Intersections of half-planes

Compute $\bigcap_{i \in I} h_i$, each h_i is a half-plane

$\llcorner$ $a_i x + b_i y \leq c$

$H = \{h_i \mid i \in I\}$

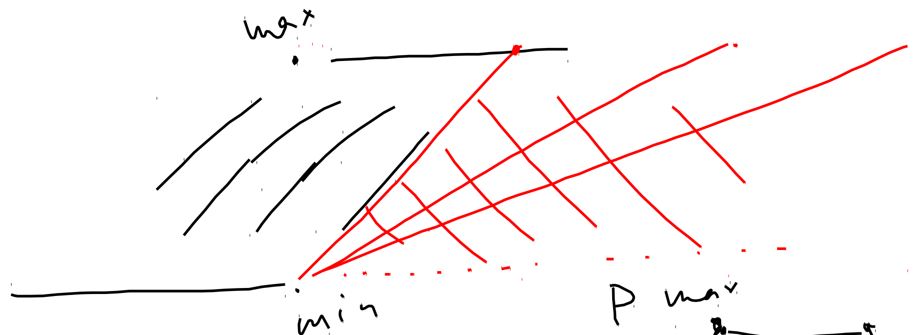
Will compute recurrently, i.e. split H into two sets of similar size, $H = H_1 \cup H_2$, compute

$C_1 = \bigcap H_1$, $C_2 = \bigcap H_2$ $\rightsquigarrow \bigcap H = C_1 \cap C_2$

Need to represent C_1, C_2 effectively, so that $\bigcap H$ can be computed easily.

Using the lexicographic order, for each intersection of half-planes $C = \bigcap_i h_i$, there is a maximal vertex of C or it is unbounded from above, similarly for minimal. There result 4 possibilities

