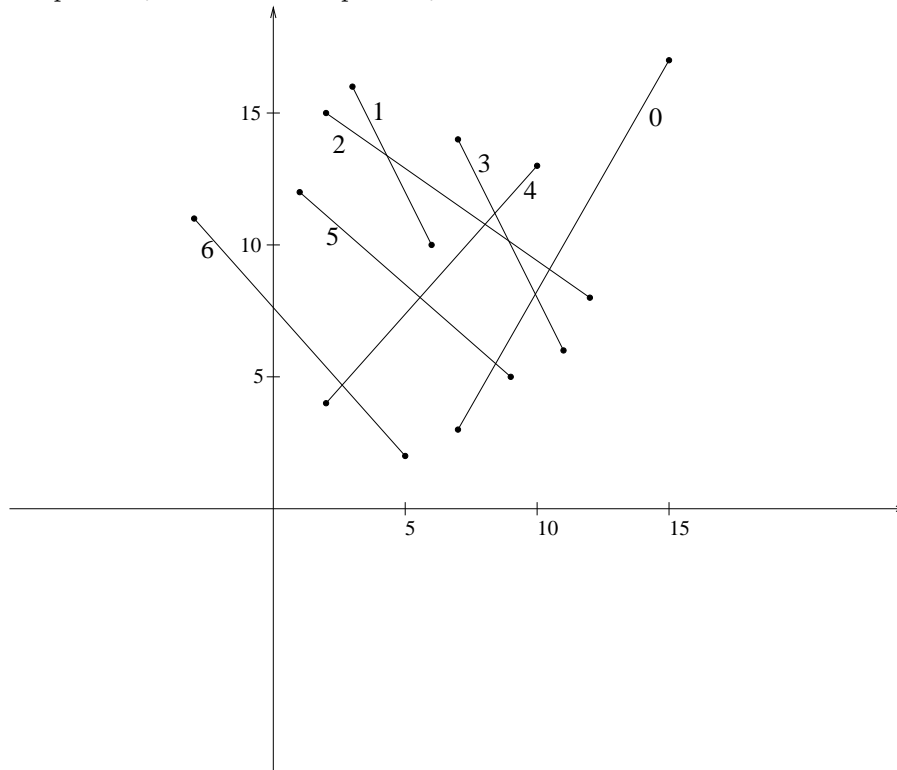


COMPUTATIONAL GEOMETRY

Example of Bentley-Ottmann Sweep

Consider the set S of 7 line segments given by $S = \{s_0, s_1, \dots, s_6\} = \{((15,17),(7,3)), ((3,16),(6,10)), ((2,15),(12,8)), ((7,14),(11,6)), ((10,13),(2,4)), ((1,12),(9,5)), ((-3,11),(5,2))\}$, where each segment $s_i = (a_i, b_i)$ has upper endpoint a_i and lower endpoint b_i .



Apply the Bentley-Ottmann sweepline algorithm to S . Give the event queue Q and the sweep status \mathcal{L} just after each event. (Use the notation as in the text that x_{ij} denotes the point (if any) at the intersection of segment s_i and segment s_j .)

Show the chart, just as done in class. (You need not compute the actual intersection points x_{ij} .)

(a). Apply the “usual” Bentley-Ottmann sweep, without removals from the event queue.

(b). Apply the “modified” Bentley-Ottmann sweep (as described in class; see comment before Theorem 2.4 in BKOS, or see problem 2, Section 7.8.1, page 269 of O’Rourke), which removes from the event queue those crossing events that no longer correspond to segments that are adjacent in the sweep line status (SLS).

I show the execution of the algorithm in the table below. I use one table to illustrate both the “usual” and the “modified” versions of the algorithm: I place an event point x_{ij} in square brackets (“ $[x_{ij}]$ ”) if it would not be present if we follow the modified Bentley-Ottmann algorithm, which deletes crossing events from the queue whenever the corresponding segments stop being adjacent in the sweep status (they are reinserted later, when the segments are again adjacent).

