CS 7301.003.20F Lecture 7—September 9, 2020

Main topics are #homotopy .

Older Online Notes

- Some of you have pointed out that Erickson took down his old notes.
- He's teaching a new version of the class, but there's no guarantee he'll cover things in the same order, and he's pruned a lot of things from his notes.
- I'll look for alternative sources of reading before the next lecture. And I'll continue to put my own notes on the website.

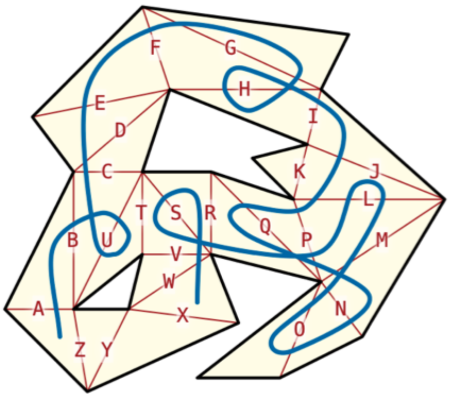
Testing Homotopy

- With another set of fresh definitions out of the way, let's look at a computation problem. We're going to ask whether two polygonal paths are homotopic which is equivalent to asking if a given polygonal loop in a polygon with holes is contractible
- Recall, a *simple polygon* is the closure of the interior of a simple closed polygonal chain. We write partial P to denote its boundary (the polygonal chain) and P circ to denote its interior.
- A *polygon with holes* is a set P = P_0 \ (P_1 circ cup P_2 circ cup … cup P_h circ) where P_0 is a simple polygon and P_1, …, P_h are disjoint simple polygons called the *holes* in the interior of P_0. We call partial P_0 the *outer boundary* and each partial P_i an *inner boundary* of P.
- First off, our problem is trivial if there are no holes.
- Lemma: Let P be an arbitrary simple polygon. Every loop in P is contractible.
    - P is homeomorphic to a triangle by the Jordan-Schönflies theorem, and triangles are convex.
    - Let ell : [0, 1] ➜ Δ be a loop in triangle Δ. Fix a coordinate system with the origin at the basepoint of ell.
    - Homotopy h : [0, 1]^2 ➜ Δ with h(s, t) := (1 - s) ell(t) is a contraction. Use the inverse homeomorphism to turn it into a contraction in P.
- In particular, paths with common endpoints in a simple polygon are always homotopic.
- But when you have holes, two paths may "wind around" the holes in different ways, preventing homotopy.
- Our algorithm takes place over several steps.
    1. Compute a frugal triangulation of the polygon with holes.
    2. Compute the *crossing sequences* of the polygonal paths with the triangle edges.
    3. *Reduce* the crossing sequences by removing pairs of adjacent identical symbols.

4. Report the paths are homotopic if and only if the reduced crossing sequences are identical.

Crossing Sequences

- Let Δ be a frugal triangulation of polygon P with holes and pi be a polygonal path in P. We'll assume pi avoids the vertices of P, but the algorithm can be modified a bit if that's not the case.
- Path pi *crosses* triangulation edge e if a subpath of pi is contained in e and the edges of pi preceding and following the subpath lie on opposite sides of e.
- The *crossing sequence* X_Δ(pi) of pi is the sequence of edges of Δ that pi crosses in order along pi.



A path with crossing sequence ABUUCDEFGGHHHIJLPQQPMNOONMLLPQRSSSVW

- Lemma: Any two paths with the same endpoints and the same crossing sequence are homotopic.
    - We show for any path pi, there is a *canonical path* pi' with the same endpoints and crossing sequence as pi that is homotopic to pi.
    - Let p and q be the endpoints of pi. We go from p to the centroid of its triangle, to the midpoint of the first edge crossed, to the centroid of the next triangle, to the midpoint of the next edge crossed… to the centroid of q's triangle, to q.
    - The homotopy fixes one point of pi per triangle and moves the points between so the path goes through the edge midpoints. Then, it moves the points within each triangle so the path goes through the centroids. These local changes must be homotopies, because they exist within quadrilaterals defined by pairs of adjacent triangles.
- Unfortunately, the converse of this lemma is false! You can have two homotopic paths with different crossing sequences. Imagine a contractible loop that goes back and forth over a single triangulation edge over and over before returning to its basepoint.

