When you turn the corner
And you run into yourself
Then you know that you have turned
All the corners that are left.
— Langton Hughes, "Final Curve" (1951)

3 Testing Homotopic Paths in the Plane

In the previous lecture, we saw an algorithm to compute the shortest path in a polygon with holes that is homotopic to a given path. Here we consider an apparently easier problem: Given two paths in P, are they homotopic? Or equivalently, given a loop ℓ in P, is it contractible. We can solve both problems by computing the reduced crossing sequence(s) of the input path(s). Two paths are homotopic if and only if they have the same reduced crossing sequence, and a loop is contractible if and only if its reduced crossing sequence is the empty string. We can test both of these conditions in time $O(nk)$ using the first two phases of our previous algorithm.

In this lecture, I'll describe a faster algorithm for both of these problems for *simple* paths and loops, originally due to Cabello *et al.* [4], with some simplifications by Efrat *et al.* [6]. Most of the lecture will concentrate on testing whether a simple loop in a polygon with holes is contractible. We will briefly return to testing homotopy of simple paths at the end.

3.1 Sentinel Points

If the number of holes in P is small, we can test contractibility more quickly by applying a little more topology. Let $S = \{s_1, s_2, \ldots, s_h\}$ denote a set of h *sentinel* points, one chosen arbitrarily inside each hole of P. Because P is a proper subset of the space $\mathbb{R}^2 \setminus S$, any function from $[0,1]^2$ to P is also a function from $[0,1]^2$ to $\mathbb{R}^2 \setminus S$. It follows that any contractible loop in P is also contractible in $\mathbb{R}^2 \setminus S$. In fact, the converse is true as well.

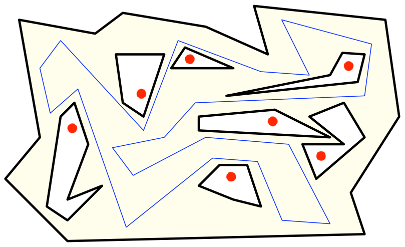
Lemma 3.1. *Let ℓ be a loop in P. If ℓ is contractible in $\mathbb{R}^2 \setminus S$, then ℓ is contractible in P.*

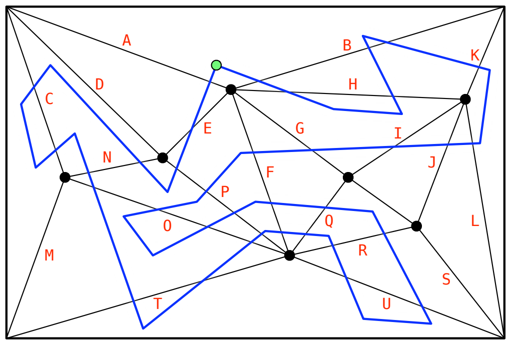
Proof: Let P_0 denote the outer boundary of P, and let $P_1, \ldots, P_h$ denote the boundaries of its holes, indexes so that each sentinel point s_i lies in the interior of the corresponding polygon P_i. The Jordan-Schönflies theorem implies that for each i, there is a homeomorphism $\phi_i \colon \mathbb{R}^2 \to \mathbb{R}^2$ such that the restriction of ϕ_i to S^1 is the cycle P_i. Without loss of generality, we can assume that $\phi_i(0) = s_i$, where 0 denotes the origin. Let $u \colon \mathbb{R}^2 \setminus 0 \to S^1$ be the function $u(x) = x/\|x\|$; this function is clearly continuous. Then the function $\Phi_i = \phi_i \circ u \circ \phi_i^{-1}$ maps $\mathbb{R}^2 \setminus s_i$ continuously onto P_i. Finally, let $\Phi \colon \mathbb{R}^2 \setminus S \to P$ denote the function

$$\Phi(x) = \begin{cases} x & \text{if } x \in P, \\ \Phi_0(x) & \text{if } x \text{ is outside } P_0, \\ \Phi_i(x) & \text{if } x \text{ is inside } P_i. \end{cases}$$