1) max & min vertices exist



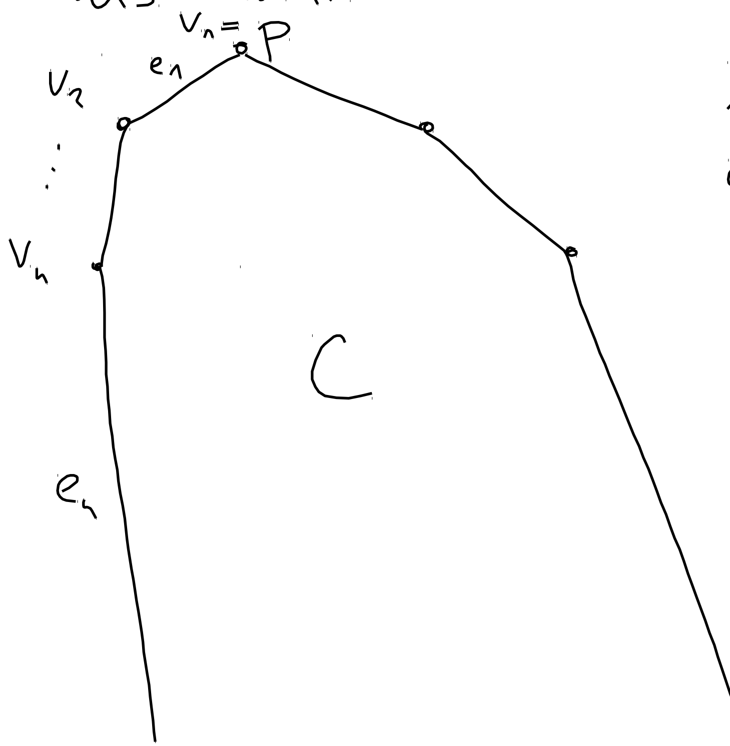
cannot happen

C described by two paths from p to q
left path, right path

both are sequences of the form

$$(p = v_1, e_1, v_2, \dots, v_n, e_n, v_{n+1} = q)$$

2. C has max vertex but not min



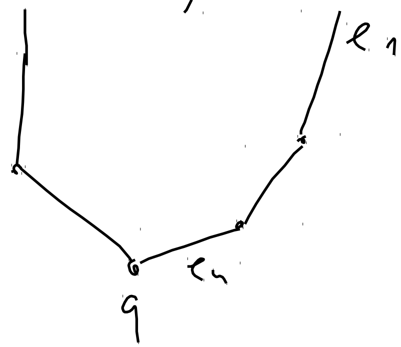
left and right paths are of the form

$$(p = v_1, e_1, \dots, v_n, e_n)$$

a half-line

3. min exists, max does not

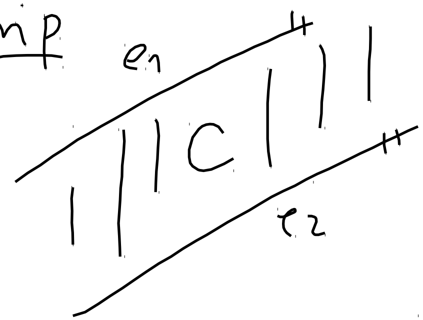
paths of the form $(e_1, v_2, \dots, v_n, e_n, v_{n+1} = q)$



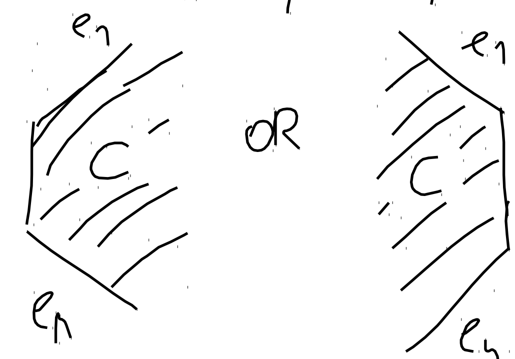
4. either C is a <u>half-plane</u>

either $(e),()$ OR $(),(e)$

or <u>strip</u>



has two paths $(e_1), (e_2)$



or there is a vertex

and then there is only one path

$(e_1, v_1, \dots, v_n, e_n), ()$ <u>OR</u> $(), (e_1, v_1, \dots, v_n, e_n)$

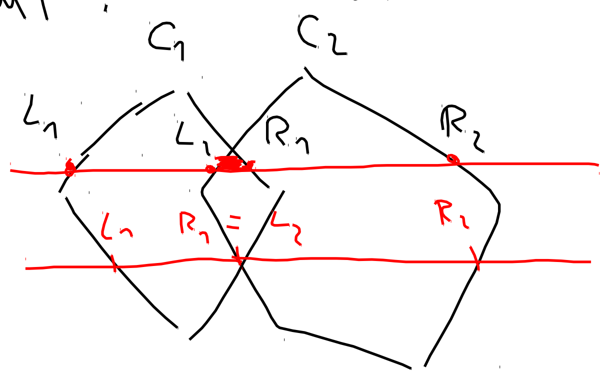
Algorithm for the intersection

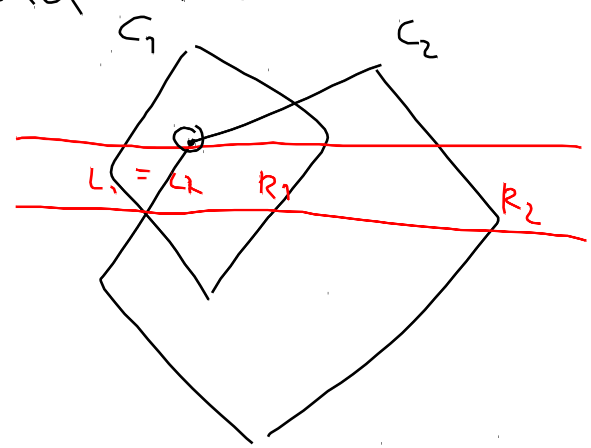
input - C_1 , C_2

both are intersections of half-planes described using doubly connected edge lists for their left and right paths.

output - the same for $C_1 \cap C_2$

important: to understand the vertices of $C_1 \cap C_2$.

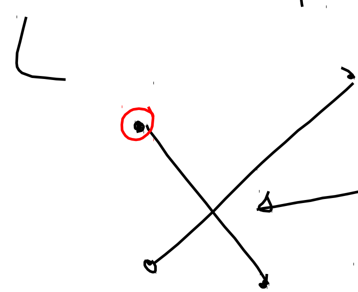




- a vertex of C_1 lying inside C_2
- C_2 C_1
- intersection of left paths — on the left path of $C_1 \cap C_2$
- right — right
- intersection of a left path of C_i — max, min points and right path of C_j — of $C_1 \cap C_2$

Sweep line method

- events are vertices of C_1, C_2 and intersections of edges of C_1, C_2



intersection pt. is added when processing the lower of the events above

- cover the poss. that there are no events above

1) treat edges with no upper endpoint
= half-lines appearing at the beginnings of paths in C_1, C_2

