Computational Geometry

Dr. Marco Pellegrini
Lecture 7: sweeping line (I)

1

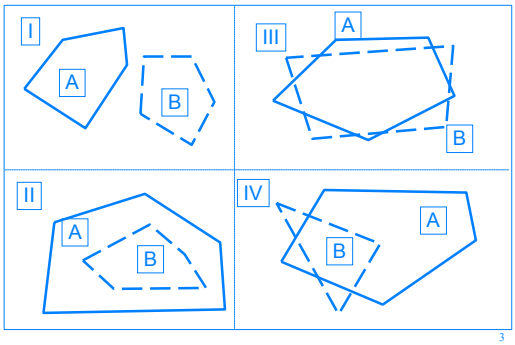
Summary of lecture 7

- Lecture 6:
  Triangulation of simple polygons.
- Lecture 7:
  Testing if two polygons intersect,
  Testing if any pair of segments among many segments intersect.
  The sweeping line method.

2

Given two convex polygons do they intersect?

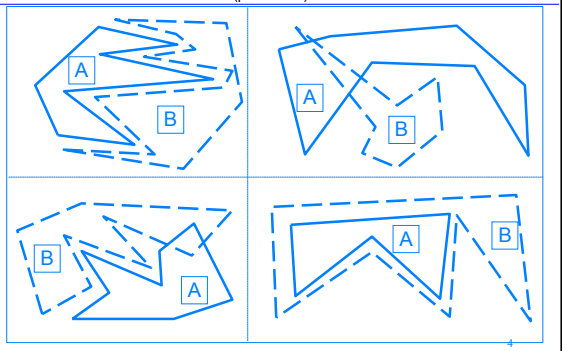
(problem 1)



3

Given two simple polygons, do they intersect?

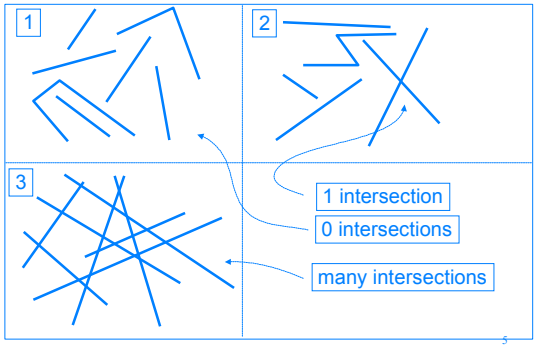
(problem 2)



4

Given any set of segments, is there an intersection?

(problem 3)



5

Exercise

- Write 3 algorithms to solve the problems of the previous 3 slides.
- Hint: 1) start from the last problem.
  2) USE any code that we have discussed previously as a procedure that you can call.
  3) aim at quadratic complexity.
  4) you should reuse also algorithms that you are writing in this exercise, when possible.

6

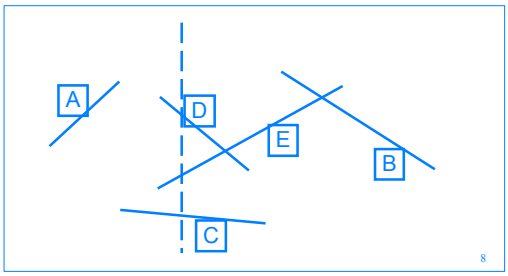
Observations

- You have written 3 algorithms to solve problems 1,2 and 3 which probably are $O(n^2)$.
- The part of the code that takes quadratic time is the solution to problem 3, detecting if a pair of segments intersect or not (problem 3).
- Can we solve problem 3 in less than quadratic time?
- If we can do that we can solve also problems 1 and 2 more efficiently.

7

Observation 1

- If two segments are on different sides of a vertical line, those two segments cannot intersect.



8

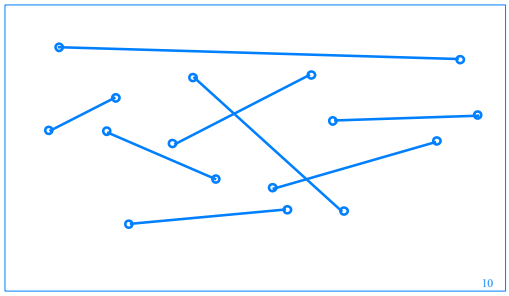
Observation 2

- If two segment intersect, then there is a vertical line passing through both of them.
- Conclusions: We have to use the test INTERSECT only on pairs of segments s1, s2 that are crossed by a vertical line.
- Problem: there are infinite vertical lines!
- Problem: if a vertical line cuts through all of the segments we have still to test all pairs of segments!