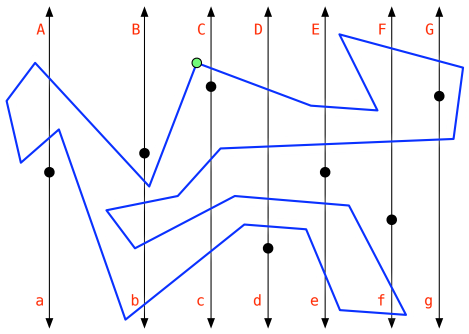
This function is continuous and obviously fixes P. Thus, for any homotopy $h \colon [0,1]^2 \to \mathbb{R}^2 \setminus S$ between two paths in P, the function $\Phi \circ h \colon [0,1]^2 \to P$ is a homotopy in P between the same two paths. □

Thus, to test the contractibility of a loop in P, it suffices to work in the simpler space $\mathbb{R}^2 \setminus S$. We can construct a 'triangulation' of $\mathbb{R}^2 \setminus S$ (or more accurately, a triangulation of $R \setminus S$ where R is a sufficiently large rectangle) in $O(h \log h)$ time. Any loop composed of k segments crosses the edges of this triangulation at most $O(hk)$ times, so we can compute the reduced crossing word of any loop in


A polygon with holes P and a set S of sentinel points.



A triangulation of $R \setminus S$.

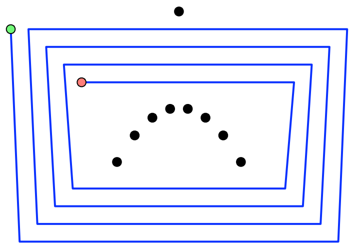


A division of $\mathbb{R}^2 \setminus S$ into vertical slabs.

$O(hk)$ time. If the number of holes is small, this is considerably faster than our previous $O(nk)$ time bound.

We can further simplify the algorithm by subdividing the plane with vertical lines through the points in S instead of a triangulation. To keep things simple, we assume no two sentinel points have the same x-coordinate, so each sentinel divides a vertical line into two infinite rays; we give each ray a unique label. We can still compute crossing words in time $O(k + x) = O(kh)$, and we have the same correspondence between equivalent crossing words and homotopic paths.

For many triangulations, and for all sets of vertical lines, there are infinite families of paths that achieve the worst-case bound $x = \Theta(hk)$. For example, consider a path that spirals around all the sentinel points $\Theta(h)$ times, using a constant number of segments per turn. Unfortunately, it is not even possible to avoid this worst-case behavior by choosing the triangulation carefully, or even using another subdivision of $\mathbb{R}^2 \setminus S$ into constant-complexity polygons with all vertices in (or very far away from) S. (The last two conditions are required by our algorithm.)


A path with a long reduced crossing word, for *any* suitable subdivision.

For the rest of the lecture, to be consistent with the literature (and common practice), I'll denote the number of sentinel points by n, instead of h.

3.2 Rectifying Simple Loops

Let $S = \{p_1, \ldots, p_n\}$ be a set of points in the plane, and let ℓ be a simple polygonal loop with k edges in $\mathbb{R}^2 \setminus S$. In the next two sessions, I'll describe an algorithm to determine whether ℓ is contractible in $\mathbb{R}^2 \setminus S$ in $O((n+k)\log(n+k))$ time. This problem is equivalent to checking whether any point in S lies in in the interior of ℓ, and thus can be solved more efficiently by other means, but the algorithm we describe here generalizes to the harder problem of checking whether two simple paths are homotopic.

To simplify the algorithm, assume the points in S and vertices of ℓ all have distinct x-coordinates. The contractibility algorithm begins by transforming S and ℓ into a simpler but equivalent form; Cabello *et al.* refer to this process as *rectification*.

