Lecture 6 - Half-plane intersection

Consider a set $H = \{h_1, \ldots, h_n\}$ of half-planes

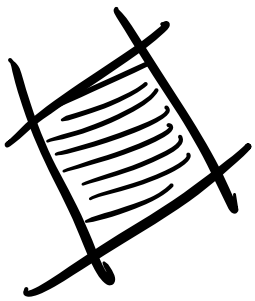
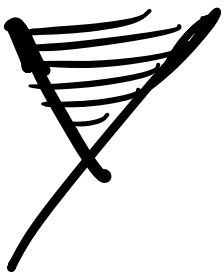
$h_i : a_i x + b_i y \leq c_i$

Goal : compute intersection $C = \bigcap_{h_i \in H} h_i$

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

certain convex sets.

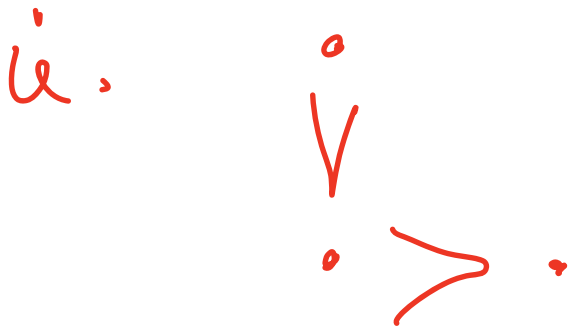
- Half-plane  is convex set.
- Intersection of convex sets is convex

Eg.  or 

- What shapes of convex sets can arise as intersection of Finitely many half-planes?
  - How can we represent them computationally?

- Consider lex. ordering

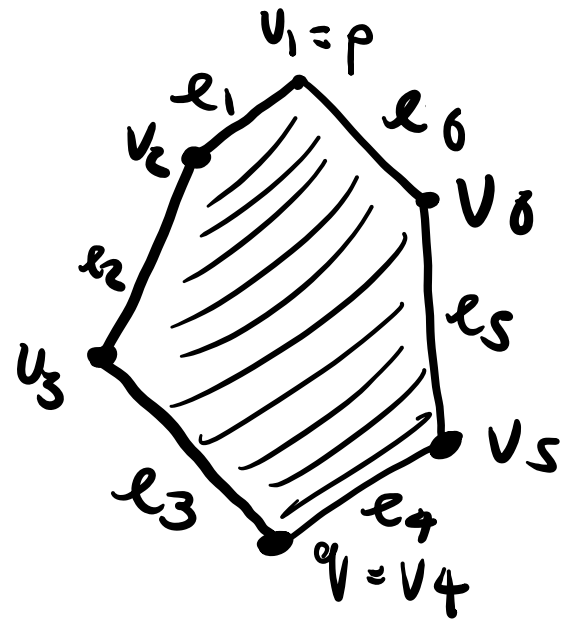
$$p > q \iff$$
$$p_y > q_y \text{ or } (p_y = q_y \;\&\; p_x < q_x)$$



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

1) C has max p & min. q in lex order:

The boundary of C splits into left path L(C) & right path R(C) from top to bottom:



$L(C) = (v_1, e_1, v_2, e_2, v_3, e_3, v_4)$

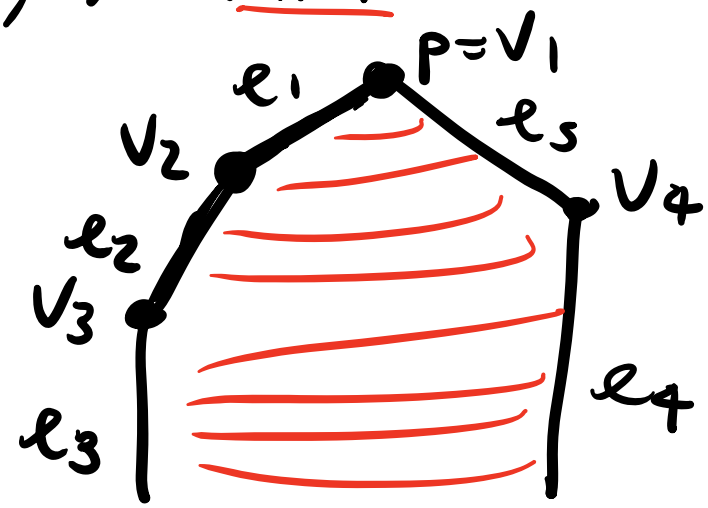
$R(C) = (v_1, e_0, v_0, e_5, v_5, e_4, v_4)$.