9

Observation 3

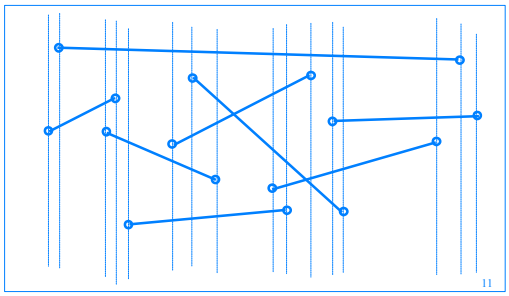
- Draw vertical lines through the end-points of the given segments.



10

Conclusion 3

- Every other vertical line will meet the same segments met by one neighbor lines we draw.



11

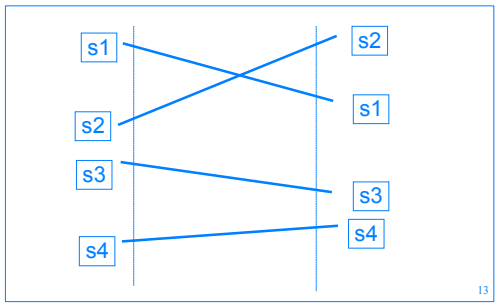
Observation 4

- If there are n segments, we need to use ONLY at most 2n vertical lines in order to have the same result that we would get by considering all infinite vertical lines.
- From now we consider only those 2n vertical line, sorted from left to right.
- Between two consecutive vertical lines there are no end-point of the segments.
- We call the region between two consecutive vertical lines a STRIP. Now we concentrate on what may happen within a strip.

12

Observation 5

- Consider the order in which segments are intersected by the two vertical lines bounding a strip.



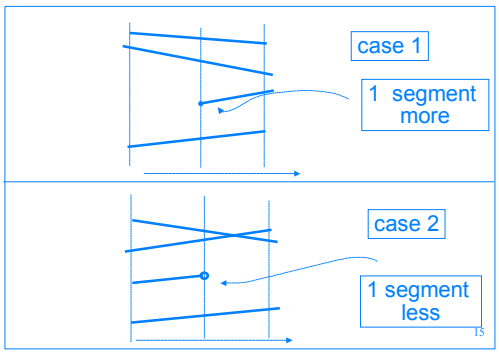
13

Observation 5, contd.

- When two segments change their relative vertical order within the strip we have an INVERSION.
- If two segments intersect, they generate an inversion.
- Conversely, an inversion of two segments can happen only if they intersect.
- First conclusion is: if we do not find any intersection in the strip, then the order of the segments is the same at both ends of the strip.
- When we pass to the next strip the only difference is one segment to be inserted or deleted.

14

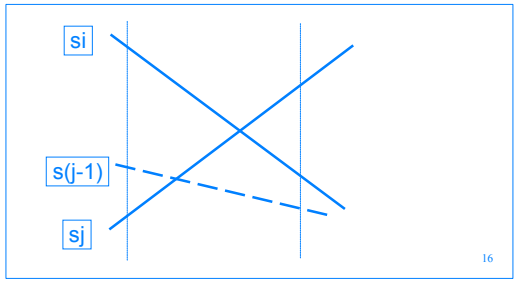
Two adjacent strips



15

Observation 6.

- If two segments meet in a strip, then there are (other) two consecutive segments in the strip that meet each other.



16

Conclusion 6

- To detect an intersection we can just check pairs of consecutive segments in the vertical order at the left side of a strip.
- If, during these test we find an intersection, we stop.
- If we do not find an intersection among those pairs of segments, then there is no intersection among any pair of segment in the strip.
- Moreover in this second case the order at the right-side of the strip has not changed, except in. one position.
- In the next strip there are only 1 or 2 new pairs of consecutive edges to check.