A path in the plane is ***monotone*** if it intersects any vertical line at most once, or equivalently, if it is the graph of a function over some real interval. Our algorithm begins by subdividing ℓ into a sequence of monotone paths. Because ℓ is a loop, we have $m \geq 2$, and because each monotone path uses at least one edge of ℓ, we have $m \leq n$.

Say that a monotone path α is ***directly above*** a path β if there is a vertical line segment of non-zero length whose top endpoint lies on α and whose bottom endpoint lies on β. We can similarly define when a point is directly above a monotone path or vice versa, by considering the point to be a zero-lenght monotone path. We write $\boldsymbol{\alpha \succ \beta}$ to denote that α is directly above β.

Lemma 3.2. *For any set of disjoint monotone paths (and points), the relation $\succ$ is acyclic.*

Proof: For the sake of contradiction, let $\alpha_1 \succ \alpha_2 \succ \cdots \succ \alpha_r \succ \alpha_1$ be a minimum-length cycle, where each α_i is a monotone path in the set. Obviously $r \geq 2$.

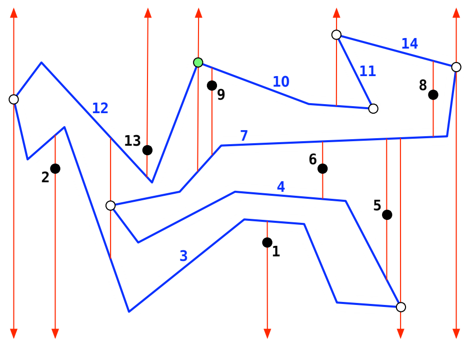
Think of each α_i as the graph of a function over some real interval. Because $\alpha_1 \succ \alpha_2$, we have $\alpha_1(x) > \alpha_2(x)$ for some real x. If $\alpha_2 \succ \alpha_1$, then $\alpha_1(z) < \alpha_2(z)$ for some real z, so by the intermediate value theorem, there must be some real y between x and z such that $\alpha_1(y) = \alpha_2(y)$. But this is impossible, because α_1 and α_2 are disjoint. We conclude that $\alpha_2 \not\succ \alpha_1$, which means $r \geq 3$.

Rotate the indices if necessary, so that the right endpoint of α_1 has smaller x-coordinate that the right endpoint of any other path α_i. In particular, the right endpoints of α_2 and α_r are completely to the right of α_1. Thus, the right endpoint of α_1 is both directly above some point of α_2 and directly below some point of α_r. It follows that $\alpha_r \succ \alpha_2$, which contradicts the minimality of the cycle. □

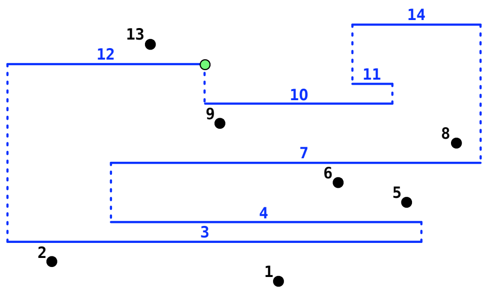
We can construct a directed graph consistent with $\succ$ by placing a vertical line segment, called a ***fence***, through each endpoint or sentinel point, which is as long as possible without crossing any other path. (If no path lies directly above or below an endpoint, that endpoint's fence is either a ray or a line.) The paths and fences partition the plane into simply-connected regions bounded by at most two paths (above and below) and at most two fences (on the left and right). These $O(n+k)$ fences can be found in $O((n+k)\log(n+k))$ time using a straightforward sweep-line algorithm. If k is significantly larger than n, this time bound can be improved to $O(k + n\log^{1+\varepsilon} n)$ for any $\varepsilon > 0$, using an algorithm of Bar-Yehuda and Chazelle [1].

