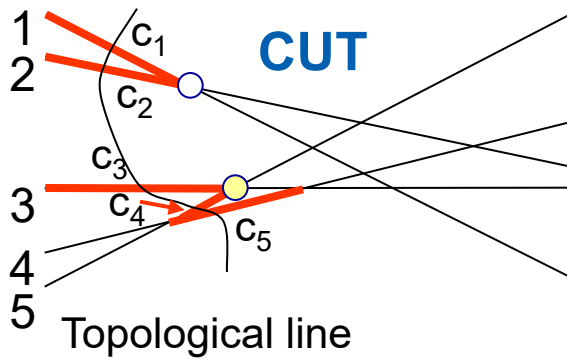
Swapped

Stack S
 Ready vertex first edge idx

c1



4c) Update the horizon trees – UHT and LHT



Array of line equations E
 $y = a_i x + b$

1	a_1	b_1
2	a_2	b_2
3	a_3	b_3
4	a_4	b_4
5	a_5	b_5

UHT array
 Delimiting lines indices

1	$-\infty$	2
2	$-\infty$	5
3	5	4
4	5	6
5	3	6
6	$-\infty$	$+\infty$

LHT array
 Delimiting lines indices

1	$-\infty$	6
2	$-\infty$	1
3	5	1
4	5	3
5	3	1
6	$+\infty$	$-\infty$

CUT edges N
 Pairs of line indices delimiting the edge

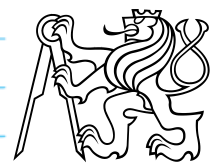
c_1	$-\infty$	2
c_2	$-\infty$	1
c_3	4	3
c_4	$-\infty$	5
c_5	5	3

CUT Lines C
 Indices of supporting lines

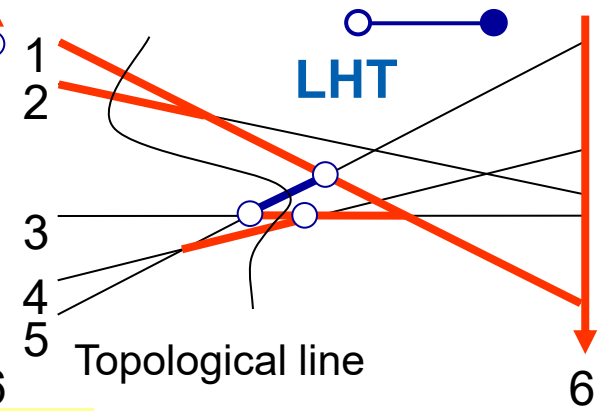
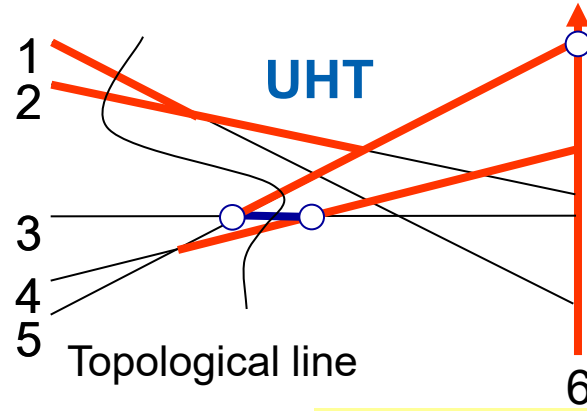
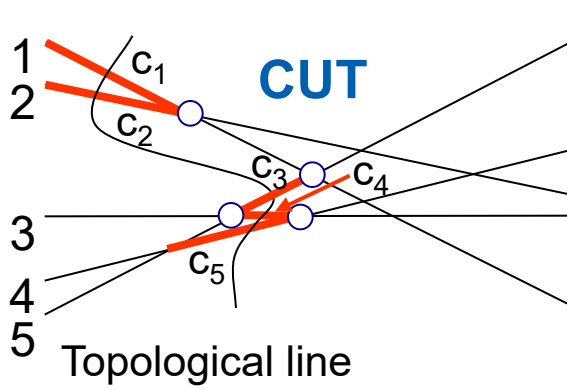
c_1	1
c_2	2
c_3	5
c_4	3
c_5	4