C is determined by L(C) & R(C).

In each case, represent

C by two sequences LC & RC consisting of vertices, edges, half-edges & unbounded edges.

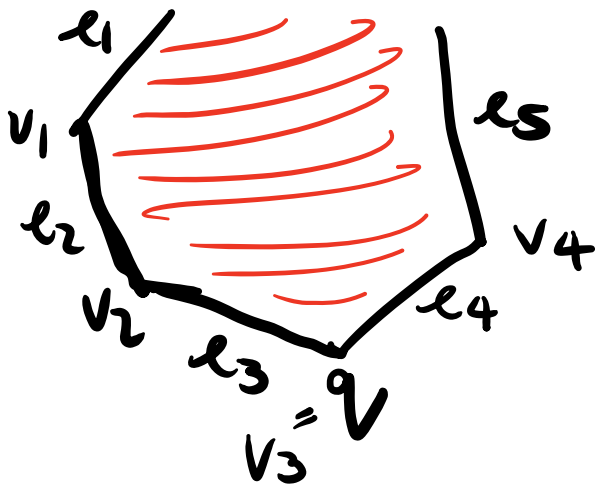
2) C has max, no min



LC, RC are sequences beginning with a vertex & ending with a half-edge.

3) C has no max, has min

LC, RC begin with half-edges & end in vertices



$LC = (e_1, v_1, e_2, v_2, e_3, v_3)$

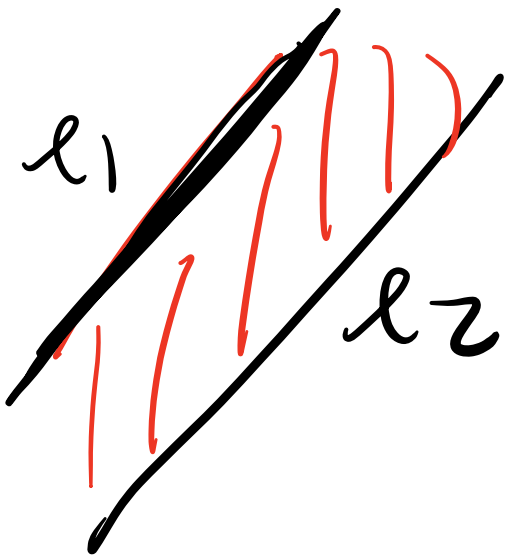
$RC = (e_5, v_4, e_4, v_3)$

half-edges.

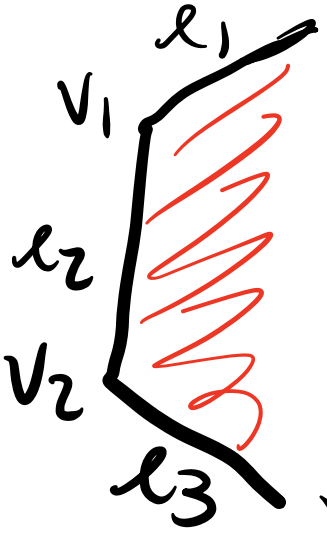
4) C has no min or max.