Let us write $\alpha \rhd \beta$ to denote that some fence touches both α and β, and the fence point on α is above the fence point on β. An easy inductive argument implies that the relations $\succ$ and $\rhd$ have the same transitive closure. We can easily extract the relation $\rhd$ directly from the fence decomposition, after which a simple topological produces a vertical ranking of the paths and sentinel points so that $\alpha \succ \beta$ implies $\text{rank}(\alpha) > \text{rank}(\beta)$.

Finally, we define a set of ***rectified*** paths and points by replacing all y-coordinates with vertical ranks. Thus, for each monotone path α_i, the corresponding rectified path $\bar{\alpha}_i$ is a horizontal line segment whose y-coordinate is the path's vertical rank, and whose endpoints have the same x-coordinates as the



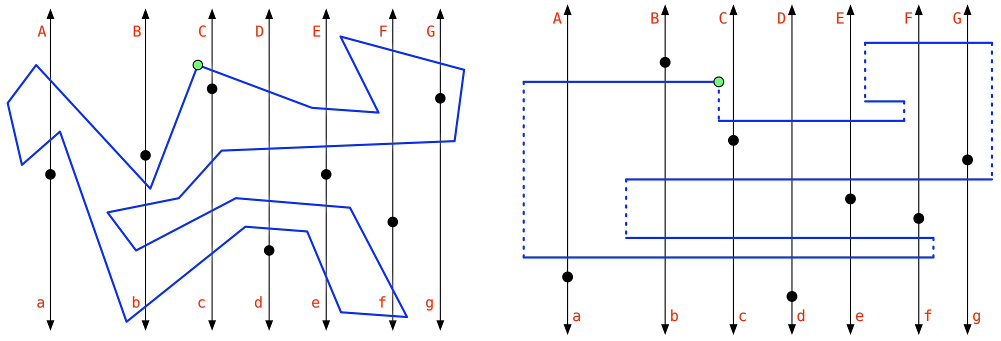
A fence decomposition of the example loop.
Circles are endpoints of monotone paths. Numbers indicate vertical ranks.



The resulting rectified paths and points.

endpoints of α_i. Whenever two paths α_i and α_j share and endpoint, we connected the corresponding endpoints of $\bar{\alpha}_i$ and $\bar{\alpha}_j$ with a vertical segment (dashed in the figures).

Let $\bar{S}$ denote the set of rectified sentinel points, and let $\bar{\ell}$ denote the loop composed of rectified paths and vertical connectors. It is not hard to prove that the original loop ℓ is contractible in $\mathbb{R}^2 \setminus S$ if and only if the rectified loop $\bar{\ell}$ is contractible in $\mathbb{R}^2 \setminus \bar{S}$. Moreover, if we define crossing sequences in terms of vertical rays shot up and down from each sentinel point, then ℓ and $\bar{\ell}$ have identical crossing sequences in their respective environments.



