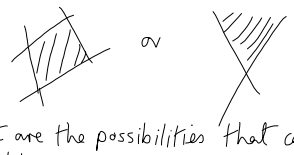
Half-plane intersection

- Consider a set $H = \{h_1, \dots, h_n\}$ of half-planes
  $h_i : a_i x + b_i y \leq c_i$
- Goal: compute intersection
  $C = \bigcap_{h_i \in H} h_i$.

Approach: recursive (divide & conquer)
That is, divide H into two sets H_1 & H_2 of roughly the same size
Compute $C_1 = \bigcap_{h_i \in H_1} h_i$, $C_2 = \bigcap_{h_i \in H_2} h_i$
& then $C = C_1 \cap C_2$.

říj 25-15:57

- Half-plane is convex set.
- Intersection of convex sets is convex.

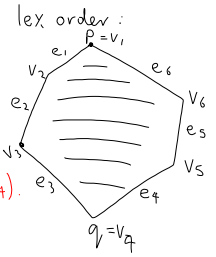
Eg:  or

- What are the possibilities that can arise?
  How can we represent them computationally?
- Consider lex ordering $p > q \Leftrightarrow$
  $p_y > q_y$ or $p_y = q_y$ & $p_x < q_x$.

říj 25-16:08

Consider non-empty convex subsets of the plane, which are intersections of fin. many half-planes & not subset of a line.

1) C has max p & min q in lex order:
Then boundary of C splits into a left path $L(C)$ & right path $R(C)$ from top to bottom:
$L(C) = (v_1, e_1, v_2, e_2, v_3, e_3, v_4)$
$R(C) = (v_1, e_6, v_6, e_5, v_5, e_4, v_4)$.
C is determined by sequences $L(C)$ & $R(C)$.



říj 25-16:12

- In each case, we'll represent C by a pair $L(C)$ & $R(C)$ of sequences consisting of vertices, edges (& half-edges).

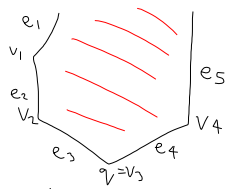
2) C has max, no min:
$L(C)$ & $R(C)$ are sequences beginning with a vertex & ending with a half-edge.

$L(C) = (v_1, e_1, v_2, e_2, v_3, e_3)$
$R(C) = (v_1, e_5, v_4, e_4)$
half edges

říj 25-16:22

3) C has no max, has min.

$L(C)$ & $R(C)$ are sequences beginning with half edges & ending in vertices.

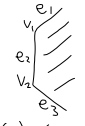


$LC = (e_1, v_1, e_2, v_2, e_3, v_3)$
$RC = (e_5, v_4, e_4, v_3)$
half-edges

říj 25-16:26

4) C has no min or max:

$L(C) = (e_1)$, $R(C) = (e_2)$ 
A strip

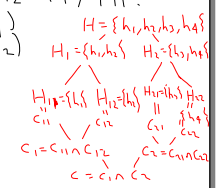
or 
$L(C) = (e_1, v_1, e_2, v_2, e_3)$
$R(C) = \emptyset$

or 
$L(C) = \emptyset$
$R(C) = (e_1, v_1, e_2, v_2, e_3)$

říj 25-16:29

Algorithm : HalfPlaneIntersection (H)

- Input $H = \{h_1, \dots, h_n\}$ set of n half-planes
- Output : Intersection C of H, described using sequences $L(C)$ & $R(C)$ of vertices, edges (half-edges, unbounded) describing its left & right path.
  - If $n=1$, determine $L(C)$ & $R(C)$
  - Else, put $H_1 = \{h_1, \dots, h_{\lfloor n/2 \rfloor}\}$ & $H_2 = H / H_1$.
  - Set $C_1 =$ HalfPlaneIntersection(C_1)
    $C_2 =$ " " (C_2)
    $C =$ IntersectionOfTwo(C_1, C_2)



říj 25-16:33

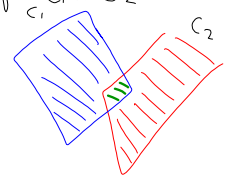
Algorithm : IntersectionofTwo (C_1, C_2)

- Input C_1, C_2 : int. of half-planes described using vertices & edges lists of left & right boundaries $L(C_1), R(C_1), L(C_2), R(C_2)$
- Output - $C_1 \cap C_2$ described using lists $L(C_1 \cap C_2)$ & $R(C_1 \cap C_2)$

↘ Over page

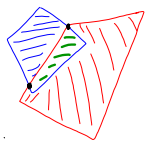
říj 25-16:42

Important : understand vertices of $C_1 \cap C_2$



(*) Vertices of $L(C_1 \cap C_2)$:

- vertices of $L(C_1)$ lying inside C_2.
- vertices of $L(C_2)$ lying inside C_1.
- points of intersection of left boundaries of C_1 & C_2.
- points of intersection of left path of C_i & right path of C_j, for $i \neq j$. (These are max & min points of intersection)



- Similarly can describe $R(C_1 \cap C_2)$.

říj 25-16:45

- Use sweep line method.
- Events : vertices of C_1, C_2 plus points of intersection
- Use queue of events Q
- Ordered sequence of edges T intersecting the Sweep-line.
  (No need for binary balanced tree - at most 4 edges intersect sweep line.)

(1) If C_1 or C_2 has no max, compute intersection of upper half edges (appearing at start of boundary) with the boundary of the other polygon, & add intersections to the queue.

říj 25-16:55

(2) Add vertices of C_1 & C_2 to queue.

(3) At event point v,
decide if $v \in L(C_1 \cap C_2)$ or $v \in R(C_1 \cap C_2)$ using (*) if so, add to appropriate list.

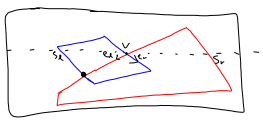
- If v is first element of $L(C_1 \cap C_2)$ or $R(C_1 \cap C_2)$ find edges with v as lower endpoint belonging to intersection, & decide which belong to $L(C_1 \cap C_2)$ / $R(C_1 \cap C_2)$. Add them to approp. path.

Eg. - if e appears before v in $L(C_1)$ then it belongs to $L(C_1 \cap C_2) \Leftrightarrow$ belongs to C_2.
if e appears before v in $R(C_1)$ then $e \in R(C_1 \cap C_2) \Leftrightarrow e$ belongs to C_2.

říj 25-17:03

(4) Look at edges going downwards from v - decide which lies in $C_1 \cap C_2$ & on which path $L(C_1 \cap C_2)$ or $R(C_1 \cap C_2)$. Add to path.

(5) - Let e_ℓ, e_r be leftmost / rightmost edges going down from v.
- To e_ℓ, find left-edge s_ℓ from the other set
  Find right edge s_r to e_r from the other set.
  Calculate intersections $s_\ell e_\ell$ & $e_r s_r$ & add them to queue.



říj 25-17:16

(6) If v is last element in $L(C_1 \cap C_2)$ & $R(C_1 \cap C_2)$
- so no edges in both C_1, C_2 going down from v, then we have computed intersection. Therefore we empty the queue.
- Otherwise, delete v from the queue.

Complexity: $T(1) = O(1)$

$T(n) = 2T(n/2) + \text{Time}(\text{Intersection of Two } (n/2, n/2))$

Int of Two $(C_1, C_2) = O(n_1 + n_2)$

n_1 vertices n_2 vertices

So $T(n) = 2T(n/2) + O(n)$

$\Rightarrow T(n) = O(n \log n)$.

říj 25-17:24