Computational Geometry

Dr. Marco Pellegrini
Lecture 8: sweeping line (II)

1

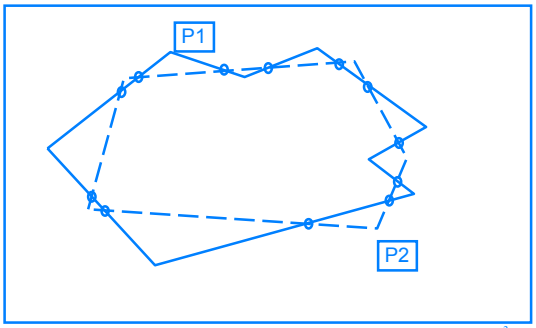
Summary of lecture 8

- Summary of lecture 7:
  Finding 1 intersection among segments.
  The sweeping line approach.
- Questions, collect exercises
- Summary of lecture 8:
  More on sweeping line:
  Find all intersections among segments.
  Find intersection of convex polygons.

2

Given some segments, find all intersections.

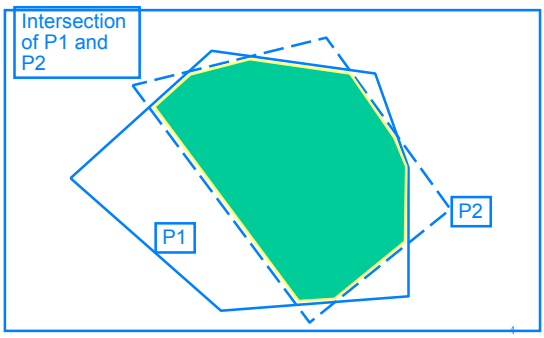
(problem 4)



3

Given two convex polygons, find their intersection

(problem 5)



4

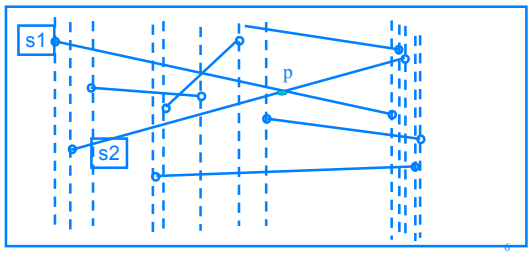
Observations:

- We saw in lecture 7 how to use the sweeping line approach to detect 1 intersection among segments.
- Here we extend the sweeping line method to the problem of finding all intersections.
- In lecture 7 as soon as we found an intersection we stopped the algorithm.
  The correctness of the algorithm in lecture 7 depends critically on this fact, since we assume that the vertical order of the segments in each strip we visit does not change (except for insertion/deletion of 1 segment), between adjacent strips.
  This is true as long as we do not have found any intersection.

5

What happens at an intersection point.

- If s1 and s2 intersect at point p, their relative order changes at p. And moreover, s1 and s2 are neighbors in the vertical order near p.



6

What is the algorithm to do?

- When our (ideal) sweeping line passes through the intersection point p, the algorithm HAS to switch the order of the segments s1 and s2 in the dictionary holding the vertical order of the segments.
- Since it is expensive to find out how many times we have checked the pair (s1,s2), every time we check s1 and s2, finding the intersection p the algorithm has to REMEMBER p, and the strip containing p, or better it has to remember the position of p among the endpoints in the sorted array of the end-points.

7

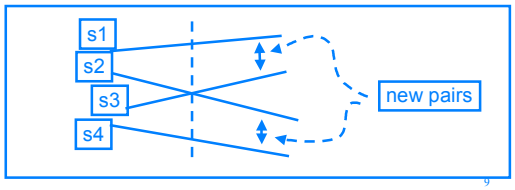
How do we do it. 1

- (first the second point of previous slide)
  We do not use a SORTED ARRAY A to hold the sorted end-points of the segments, instead we use a HEAP H. The heap will allow us to insert efficiently new points (intersection points) as we find them dynamically during the execution of the algorithm.
- For each point in the heap we carry a tag saying if it is a left/right endpoint or an intersection point, and of which segments
- Since we visit the strips in left-to-right order, the operation EXTRACT-MIN gives us the next point to process in the right order.

8

How we do it. 2

- When we reach an intersection point within a strip, we have to switch the two intersecting segments in the Dictionary.
  This creates potentially 2 new pairs to check for intersections.



9

Algorithm for problem 4.

```
ALL-INTRESECTIONS(S:set of segments)
  CREATE(T), CREATE(H)
  Put all end-points of segments of S into H.
  WHILE H is not empty DO
    p = EXTRACT-MIN(H)
    IF p is left-end-point of s THEN
      p1 = INTERSECTION(s, ABOVE(T,s))
      p2 = INTERSECTION(s, BELOW(T,s))
      INSERT p1 and p2 in H, if they exist and not yet in H.
      INSERT(s,T)
    ELSE IF p is right-end-point of s THEN
      p1 = INTERSECTION(BELOW(T,s), ABOVE(T,s))
      INSERT p1 in H if it exists and not yet in H
      DELETE(s,T)
    ELSE %p is an intersection of (s1,s2)%
      OUTPUT((p,s1,s2))
      SWITCH(s1,s2,T)
      p1 = INTERSECT(s2, ABOVE(T,s2))
      p2 = INTERSECT(s1, BELOW(T,s1))
      INSERT p1 and p2 in H if they exist and not yet in H
    END IF
  END WHILE
```

10

Observations 1.