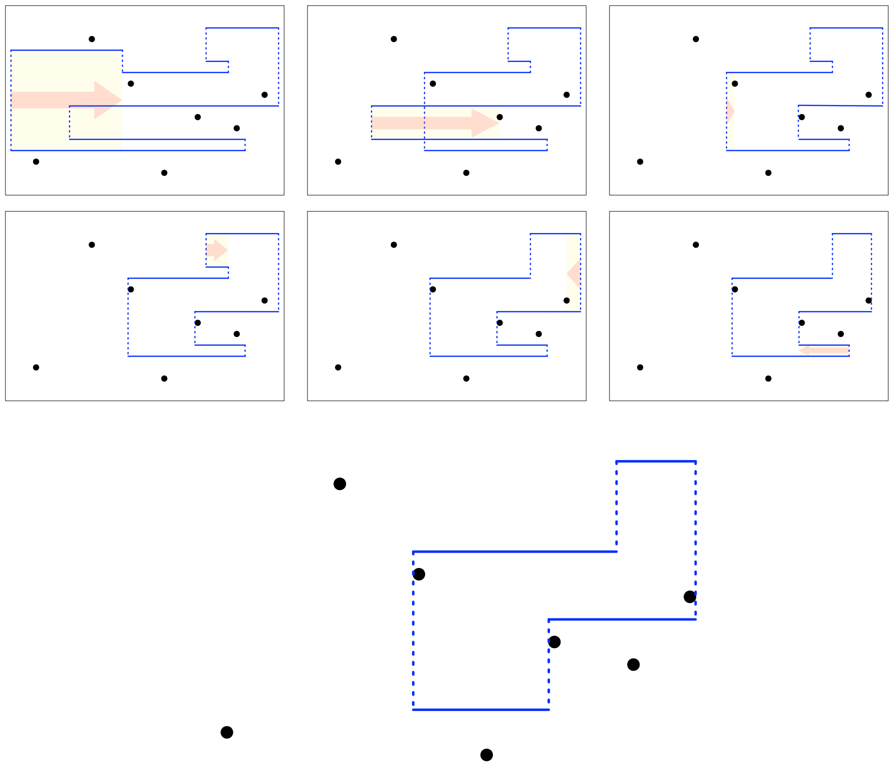
The original loop and the rectified loop have the same crossing sequence: `bAAbcDeffeDcbbcDEFgGFEDC`.

3.3 Reducing Rectified Paths

After we compute the rectified loop, we ***reduce*** it by moving vertical segments. In any orthogonal loop, there are two types of vertical segments: *brackets*, where the adjacent rectified paths are on the same side, and *steps*, where the adjacent rectified paths are on opposite sides. Initially, every vertical segment in the rectified loop is a bracket, but the reduction process can introduce steps.

To reduce the rectified loop, we ***slide*** each bracket as far inward as far as possible, shortening its adjacent rectified paths. We stop sliding either when the bracket reaches a sentinel point, or the bracket reaches the basepoint,[1] or when one of the adjacent paths collapses to a point. Sliding a bracket inward reduces the crossing sequence of the loop, and therefore preserves the contractibility of the loop. Conversely, any repeating pair can be removed from the crossing sequence by a sequence of bracket slides.

A bracket slide that collapses a path to a point can merges two vertical segments, or temporarily create a zero-width vertical spike (which we remove). However, we must be careful *not* to carelessly remove spikes that contain sentinel points, or to disconnect the loop from the basepoint. Thus, if a bracket slide ends at a sentinel point, we record whether the bracket lies just to the right or just to the left of that point. We also allow at most one vertical spike, whose tip is the basepoint of the loop.



A sequence of bracket slides and the final reduced rectified loop.

To determine efficiently how far to slide each bracket, we use a so-called *segment-dragging* data structure due to Chazelle [5]. Chazelle's data structure stores any set of n points in the plane, in $O(n)$ space, so that given a vertical query segment, the leftmost point (if any) to the right of the segment can be reported in $O(\log n)$ time; the data structure can be constructed in $O(n \log n)$ time. Each bracket slide either moves a bracket up against a point or deletes a horizontal segment. Thus, the cycle is completely reduced after at most $2k$ bracket slides. We conclude that the total running time of the reduction algorithm is $O((n+k)\log n)$.

We claim that the reduced rectified loop is empty if and only if the input loop is contractible. The forward implication is trivial; the sequence of bracket slides is a homotopy. Suppose the input lop is contractible. Then the crossing sequence of the reduced loop is empty, which means the reduced rectified

[1]In fact, if we only want to check whether the loop is contractible, we can ignore the basepoint and treat the loop as a cycle. A loop is contractible if and only if the corresponding cycle is *freely* contractible.

loop lies between two vertical lines through sentinel points. In particular, there are no sentinel points in the bounding box of the reduced rectified loop. The leftmost and rightmost vertical edges of any orthogonal loop are brackets; if the reduce loop is non-empty, we can slide at least one of these brackets inward. Thus, the reduced loop must be empty.

