12. Visibility Graphs and 3-Sum

Lecture on Monday 9th November, 2009 by Michael Hoffmann <hoffmann@inf.ethz.ch>

12.1 Sorting all Angular Sequences.

Theorem 12.1 *Consider a set P of n points in the plane. For a point $q \in P$ let $c_P(q)$ denote the circular sequence of points from $S \setminus \{q\}$ ordered counterclockwise around q (in order as they would be encountered by a ray sweeping around q). All $c_P(q)$, $q \in P$, collectively can be obtained in $O(n^2)$ time.*

Proof. Consider the projective dual P^* of P. An angular sweep around a point $q \in P$ in the primal plane corresponds to a traversal of the line q^* from left to right in the dual plane. (A collection of lines through a single point q corresponds to a collection of points on a single line q^* and slope corresponds to x-coordinate.) Clearly, the sequence of intersection points along all lines in P^* can be obtained by constructing the arrangement in $O(n^2)$ time. In the primal plane, any such sequence corresponds to an order of the remaining points according to the slope of the connecting line; to construct the circular sequence of points as they are encountered around q, we have to split the sequence obtained from the dual into those points that are to the left of q and those that are to the right of q; concatenating both yields the desired sequence. □

12.2 Segment Endpoint Visibility Graphs

A fundamental problem in motion planning is to find a short(est) path between two given positions in some domain, subject to certain constraints. As an example, suppose we are given two points $p, q \in \mathbb{R}^2$ and a set $S \subset \mathbb{R}^2$ of obstacles. What is the shortest path between p and q that avoids S?

Observation 12.2 *The shortest path between two points that does not cross a set of polygonal obstacles (if it exists) is a polygonal path whose interior vertices are obstacle vertices.*

One of the simplest type of obstacle conceivable is a line segment. In general the plane may be disconnected with respect to the obstacles, for instance, if they form a closed curve. However, if we restrict the obstacles to pairwise disjoint line segments then there is always a free path between any two given points. Apart from start and goal position, by the above observation we may restrict our attention concerning shortest paths to straight line edges connecting obstacle vertices, in this case, segment endpoints.

Definition 12.3 *Consider a set S of n disjoint line segments in $\mathbb{R}^2$. The* **segment endpoint visibility graph** *$\mathcal{V}(S)$ is a plane straight line graph defined on the segment endpoints. Two segment endpoints p and q are connected in $\mathcal{V}(S)$ if and only if*

- *the line segment $\overline{pq}$ is in S or*
- $\overline{pq} \cap s \subseteq \{p, q\}$ *for every segment $s \in S$.*

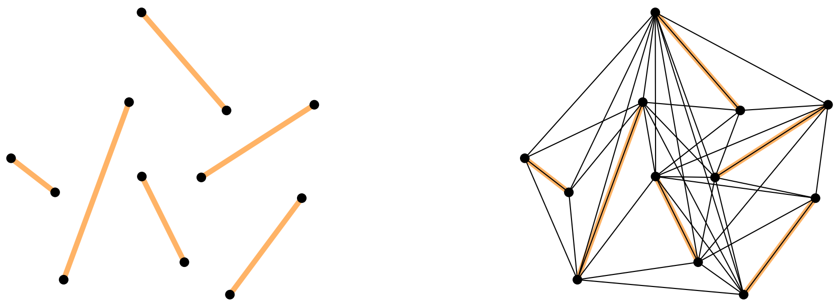


Figure 12.1: *A set of disjoint line segments and their endpoint visibility graph.*

If all segments are on the convex hull, the visibility graph is complete. If they form parallel chords of a convex polygon, the visibility graph consists of copies of K_4, glued together along opposite edges and the total number of edges is linear only. In any case, these graphs are Hamiltonian. :-)

Constructing $\mathcal{V}(S)$ for a given set S of disjoint segments in a brute force way takes $O(n^3)$ time. (Take all pairs of endpoints and check all other segments for obstruction.)