- We assume that there are no vertical segments, that no more than two segments may intersect at the same point.
- These assumptions make the main idea easier to grasp. When you write actual code you have to take care of these cases too.
- Before inserting a new point in the heap H we check if it is there already, so can implement it as a combination of MEMBER and INSERT.
- If we do not eliminate duplicates we may loop for ever.

11

Observation 2.

- We assume that the procedure INTERSECT(s1,s2) returns the intersection point p or a flag if p does not exists.
- Practically one should use the procedure INTERSECT based on the left-turn tests to check if the intersection exists and only then compute the intersection point.
- Also, ABOVE(T,s), BELOW(T,s) should return flags if such segments above and below do not exist.

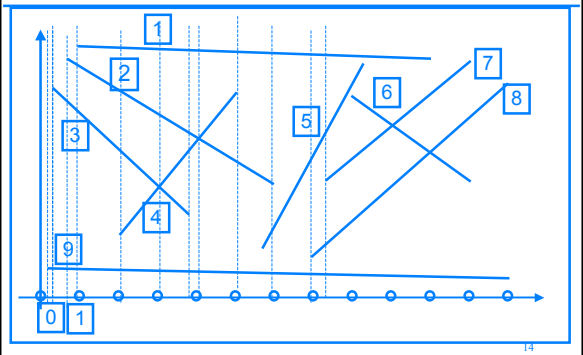
12

Exercise.

- Run the algorithm to find all intersections on the set of segments given in the next slide.
- You should show the changes in the vertical order of the segments (stored in a Dictionary, implemented as a 2-3-tree). Just the order, not the whole data structure.
- You should show the content of the heap H, but not the heap organization.
- Operations, INSERT, DELETE, EXTRACT-MIN, INTERSECT, ABOVE, BELOW are computed by yourself with your eye and brain.
- Show all intersection in the order in which they are printed out by the algorithm.

13

DATA



14

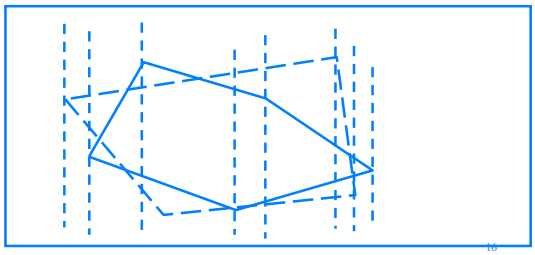
Asymptotic analysis

- Let us call n the number of segments and k the number of intersections.
- In the heap we will insert in total 2n+k points and delete 2n+k points, the cost of each operation is O(log (size H)), that is also O(log n) because $2n+k < n^2$, for n large enough.
- For each point in the heap we perform at most 2 operations among ABOVE, BELOW, SWITCH on the Dictionary T, each of cost O(log n).
- For each point in the queue we test at most two pairs for possible intersection. Cost O(1).
- In total the time of the algorithm is O((n+k)log n)

15

Intersection of convex polygons.

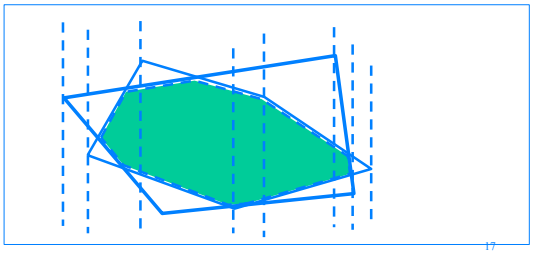
- A vertical line meets a convex polygon in 2 points, 2 convex polygon in at most 4 points. So we do not need a 2-3-tree. An array of 4 elements is enough.



16

Intersection of convex polygons 2.

- The boundary of the intersection region is always the second and third segment in the vertical order, when there are 4 segments meeting a vertical line.



17

Finding intersection of convex polygons.

(Hint and homework)

- To find the intersection of convex polygons we tun the general algorithm and, every time the segment in second and third position changes we record this in the output.
- HOMEWORK: write a detailed algorithm to compute the intersection of two convex polygons. The output must be the sequence of vertices of the convex polygon.
  Due in 1 week.

18

Conclusions

- We have seen that we can extend the sweeping line method to compute all intersections and to compute the intersection of two convex sets.
- One can compute also the intersection of two simple polygon. One has to be careful though since there can be different disconnected pieces.
- The sweeping line method is efficient when some but not all of the pairs of edges intersect.

19