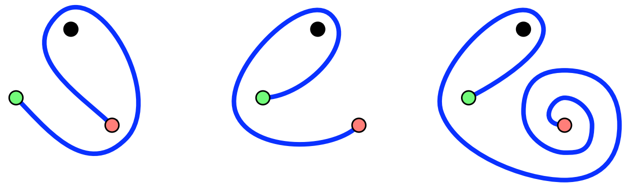
3.4 Detecting Homotopic Simple Paths

This algorithm can be modified to determine whether two simple paths π and π' are homotopic in the same $O((n+k)\log(n+k))$ running time. Most of these modifications are easy, but we do have to address a few subtleties. Most importantly, we have (at least) three choices for the precise definition of 'homotopy', depending on the desired behavior at the endpoints of π and π', each of which leads to different results. (All three variants are equivalent for the contractibility problem.)

- **Tacks:** The endpoints of π and π' are not obstacles; during a homotopy, the moving path can pass freely over the endpoints. This is arguably the most natural variant, but (surprisingly) it presents the most technical difficulty in the algorithm. I'll describe the algorithm for this variant last.

- **Pins:** The endpoints of π and π' are sentinel points in S; during a homotopy, the moving path must never pass over either endpoint. This is the simplest case to deal with algorithmically; however, according to our existing definitions, this case cannot happen at all! We can generalize our definitions as follows to avoid the discrepancy. A *legal* path is a function $\pi\colon [0,1] \to \mathbb{R}^2$ such that $\pi(t) \in S$ implies $t = 0$ or $t = 1$, and a *legal* homotopy is a homotopy through legal paths. Now we can formally state the problem as follows: Given two simple legal paths, are they legally homotopic? I'll describe the algorithm for this variant in the next subsection.

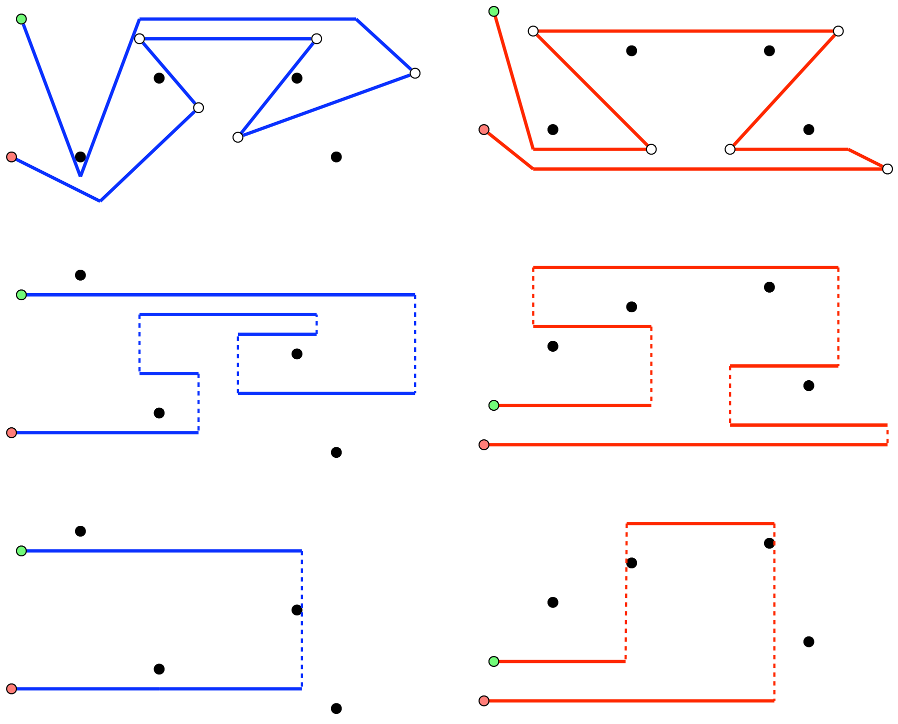
- **Pushpins:** The endpoints of π and π' lie on the boundary of P; during a homotopy, the moving path must never pass over the holes, and therefore cannot pass over the endpoints of π. This case differs from the previous one in how paths that wind around their endpoints are classified. A path that winds multiple times around an *endpoint* is (legally) homotopic to a path that does not, but a path that winds several times around a *hole* is not homotopic to a path that does not. Nevertheless, we can reduce the pushpin model to the pin model, by replacing each hole P_i (*including* the exterior of the outer boundary) with a *pair* of sentinel points s_i and s_i'. If the endpoints of π and π' lie on boundaries P_i and P_j, we add segments to the ends of π and π', extending them to corresponding sentinel points s_i and s_j.