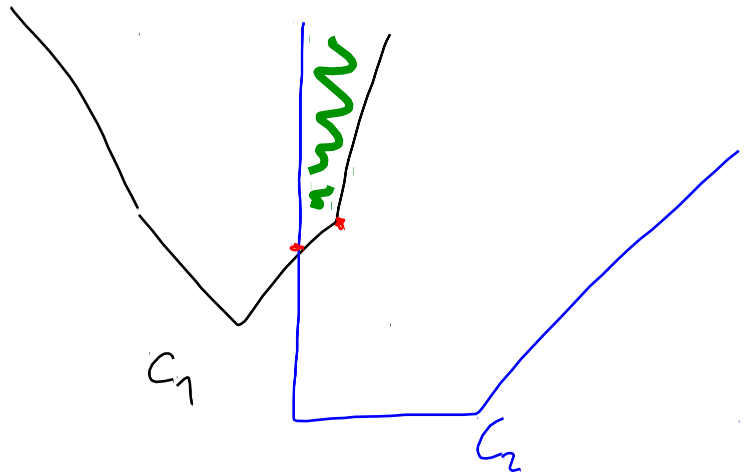
– at most four such $\rightsquigarrow$ compute their intersections and add them to the queue of events

2) add all vertices of C_1, C_2 as events.

We will keep the list of (at most 4) intersections of the boundaries of C_1, C_2 with the sweep line.

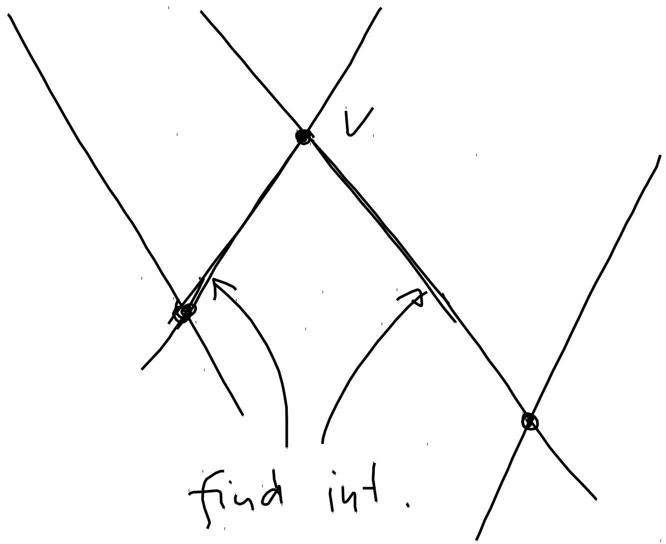
When sweep line passes through an event = a point v:

- decide if v is a vertex of $C_1 \cap C_2$ and if it lies on the left or right path.
  - if v is the first vertex in a path we have to decide if it is preceded by an edge

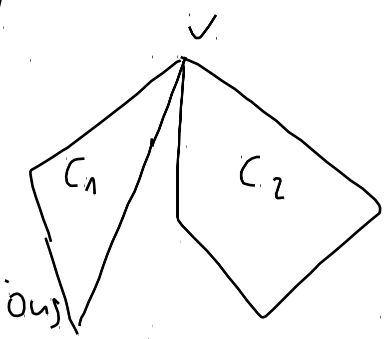


- look through the edges going down from v and decide which is in the left and the right path of $C_1 \cap C_2$.

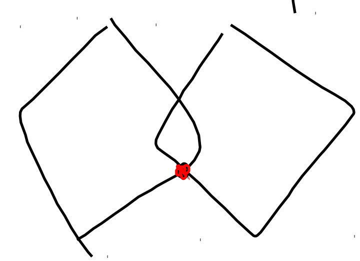
- need to compute new intersections:



find edges going down from v (at most 4) and for each of them find intersections with all edges that intersect the sweep line



- when we find a vertex common to the left and right paths of $C_1 \cap C_2 \Longrightarrow$ we can finish



Running time:

IntersectionOfTwo

C_1 with n_1 vertices

C_2 n_2

$O(n_1+n_2)$ events, each handled in constant time

$\Rightarrow$ time $O(n_1+n_2)$

The running time of the intersection algorithm $T(n)$ is

$$T(n) = 2T\left(\frac{n}{2}\right) + O(n) \rightsquigarrow T(n) = O(n \log n)$$