Theorem 12.4 (Welzl [3]) *The segment endpoint visibility graph of n disjoint line segments can be constructed in worst case optimal $O(n^2)$ time.*

Proof. We have seen above how all sorted angular sequences can be obtained from the dual line arrangement in $O(n^2)$ time. Topologically sweep the arrangement from left to right (corresponds to changing the slope of the primal rays from $-\infty$ to $+\infty$) while maintaining for each segment endpoint p the segment $s(p)$ it currently "sees" (if any). Initialize by brute force in $O(n^2)$ time (direction vertically downwards). Each intersection of two lines corresponds to two segment endpoints "seeing" each other along the primal line whose dual is the point of intersection. In order to process an intersection, we only need that all preceding (located to the left) intersections of the two lines involved have already been processed. This order corresponds to a topological sort of the arrangement graph where all edges are directed from left to right. A topological sort can be obtained, for instance, via (reversed) post order DFS in linear time.

When processing an intersection, there are four cases. Let p and q be the two points involved such that p is to the left of q.

1. The two points belong to the same input segment $\rightarrow$ output the edge pq, no change otherwise.
2. q is obscured from p by $s(p)$ $\rightarrow$ no change.
3. q is endpoint of $s(p)$ $\rightarrow$ output pq and update $s(p)$ to $s(q)$.

4. Otherwise q is endpoint of a segment t that now obscures $s(p) \rightarrow$ output pq and update $s(p)$ to t.

Thus any intersection can be processed in constant time and the overall runtime of this algorithm is quadratic. □

12.3 3-Sum

The 3-Sum problem is the following: Given a set S of n integers, does there exist a three-tuple[1] of elements from S that sum up to zero? By testing all three-tuples this can obviously be solved in $O(n^3)$ time. If the tuples to be tested are picked a bit more cleverly, we obtain an $O(n^2)$ algorithm.

Let $(s_1, \ldots, s_n)$ be the sequence of elements from S in increasing order. Then we test the tuples as follows.

```
For $i = 1, \ldots, n-2$ {
    $j = i$, $k = n$.
    While $k \geq j$ {
        If $s_i + s_j + s_k = 0$ then exit with triple $s_i, s_j, s_k$.
        If $s_i + s_j + s_k > 0$ then $k = k - 1$ else $j = j + 1$.
    }
}
```

The runtime is clearly quadratic (initial sorting can be done in $O(n \log n)$ time). Regarding the correctness observe that the following is an invariant that holds at begin of the inner loop: There exists no suitable triple that contains s_i and s_ℓ for any $\ell < j$ or $\ell > k$.

Interestingly, this is the essentially the best algorithm known for 3-Sum. It is widely believed that the problem cannot be solved in sub-quadratic time, but so far this has been proven in some very restricted models of computation only, such as the linear decision tree model [1].

12.4 3-Sum hardness

There is a whole class of problems that are equivalent to 3-Sum up to sub-quadratic time reductions [2]; such problems are referred to as **3-Sum-hard**.

Definition 12.5 *A problem P is 3-Sum-hard if and only if every instance of 3-Sum of size n can be solved using a constant number of instances of P—each of $O(n)$ size—and $o(n^2)$ additional time.*

[1]That is, an element of S may be chosen twice or even three times, although the latter makes sense for the number 0 only. :-)

As an example, consider the Problem **GeomBase**: Given n points on the three horizontal lines $y = 0$, $y = 1$, and $y = 2$, is there a non-horizontal line that contains at least three of them?

GeomBase can be reduced to 3-Sum as follows. For an instance $S = \{s_1, \ldots, s_n\}$ of 3-Sum, create an instance P of GeomBase in which for each s_i there are three points in P: $(s_i, 0)$, $(-s_i/2, 1)$, and $(s_i, 2)$. If there are any three collinear points in P, there must be one from each of the lines $y = 0$, $y = 1$, and $y = 2$. So suppose that $p = (s_i, 0)$, $q = (-s_j/2, 1)$, and $r = (s_k, 2)$ are collinear. The inverse slope of the line through p and q is $\frac{-s_j/2-s_i}{1-0} = -s_j/2 - s_i$ and the inverse slope of the line through q and r is $\frac{s_k+s_j/2}{2-1} = s_k + s_j/2$. The three points are collinear if and only if the two slopes are equal, that is, $-s_j/2 - s_i = s_k + s_j/2 \iff s_i + s_j + s_k = 0$.