Three paths in the plane minus points. All three paths are homotopic in the tack model.
Only the last two are homotopic in the pin model. Every pair is non-homotopic in the pushpin model.

3.4.1 Pins

First let's consider the pin model. Given two simple legal paths π and π', the algorithm starts by independently rectifying and reducing each path. We must make one change to the reduction algorithm to ensure that the reduced paths are simple. The only way to introduce a self-intersection (in the pin model) is to slide a bracket over another smaller bracket with the same handedness. Thus, if we always slide either the rightmost free left bracket or the leftmost free right bracket, the path remains simple throughout the reduction process. By keeping brackets in a priority queue, we can select an extreme bracket in $O(\log k)$ time, so this restriction does not change the algorithm's running time.

Now we must compare the reduced rectified paths. Unfortunately, even if the input paths are homotopic, their reduced rectified paths be very different, because they were reduced with respect to different vertical rankings. However, if $\pi \simeq \pi'$, both rectified reduced paths are consistent with the shortest path π^* in their homotopy class. Thus, we can compare the reduced rectified paths as follows. First, we check that both paths have the same sequence of bracket x-coordinates; if not, we report that π and π' are not homotopic. Next, we compute a fence decomposition of each rectified reduced path and its associated sentinels, using the same sweep algorithm as before. If the two fence decompositions are combinatorially identical, then the two reduced rectified paths have a common re-rectification and therefore equal crossing sequences, so the original paths π and π' are homotopic. Otherwise, $\pi \not\simeq \pi'$.


Two homotopic paths, rectified and then reduced.

3.4.2 Tacks

In the peg model, it may be impossible to avoid self-intersections; the shortest path homotopic to a simple path is not necessarily simple. In this setting, if we only slide extreme brackets, as in the previous section, the only way to introduce a self-intersection is to slide a bracket over one of the endpoints. Since each vertical segment can cross each endpoint at most once, each reduced rectified path crosses itself at most $2k$ times.

These self-intersections prevent us from using the same straightforward sweepline algorithm to compute a common 'above' relation for the reduced rectified paths; the relation $\prec$ we defined earlier may not be a partial order. However, we can still compare reduced rectified paths by restricting $\prec$ to a *bipartite* relation between sentinel points and monotone subpaths.

Lemma 3.3. *Paths π and π' are homotopic if and only if (1) the reduced rectified paths have the same sequence of bracket x-coordinates, and (2) corresponding monotone subpaths lie above corresponding sentinel points.*

Proof: The reduced rectified paths $\hat{\pi}$ and $\hat{\pi}'$ are rectifications of the shortest paths $\bar{\pi}$ and $\bar{\pi}'$ homotopic to π and π', respectively. Suppose $\pi \simeq \pi'$. Then $\bar{\pi} = \bar{\pi}'$, which implies that the maximal monotone subpaths of $\hat{\pi}$ and $\hat{\pi}'$ end at the same sequence of x-coordinates, so condition (1) is satisfied. Similarly, $\hat{\pi}$ and $\hat{\pi}'$ have the same crossing sequences with respect to their respective rectified sentinel points $\hat{S}$ and $\hat{S}'$. In particular, each monotone subpath of $\hat{\pi}$ has the same crossing sequence as the corresponding subpath of $\hat{\pi}'$, so condition (2) is satisfied.