$LC = (e_1)$

$RC = (e_2)$



or



$LC = (e_1, v_1, e_2, v_3, e_3)$

$RC = \phi$

$LC = \phi$

$RC = (e_1, v_1, e_2, v_2, e_3)$

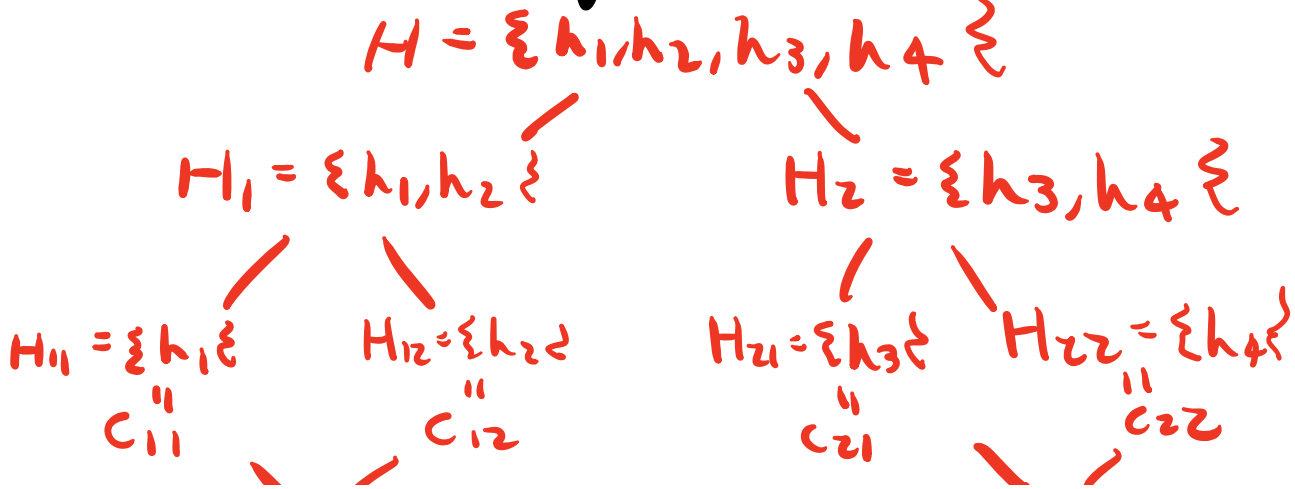
Algorithm : Half-plane Intersection (H)

- Input $H = \{h_1, \ldots, h_n\}$ set of half-planes
- Output : Intersection C of H, described using sequences LC & RC of vertices, edges, half-edges & unbounded edges describing left & right path.

- If $n = 1$, determine LC & RC.
- Else, put

$H_1 = \{h_1, \ldots, h_{\lfloor n/2 \rfloor}\}$ & $H_2 = H / H_1$

- Set C_1 = Half-plane Intersection (H_1)
  C_2 = Half-plane Intersection (H_2)
  C = Intersection of Two (C_1, C_2)



$C_1 = C_{11} \cap C_{12}$ $\quad$ $C_2 = C_{21} \cap C_{22}$

$C = C_1 \cap C_2$.

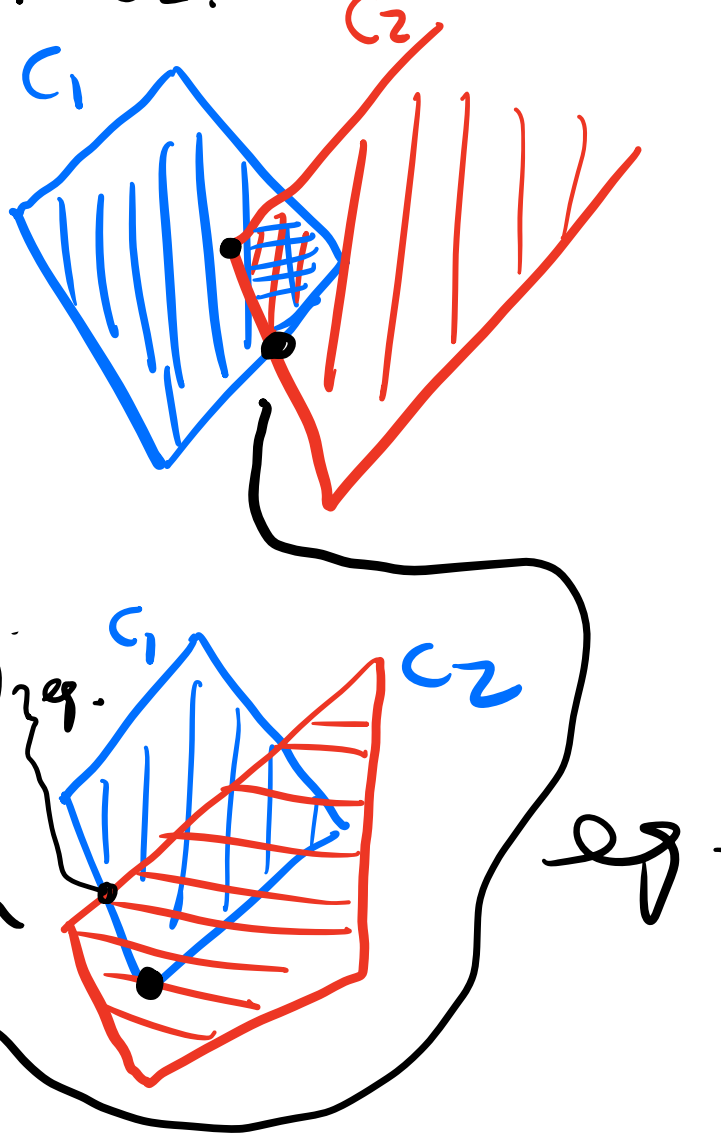
Algorithm : Intersection of Two (C_1, C_2)