Stack S
 Ready vertex first edge idx

c_1



4d) Determine new cut edges endpoints



Array of line equations E
 $y = a_i x + b$

1	a_1	b_1
2	a_2	b_2
3	a_3	b_3
4	a_4	b_4
5	a_5	b_5

UHT array
 Delimiting lines indices

1	$-\infty$	2
2	$-\infty$	5
3	5	4
4	5	6
5	3	6
6	$-\infty$	$+\infty$

LHT array
 Delimiting lines indices

1	$-\infty$	6
2	$-\infty$	1
3	5	1
4	5	3
5	3	1
6	$+\infty$	$-\infty$

CUT edges N
 Pairs of line indices delimiting the edge

c_1	$-\infty$	2
c_2	$-\infty$	1
c_3	3	1
c_4	5	4
c_5	5	3

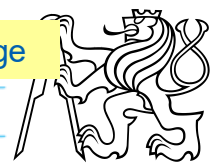
CUT Lines C
 Indices of supporting lines

c_1	1
c_2	2
c_3	5
c_4	3
c_5	4

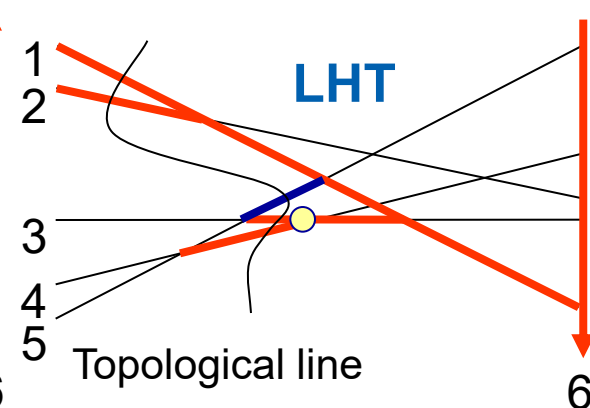
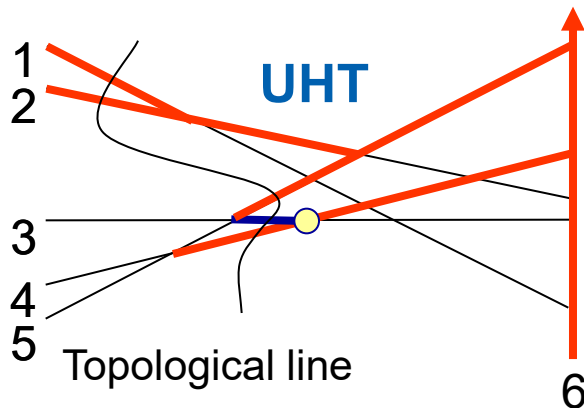
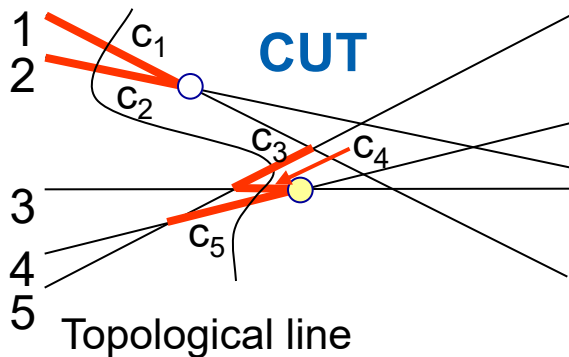
Stack S
 Ready vertex first edge idx

c_1

Intersect the trees – take the shorter edge



4e) Intersect with neighbors – push into S



Array of line equations E
 $y = a_i x + b$

1	a_1	b_1
2	a_2	b_2
3	a_3	b_3
4	a_4	b_4
5	a_5	b_5

UHT array
 Delimiting lines indices

1	$-\infty$	2
2	$-\infty$	5
3	5	4
4	5	6
5	3	6
6	$-\infty$	$+\infty$

LHT array
 Delimiting lines indices

1	$-\infty$	6
2	$-\infty$	1
3	5	1
4	5	3
5	3	1
6	$+\infty$	$-\infty$

CUT edges N
 Pairs of line indices delimiting the edge

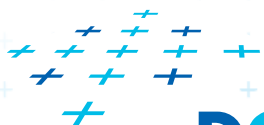
c_1	$-\infty$	2
c_2	$-\infty$	1
c_3	3	1
c_4	5	4
c_5	5	3