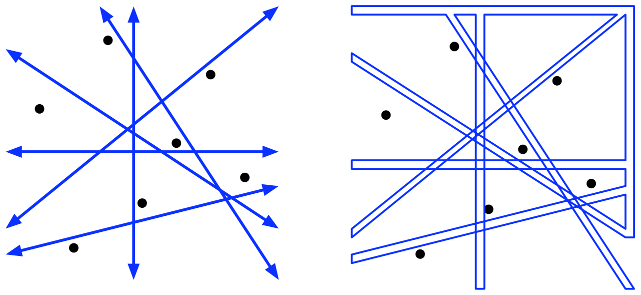
Conversely, if conditions (1) and (2) are satisfied, then each monotone subpath of $\hat{\pi}$ has the same crossing sequence as the corresponding subpath of $\hat{\pi}'$, which implies that $\hat{\pi}$ and $\hat{\pi}'$ have the same crossing sequences, which implies that $\pi \simeq \pi'$. $\square$

We have already seen that condition (1) can be tested in $O(k)$ time. Cabello *et al.* [4] describe a sweepline algorithm to test condition (2) in $O((n+k)\log(n+k))$ time; we refer the reader to the paper for further details. The fact that each reduced rectified path crosses itself only $O(k)$ times is crucial for the time analysis.

3.5 Non-Simple Paths and Loops

Given two *non-simple* paths in a polygon with holes, it is possible to check whether they are homotopic in $O((n + n^{2/3}k^{2/3} + k)\,\text{polylog}(n+k))$ time, using an algorithm of Bespamyatnikh [2]; see also [3]. This algorithm is always faster than the algorithm of Hershberger and Snoeyink [8] described in the previous lecture. The algorithm is quite complex; I won't even attempt a sketch here.

There is strong evidence that Bespamyatnikh's algorithm is optimal, at least up to polylogarithmic factors, even for the special case of checking whether a cycle is contractible. Cabello *et al.* [4] describe a reduction from *Hopcroft's problem*—Given a set of n points and k lines in the plane, does any point lie on any line?—to testing the contractibility of a non-simple cycle of $O(k)$ edges, in the plane minus a set of n points. Erickson [7] proved a lower bound of $\Omega(n\log k + n^{2/3}k^{2/3} + k\log n)$ on the complexity of Hopcroft's problem in a natural (but restricted) model of computation.


Reducing Hopcroft's problem to the contractibility of a non-simple cycle.

References

[1] R. Bar-Yehuda and B. Chazelle. Triangulating disjoint Jordan chains. *Int. J. Comput. Geom. Appl.* 4(4):475–481, 1994.

[2] S. Bespamyatnikh. Computing homotopic shortest paths in the plane. *J. Algorithms* 49(2):284–303, 2003.

[3] S. Bespamyatnikh. Encoding homotopy of paths in the plane. *Proc. LATIN 2004: Theoretical Infomatics*, 329–338, 2004. Lecture Notes Comput. Sci. 2976, Springer-Verlag.

[4] S. Cabello, Y. Liu, A. Mantler, and J. Snoeyink. Testing homotopy for paths in the plane. *Discrete Comput. Geom.* 31:61–81, 2004.

[5] B. Chazelle. An algorithm for segment-dragging and its implementation. *Algorithmica* 3:205–221, 1988.

[6] A. Efrat, S. G. Kobourov, and A. Lubiw. Computing homotopic shortest paths efficiently. *Comput. Geom. Theory Appl.* 35(3):162–172, 2006.

[7] J. Erickson. New lower bounds for Hopcroft's problem. *Discrete Comput. Geom.* 16:389–418, 1996.

[8] J. Hershberger and J. Snoeyink. Computing minimum length paths of a given homotopy class. *Comput. Geom. Theory Appl.* 4:63–98, 1994.