- Input C_1, C_2 : int. of sets of halfplanes desc. using lists of l.& r. boundaries LC_1, RC_1, LC_2, RC_2.
- Output - $C_1 \cap C_2$ descr. using lists $L(C_1 \cap C_2)$ & $R(C_1 \cap C_2)$.

Complexity $O(n_1 + n_2)$

no of vertices of C_1, C_2

Important: understand vertices of $C_1 \cap C_2$.



⊛ Vertices of $L(C_1 \cap C_2)$

- vertices of $L(C_2)$ inside C_1.
- vertices of $L(C_1)$ inside C_2.
- points of int of $L(C_1)$ & $L(C_2)$
- points of intersection of left path of one & right path of the other.

these will be max / min. pts of the intersection.

Sim. $R(C_1 \cap C_2)$

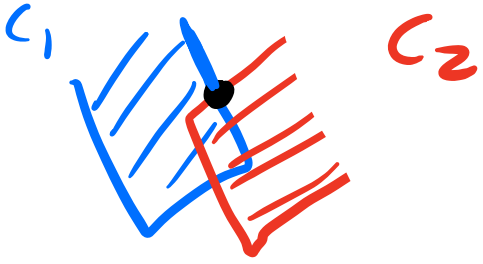
Use sweep-line method:

Q of events - vertices + intersections

T - ord. seq. of edges int. sweep-line
(at most 4, no need for tree)

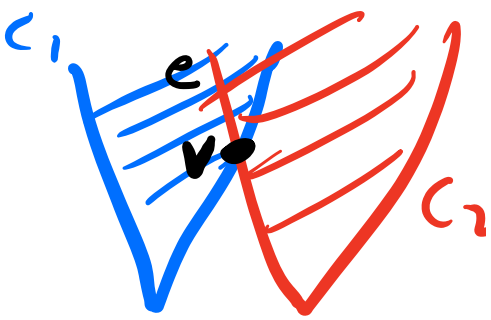
Algorithm

(1) If one of C_1, C_2 has no max, calc. int. of unbounded edges & add to Q



(2) a) At event point v, using (*) decide if $v \in L(C_1 \cap C_2)$ $v \in R(C_1 \cap C_2)$ & add.

b) If v is first-element of $L(C_1 \cap C_2)$ or $R(C_1 \cap C_2)$ Find edges w' v as lower endpoint & decide which belong to $L(C_1 \cap C_2)$ / $R(C_1 \cap C_2)$.



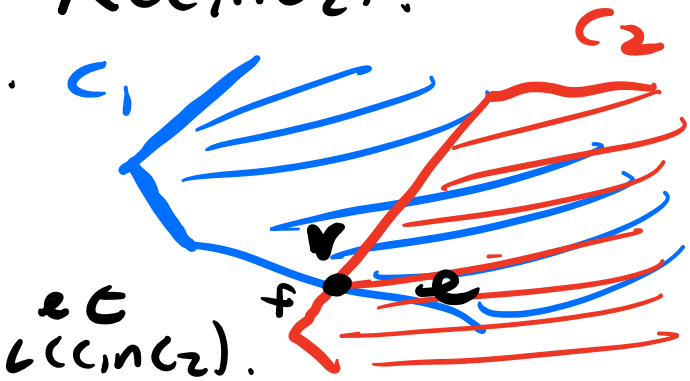
Add e to

Add to path. $L(C_1 \cap C_2)$, so $\{e, v\}$.

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

c) Update tree.

d) 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.

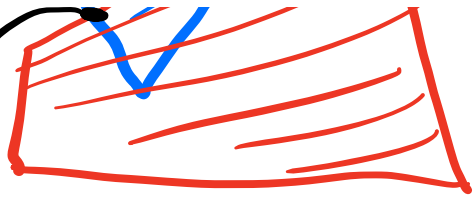


e) Let e_l, e_r be leftmost & rightmost edges going down from v

Find left edge s_l to e_l from other set.