CUT Lines C
 Indices of supporting lines

c_1	1
c_2	2
c_3	5
c_4	3
c_5	4

Stack S
 Ready vertex first edge idx

c_4
c_1



Topological sweep algorithm

TopoSweep(L)

Input: Set of lines L sorted by slope (-90° to 90°), simple, not vertical

Output: All parts of an Arrangement $A(L)$ detected and then destroyed

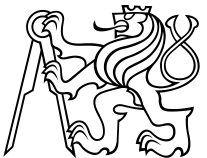
1. Let C be the **initial (leftmost) cut** – lines in increasing order of slope
2. Create the **initial UHT and LHT** incrementally:
 - a) UHT by inserting lines in decreasing order of slope
 - b) LHT by inserting lines in increasing order of slope
3. By consulting UHT and LHT
 - a) Determine the **right endpoints N** of all **edges** of the **initial cut C**
 - b) Store neighboring **lines with common endpoint** into **stack S**
(initial set of **ready vertices**)
4. Repeat until stack not empty
 - a) **Pop** next ready vertex from stack S (its upper edge c_i)
 - b) **Swap** these lines within the cut C ($c_i \leftrightarrow c_{i+1}$)
 - c) **Update** the horizon trees **UHT** and **LHT** (reenter edge parts)
 - d) Consulting UHT and LHT determine **new cut edges endpoints N**
 - e) If new neighboring edges share an endpoint -> **push** them on S

Slope



4d) Determining cut edges from UHT and LHT

- for lines $i = 1$ to n
 - Compare UHT and LHT edges on line i
 - Set the cut lying on edge i to the **shorter edge** of these
- Order of the cuts along the sweep line
 - Order changes **only at the intersection** v
(swap the neighbors in \mathcal{C})
 - Order of remaining cuts not incident with intersection v does not change
- After changes of the order, test the new neighbors for intersections
 - Store intersections right from sweep line into the stack

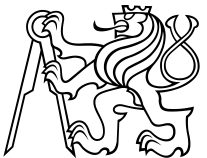


Complexity

- $O(n^2)$ intersections
=> $O(n^2)$ events (elementary steps)
- $O(1)$ amortized time for one step – 4c)
=> $O(n^2)$ time for the algorithm

Amortized time

= even though a single elementary step can take more than $O(1)$ time, the total time needed to perform $O(n^2)$ elementary steps is $O(n^2)$, hence the average time for each step is $O(1)$.



References

- [Berg] Mark de Berg, Otfried Cheong, Marc van Kreveld, Mark Overmars: Computational Geometry: Algorithms and Applications, Springer-Verlag, 3rd rev. ed. 2008. 386 pages, 370 fig. ISBN: 978-3-540-77973-5, Chapters 8., <http://www.cs.uu.nl/geobook/>
- [Mount] Mount, D.: *Computational Geometry Lecture Notes for Fall 2016*, University of Maryland, Lectures 14, 15, and 27.
<http://www.cs.umd.edu/class/fall2016/cmsc754/Lects/cmsc754-fall16-lects.pdf>
- [Edelsbrunner] Edelsbrunner and Guibas. Topologically sweeping an arrangement. TR 9, 1986, Digital www.hpl.hp.com/techreports/Compaq-DEC/SRC-RR-9.pdf
- [Rafalin] E. Rafalin, D. Souvaine, I. Streinu, "Topological Sweep in Degenerate cases", in Proceedings of the 4th international workshop on Algorithm Engineering and Experiments, ALENEX 02, in LNCS 2409, Springer-Verlag, Berlin, Germany, pages 155-156.
<http://www.cs.tufts.edu/research/geometry/other/sweep/paper.pdf>
- [Agarwal] Pankaj K. Agarwal and Mica Sharir. Arrangements and Their Applications, 1998, <http://www.math.tau.ac.il/~michas/arrsurv.pdf>