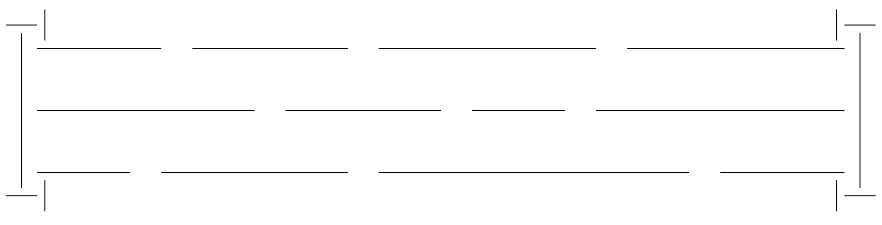
A very similar problem is **General Position**, in which one is given n arbitrary points and has to decide whether any three are collinear. For an instance S of 3-Sum, create an instance P of General Position by projecting the numbers s_i onto the curve $y = x^3$, that is, $P = \{(a, a^3) \mid a \in S\}$.

Suppose three of the points, say, (a, a^3), (b, b^3), and (c, c^3) are collinear. This is the case if and only if the slopes of the lines through each pair of them are equal. (Observe that a, b, and c are pairwise distinct.)

$$\begin{aligned}
(b^3 - a^3)/(b - a) &= (c^3 - b^3)/(c - b) \iff \\
b^2 + a^2 + ab &= c^2 + b^2 + bc \iff \\
b &= (c^2 - a^2)/(a - c) \iff \\
b &= -(a + c) \iff \\
a + b + c &= 0 .
\end{aligned}$$

Minimum Area Triangle is a strict generalization of General Position and, therefore, also 3-Sum-hard.

In **Segment Splitting/Separation**, we are given a set of n line segments and have to decide whether there exists a line that does not intersect any of the segments but splits them into two non-empty subsets. To show that this problem is 3-Sum-hard, we can use essentially the same reduction as for GeomBase, where we interpret the points along the three lines $y = 0$, $y = 1$, and $y = 2$ as sufficiently small "holes". The parts of the lines that remain after punching these holes form the input segments for the Splitting problem. Horizontal splits can be prevented by putting constant size gadgets somewhere beyond the last holes, see the figure below. The set of input segments for the segment



splitting problem requires sorting the points along each of the three horizontal lines, which can be done in $O(n \log n) = o(n^2)$ time. It remains to specify what "sufficiently small" means for the size of those holes. As all input numbers are integers, it is not hard to see that punching a hole of $(x - 1/4, x + 1/4)$ around each input point x is small enough.

In **Segment Visibility**, we are given a set S of n horizontal line segments and two segments $s_1, s_2 \in S$. The question is: Are there two points, $p_1 \in s_1$ and $p_2 \in s_2$ which can see each other, that is, the open line segment $\overline{p_1 p_2}$ does not intersect any segment from S? The reduction from 3-Sum is the same as for Segment Splitting, just put s_1 above and s_2 below the segments along the three lines.

In **Motion Planning**, we are given a robot (line segment), some environment (modeled as a set of disjoint line segments), and a source and a target position. The question is: Can the robot move (by translation and rotation) from the source to the target position, without ever intersecting the "walls" of the environment?

To show that Motion Planning is 3-Sum-hard, employ the reduction for Segment Splitting from above. The three "punched" lines form the doorway between two rooms, each modeled by a constant number of segments that cannot be split, similar to the boundary gadgets above. The source position is in one room, the target position in the other, and to get from source to target the robot has to pass through a sequence of three collinear holes in the door (suppose the doorway is sufficiently small compared to the length of the robot).

Questions

34. *What is the endpoint visibility graph for a set of disjoint line segments in the plane and how can it be constructed?* Give the definition and explain the relation to shortest paths. Describe the $O(n^2)$ algorithm by Welzl, including full proofs of Theorem 12.1 and Theorem 12.4.

35. *Is there a subquadratic algorithm for General Position?* Explain the term 3-Sum hard and its implications and give the reduction from 3-Sum to General Position.

36. *Which problems are known to be 3-Sum-hard?* List at least three problems (other than 3-Sum) and briefly sketch the corresponding reductions.

References

[1] Jeff Erickson, Lower bounds for linear satisfiability problems, *Chicago J. Theoret. Comput. Sci.* **1999**, 8.

[2] A. Gajentaan and M. H. Overmars, On a class of $O(n^2)$ problems in computational geometry, *Comput. Geom. Theory Appl.* **5** (1995), 165–185.

[3] Emo Welzl, Constructing the visibility graph for n line segments in $O(n^2)$ time, *Inform. Process. Lett.* **20** (1985), 167–171.