Find right edge

Sr to er from other set.



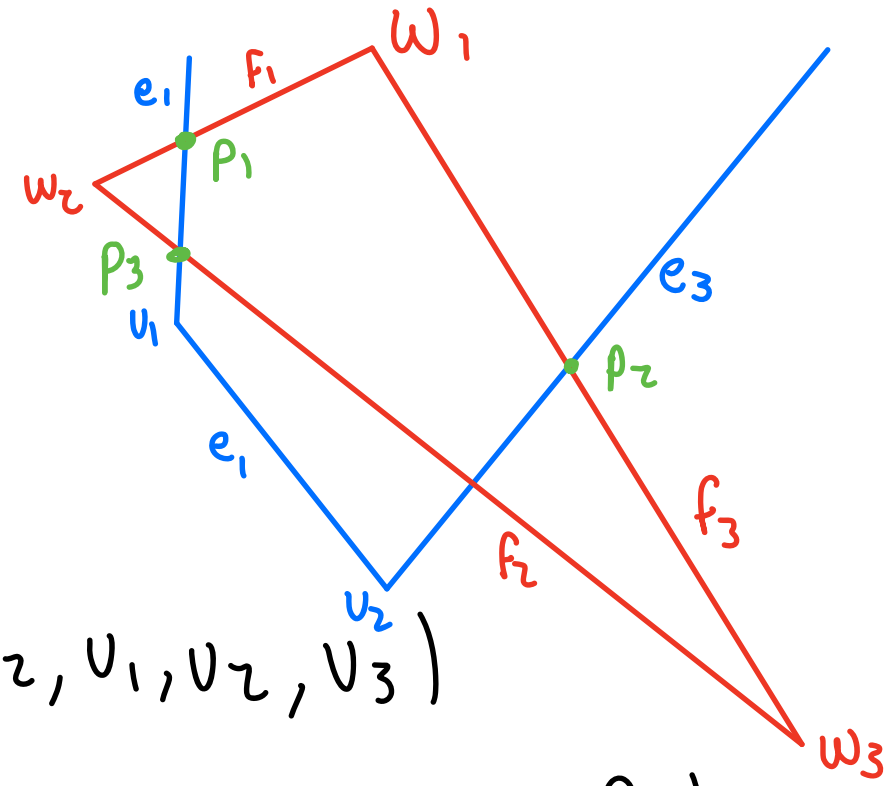
Calculate $S_2 \cap l_2$ & $S_r \cap l_r$ & add them to Q.

(5) - If v is last member of $L(C_1 \cap C_2)$ & $R(C_1 \cap C_2)$, then we have computed intersection.

Therefore we empty the queue.

- Otherwise, delete v from Q.

(See animation in E-Learning.)



$Q = (w_1, w_2, v_1, v_2, v_3)$

At w_1,

- $L = (w_1, f_1)$, $R = (w_1, f_3)$
- find p_1, p_2.

  $Q = (p_1, w_2, v_1, p_2, v_2, v_3)$

- At p_1,

  $L = (w_1, f_1, p_1, e_1)$, R same

- At w_2,

  find p_3, $Q = (p_3, v_1, p_2, v_2, v_3)$

  - - - -

- Note on complexity:
  insertions to Q take time
  $O(n_1 + n_2)$ -
  problem if we want
  complexity $O(n_1 + n_2)$
  overall.
- However, we can make
operations at event $p \in Q$
take constant time, if we
limit no. of elts in Q -
put in Q only
- uppermost verts of paths (≤ 4)
- endpoints of segs int sweepline (≤ 8)
- int. points of segs int sw. (≤ 4)

so at most 16 events at

any time.

- If we do this, need to modify algorithm to -
- add only uppermost endpoints to Q at start
- @ event p, add lower endpoints of segments coming down from p.

IntersectionOfTwo(C_1, C_2)

n_1 vertices, n_2 vertices

has complexity $O(n_1 + n_2)$

because Q & T contain ≤ constant no. of elements.

$T(n)$ complexity of half-plane intersection alg.

$T(n) = 2T(n/2) + \text{Time}(\text{Int. Of Two}(n/2, n/2))$

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

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