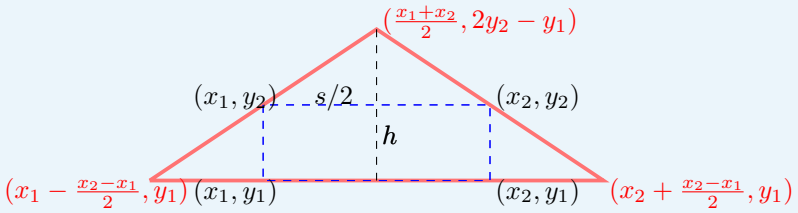
Problem 1. Let P be a set of n points in $\mathbb{R}^2$. Describe how to compute in $O(n)$ time a triangle that includes all the points of P in the interior.

Solution. Let $x_{\min}, x_{\max}, y_{\min}, y_{\max}$ be the minimum x-coordinate, maximum x-coordinate, minimum y-coordinate, maximum y-coordinate in the set of points P, respectively.

We can build a axis-parallel rectangle that cover all points of P, with coordinates $(x_{\min}, y_{\min}), (x_{\max}, y_{\min}), (x_{\min}, y_{\max}), (x_{\max}, y_{\max})$.



Consider a construction of the triangle having the upper-boundary of the rectangle as its midline. We can verify that the assignment of triangle coordinates in the figure covers all four vertices of the rectangle on its boundary, by simply calculating the midpoint coordinates of each edge.

Now, a triangle with coordinates $(\frac{x_{\min}+x_{\max}}{2}, 2y_{\max} - y_{\min}), (x_{\min} - \frac{x_{\max}-x_{\min}}{2}, y_{\min}), (x_2 + \frac{x_{\max}-x_{\min}}{2}, y_{\min})$ satisfies the requirement.

Problem 2. Let $G = (V, E)$ be a connected regular straight-line planar graph (SLPG) with $n = |E|$ segments. Explain how to compute in $O(n)$ time a triangulated SLPG $G' = (V', E')$ such that

- V' includes all the vertices of V, plus three dummy vertices that determine a triangle covering all the points of V in interior;
- $E \subseteq E'$;
- Every face of G' is covered by a face of G.

Solution. First, compute a triangle described in Problem 1 that can bound all points in G.

Now, we charge for every face the number of diagonals added. The number of diagonals added is at most $O(n)$ as proved in Problem 4. There are some sophisicated triangulation algorithms (e.g. Chazelle) which triangulates any polygon in $O(n)$ time, where n is the number of diagonals added.

It remains to argue that we can triangulate the area between the bounding triangle and the original planar graph (where the original planar graph is treated as a hole now). To remove this constraint, we can break the area into two pieces by adding a segment connecting the topmost vertex of the bounding triangle and the topmost vertex on the original graph (which can be found in $O(n)$ time), and another segment connecting one of the other two vertex to the lowest vertex in the original graph. Note that the latter segment must not cross any existing edges since the point getting connected to is the lowest point in the original graph and the y-coordinate range of this segment does not cross that of the former segment added inside this area.

Problem 3. Let G be a connected regular SLPG with n segments. Describe how to build the point-location structure we discussed in $O(n \log n)$ time.

Solution. First we obtain the triangulated SLPG in $O(n)$ time as described in Problem 2, with $O(n)$ vertices.

Lemma: A triangulated SLPG with n vertices admits an independent set with at least $n/48$ vertices.

We start with the initial triangulated SLPG, label all the faces, and call this structure T_0. We construct $T_1, T_2, \cdots, T_k$ (where k is at most $O(\log n)$ as proven by the lemma) as follows:

- Select an independent set of vertices (excluding the bounding triangles) with degree at most 11.
- Remove these vertices and generate a new structure T_{k+1};
- Retriangulate the holes of T_{k+1} in $O(n)$ time.