17

Algorithm to find an intersection

- Visit the strips in order from left to right. Keep the vertical order of the segments and when there is a new pairs of consecutive segments in the vertical order, test them.
- To insert, delete, find neighbors in an ordered set of segments we use a Dynamic Binary Balanced Tree.

18

How to store segments in a 2-3-tree

- In csac11 we saw a 2-3-tree for storing numbers, the main operation was, given two numbers, decide which it the largest.
- We can use the same algorithm to store segments if we substitute the <,>,= operations on two numbers with an operation IS_ABOVE(s1,s2,a) which is true when at the vertical line x=a, the segment s1 is above the segment s2.
- Homework: deign the test IS_ABOVE(s1,s2,a) for two non-vertical segments s1,s2 intersecting the vertical line x=a.

19

Operations on the 2-3-tree.

- We will use a 2-3-tree (a dynamic balanced binary tree) with the following operations:
- CREATE(T), generates an empty tree and call it T.
- INSERT(T,a,s), inserts segment s in T when the vertical line is x=a.
- DELETE(T,a,s), deletes s from T when the vertical line is at x=a.
- ABOVE(T,a,s), return the segment above s in T at the position x=a.
- BELOW(T,a,s), return the segment below s in T at the position x=a.

20

Algorithm for problem 3

```
ANY-INTERSECTION(S: set of segments)
  CREATE(T)
  Sort end-points of segments in S by x-coordinate
    into an array A
  FOR i=1 TO length(A)
    DO IF A[i] is the left end-point of s
        THEN INSERT(T,A[i]x,s)
            IF s intersects ABOVE(T,A[i]x,s) OR
               s intersects BELOW(T,A[i]x,s)
               THEN RETURN True
            END IF
        ELSE ( A[i] is a right-end-point of s)
            IF BELOW(T,A[i]x,s) intersects ABOVE(T,A[i]x,s)
               THEN RETURN True
            END IF
            DELETE(T,A[i]x,s)
      END IF
  END FOR
  RETURN False
```

21

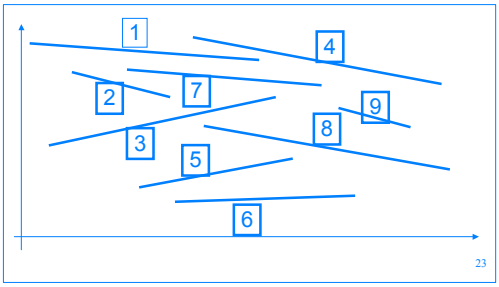
Time analysis

- Creating T takes time O(1),
  Sorting the end-points takes time O(n log n).
- The FOR loop is executed 2n times.
- Inside the FOR loop we either make one insertion and two INTERSECT tests, or
  one deletion and one INTERSECT test.
- INTERSECT tests take constant time,
  INSERTION and DELETION take time O(log size(T)),
  and size(T) $\leq$ n.
- Finally each iteration of the loop takes time O(log n),
- The whole algorithm is O(n log n).

22

Exercise:

- Simulate the algorithm ANY-INTERSECTION showing, the segments stored at leaves of T, which pairs of segments are tested, which inserted and deleted.



23

Exercise and observations.

- How many pairs of segments have you tested using the INTERSECTION test?
  How many pairs of segments are there in the picture?
- The algorithm ANY-INTERSECTION performs much fewer INTERSECTION tests than the quadratic algorithm at the beginning of class. The drawback is that we need a dynamic balanced tree to store and retrieve segments.
- Practically for n between 2 and 7 is still OK to check every pair.
  For n between 5 and say 20 or 30 one would use the algorithm ANY-INTERSECTION, where the Dictionary is implemented in an array.
  For n greater than 50 one has to use binary balanced trees (e.g. 2-3-trees, red-black trees, etc..).

24

Conclusions

- We have seen a fast method O(n log n) to test whether two segments intersect among n segments.
- The general idea is called SWEEPING LINE paradigm, which is very powerful and we will see more applications.
- We can solve problems 1,2 and 3 in the introduction in time O(n log n). We can also test a polygon for simplicity (lecture 1) in time O(n log n).
- Homework for next week: Solve problem 1 in time O(n). (It is very easy).

25