Chapter 6

Visibility

Let S be a set of n line segments in the plane (which may or may not intersect). We consider these segments as being opaque, which gives rise to the notion of visibility. We say that two points p and q are *visible* if the line segment from p to q intersects none of the given (opaque) segments.

6.1 Visibility from a Point

In this section we consider the problem where the input set of line segments defines a simple polygon P, and we wish to determine the region in the interior of P that is visible from some distinguished point p. We begin by triangulating the interior of P and identifying the triangle containing p. We then incrementaly consider the triangles adjacent to this starting triangle. Inductively, we assume that we have determined the visibility from p to an edge e on the boundary of the triangle t we are about to consider. Since P is simple, the visibility of e with respect to p is an interval $e' \subseteq e$. In considering the triangle t we extend the "cone" defined by p and the interval e' into the triangle t. This defines new visible intervals on the other two edges of t, which, if they are not edges of P, we an then cross into additional triangles to consider. Since we can consider each triangle iteratively in this way in $O(1)$ time each, this implies that the total time for computing the visible region from p is $O(n)$ plus the time to triangulate P, which can also be done in $O(n)$ time using an algorithm of Chazelle.

6.2 Upper Envelopes

The above method does not easily generalize to the case when the input opaque segments do not form a simple polygon and possibly even intersect. In this case it is more natural to consider the construction of the visible region from a point p to be an *upper envelope* problem. In this general framework we are given a collection $\mathcal{F} = \{f_1, f_2, \ldots, f_n\}$ of functions from $\mathcal{R}^1$ to $\mathcal{R}^1$. We define the *upper envelope* of $\mathcal{F}$ to be the function f defined

$$f(x) = \max_{f_i \in \mathcal{F}} \{f_i(x)\}.$$

In geometric settings it is quite common for the functions in $\mathcal{F}$ to possess a nices crossing property where it is known that any two such functions cross at most k times. It is natural, then, to study the algorithmic question of how one can efficiently construct a representation of the upper envelope function f, as well as the combinatoric question of how large such a representation must be. The most standard way of representing this function is as a list of intervals of $\mathcal{R}^1$ together with the the indices of the functions in $\mathcal{F}$ that realize the maximum in each interval.

The crossing property defined above gives rise to a combinatorial notion defined on character strings caled the *Davenport-Schinzel* sequences. Such a sequence is defined by two parameters: n

and k, where n is the number of characters and k is the maximum number of altnernations that can occur between any two characters. That is, two characters a and b can occur in such a sequence as ...a...b...a...b... with the number of a's and b's is $k+1$, but there is no such subsequence of $k+2$ a's and b's. In addition, no character is allowed to be repeated twice in a row. Letting $\lambda_k(n)$ denote the maximum length of a Davenport-Schinzel sequence with parameters n and k, it is one of the most interesting recent results of combinatorics that, for any fixed value of $k \geq 3$, that $\lambda_k(n)$ is $O(n \log^* n)$. This parameter is linear for smaller values of k.

By a simple mergesort-like divide-and-conquer algorithm, then, we can construct the upper envelope of $\mathcal{F}$ in $O(\lambda_k(n) \log n)$ time. For example, computing the upper envelope of a collection of functions defined by line segments in the plane can be done in $O(\lambda_3(n) \log n) = O(n \log n \log^* n)$ time. Equivalently, the region of the plane visible from a point can also be computed in this time.

6.3 Visibility from an Edge

Let us now consider another type of visibility: *weak visibility* from an edge. In this context we are still given a set of opaque segments, but instead of a point source of visibility we are now given an edge e. We say that a point q is weakly visible from e if there exists a point p on e such that p and q are visible. In this section we consider the problem of computing the region inside a simple polygon P that is weakly visible from a distinguished edge e on P.

6.3.1 Centroid edges

Before we describe how to efficiently solve this problem we first discuss a seemingly unrelated problem. In this problem one is given an n-node binary tree T and asked to find an edge e of T that partitions T into two subtrees T_1 and T_2 such that $n/3 \leq |T_1|, |T_2| \leq 2n/3$. This edge can be found by a simple greedy process that determines the edge that minimizes the maximim size of the subtrees defined by removing that edge from T. To compute the size of each such subtree we can preproces the tree T by computing an Euler tour defined by "marching" around the outside of the tree as an embedded planar graph. If we assign the first visit of a vertex in this tour a $+1$ and each other visit a 0, and then comput al the partial sums in this sequence, we can easily locate the centroid edge in $O(n)$ time.

6.3.2 Divide-and-conquer visibility from an edge

Let us return to the visibility from an edge problem. We begin our algorithm by triangulating P, as in our point visibility algorithm. We note that if we exclude the exterior of P, then the graph-theoretic planar dual to this trianglation defines a binary tree T. Moreover, by locating the centroid edge f in this tree, we identify a diagonal (dual to this edge) that separates the polygon P into two subpolygons P_1 and P_2 each of size at most two-thirds the size of P. The idea is to recurse on each of P_1 and P_2. Of course, one of these subpolygons, say, P_1, contains the distinguished edge e defining the visibility. For the polygon P_2 we use the edge f for this role. But before we can use recursion on P_1 and P_2 we must precisely define the problem we are recursively solving. Specifically, we define the problem as that of computing in the dual plane (under point-line duality) the subdivision defined by determining for each edge h of P the set of lines (represented as points in the dual plane) that contain lines of sight from h to the distinguished edge. Given such subdivisions for P_1 and P_2, we note that if f is weakly visible from e, then there will be a region associated with f in the dual subdivision defining the visibility from e in P_1. In this case we complete the algorithm by "clipping" the subdivision defining the weak visibility from f in P_2 to the region for f in the subdivision defining the weak visibility from e. This can be done in $O(n)$ time and gives us a representation of the weak visibility from e of P. The total time for this algorithm is $O(n \log n)$.

6.4 The Visibility Graph

This approach can also be used to construct the graph of all visibilities between pairs of vertices in a simple polygon P. The idea is to again determine a centroid diagonal edge f that separates P into subpolygons P_1 and P_2 each of size at most two-thirds that of P. We then recurse on P_1 and P_2, determining all the visibility pairs of vertices in each. What remains, then, is to compute all the visibility pairs that cross from P_1 to P_2 through the edge f. To do this we compute the visibility subdivisions from f with repsect to P_1 and P_2, and then compute all the pairs of intersecting segments between these two subdivisions. Each such interesection corresponds (in the primal) to a line of sight between two vertices, one in P_1 and the other in P_2. Since we can compute all intersecting pairs in $O((n+k)\log n)$ time, where k is the number intersections, the total time for this computation is $O(n\log^2 n + k\log n)$. In fact, using the segment intersection algorithm of Chazelle and Edelsbrunner, we can reduce this to $O(n\log^2 n + k)$ total time.