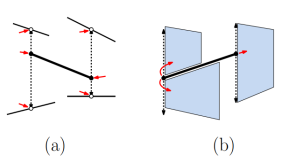
Each node removed corresponds to a new triangle in triangulation T_{k+1}, we store pointers to the triangles which it overlappes in T_k. Since there are at most 11 of these, the structure has bounded degree.

Problem 4. Prove: If a triangulated SLPG has n vertices and m edges, it must hold that $m = 3n - 6$.
Hint: First prove that there must be (triangle) face that contains only one vertex in the interior. Then, apply induction.

Proof. Every triangular face contains three edges. Every edge bounds two faces. Therefore, $2E = 3F$. By Euler's Formula, for any connected planar graph, we have $V - E + F = 2$. Therefore, it is easy to see that $E = m = 3n - 6$. □

Problem 5. Prove: The trapezoidal map defined by n non-intersecting line segments in $\mathbb{R}^2$ has complexity $O(n)$.

Proof. Each segment results in 6 vertices in the map, two from the endpoints of the segment, two from the upward vertical rays, and two from the downward vertical rays. There are in addition 4 more vertices for the bounding rectangle, which results in $6n + 4$ vertices in the map in total.



We charge each trapezoid to two segments belonging to the original line segments. Each segment can at most be charged by three trapezoids, since the left endpoint of each segment is charged twice, one from the trapezoid above-right and one from the trapezoid below-right, and the right endpoint is charged once by the trapezoid to its right. This yields a total of $3n$ trapezoids. There is one more trapezoid, namely the leftmost one, for a total of $3n + 1$. □

Problem 6. Describe an algorithm to build the trapezoidal map from n non-intersecting line segment in $\mathbb{R}^2$ using $O(n \log n)$ time.

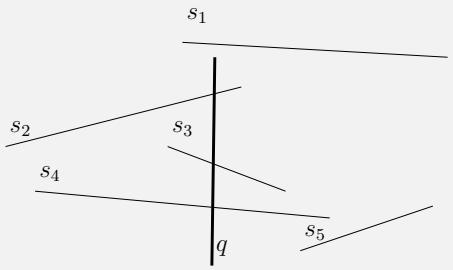
Solution. First we sort all segments with respect to the x-coordinates of their left endpoints, which can be done in $O(n \log n)$ time.

Sweep a line from $x = -\infty$ to $x = +\infty$. Use a variant of the line segment intersection theorem. Maintain a BBST for the relative ordering of the segments stabbed by the sweepline, storing the line equation of these segments, then each insertion/deletion/predecessor/successor query can be answered in $O(\log n)$ time.

- Whenever we encounter a left endpoint of a segment, find the segment immediately above/below this point to identify the two edges and vertices shot by the ray, insert this segment to the BBST;
- Whenever we encounter a right endpoint of a segment, also find the segment above/below this point and identify the two new edges and two new vertices, delete this segment from BBST.

The algorithm is done in $O(n \log n)$ time as required.

Problem 7. Let S be a set of n non-intersecting line segments in $\mathbb{R}^2$. Given a vertical segment q, a query retrieves all the segments of S intersecting q. Design a data structure of $O(n)$ space that answers a query in $O(\log n + k)$ time, where k is the number of segments reported. In the following example where $S = \{s_1, s_2, \ldots, s_5\}$, the query q retrieves $k = 3$ segments.



Solution. First, the trapezoidal map of S with n segments can be stored with $O(n)$ space, represent it with a DCEL structure.

Start the exploration from the lowest point of q and use $O(\log n)$ time to locate the face where this lowest point is located. Note that every trapezoid has at most $O(1)$ edge, and thus we can judge the edge on the trapezoid intersecting with q in $O(1)$ time. We repeat the process as described below:

- Find an edge which has not been reported of the current trapezoid intersecting with q and terminate if there is no such edge, this operation requires at most $O(1)$ time;
- Report the segment intersecting with q;
- Traverse to the opposite face (this operation is supported by DCEL, bounding face of the `Twin` edge).

The process can take at most $O(k)$ time, where k is the number of segments reported.

The total running time of this algorithm is $O(\log n + k)$.