Reduction

- What we're going to do is *reduce* crossing sequences by repeatedly removing adjacent pairs of identical symbols.

- Intuitively, you can imagine each pair of symbols removed corresponds to removing a *bigon* formed between a triangulation edge and a contiguous part of the path.
- Let x dot y denote the concatenation of strings x and y, and let eps be the empty string. An *elementary reduction* is a transformation of the form x dot aa dot x → x dot y. An *elementary expansion* is the opposite x dot y ← x dot aa dot y. An *elementary transformation* is either of those.
- Two strings are *equivalent* if one can be transformed into the other by a finite sequence of elementary transformations. This is an equivalence relation. A string is *trivial* if it is equivalent to the empty string and *reduced* if no symbol appears twice in a row.
- Lemma: Every string is equivalent to exactly one reduced string.
    - Let w be a string and w → x and w → y be elementary reductions. We'll start by arguing either x = y or there is a string z such that x → z and y → z.
        - If w = w_1 dot aa dot w_2 dot bb dot w_3 and we deleted aa for x and bb for y, then delete the other pair from either to get to z.
        - Otherwise, we must have deleted two overlapping sets of the same symbol. Either the exact same pair so x = y or w = w_1 dot aaa dot w_2, again implying x = y = w_1 dot a dot w_2.
    - Now suppose we have two distinct but equivalent reduced strings x and y. There's a sequence of elementary transformations x = w_1 <-> w_2 <-> … <-> y.
    - Suppose there's some w_{i-1} ← w_i → w_{i+1}. If w_{i-1} = w_{i+1}, we can remove w_{i-1} and w_i to get a shorter sequence. Otherwise, there is a z_i such that w_{i-1} → z_i ← w_{i+1}.
    - Inductively, we can repeatedly do these changes to put all reductions before all expansions. Let z be the shortest string after all is done. Now either z is shorter than at least one of x or y, meaning it was not reduced after all.
- So if we want to know if two strings are equivalent or if a string is trivial, we just need a way to check if their reductions are equal or empty.
- Lemma: Any string of length x can be reduced in O(x) time.
    - We could just go looking for pairs of strings over and over but that's kind of slow. Instead, we'll use a stack that holds a fully reduced prefix of the string and scan from left to right.
    - This procedure uses a special fencepost character to tell us where the end of the currently processed string lies.

```
LeftGreedyReduce(X[1..x]):
  x̄ ← 0
  X̄[0] ← •                    ⟨⟨fencepost⟩⟩
  for i ← 1 to x
      if X[i] = X̄[x̄]
          x̄ ← x̄ − 1          ⟨⟨pop⟩⟩
      else
          x̄ ← x̄ + 1
          X̄[x̄] ← X[i]        ⟨⟨push⟩⟩
  return X̄[1..x̄]
```

The linear-time left-greedy reduction algorithm.

Back to Homotopy

- Now we establish why we want these reduced crossing sequences.
- Lemma: Any two paths with the same endpoints and the same reduced crossing sequences are homotopic.
  - Let alpha and beta be two paths whose crossing sequences differ by exactly one pair e_i e_i. We may assume alpha and beta are canonical paths.
  - But then one of them has a little subpath from an edge midpoint to a triangle centroid back to the same midpoint.
  - Contract that little spur to get to the other path.
  - The lemma then follows by induction on the number of elementary reductions for both to reach the common reduced string. After these reductions/spur removals you have two copies of the same canonical path.
- Lemma: Any two homotopic paths have the same reduced crossing sequence.
  - It suffices to prove that any contractible loop ell has an empty reduced crossing sequence.
  - However, a full proof of this is surprisingly subtle. Intuitively, though, you can think of the contraction as sucking up little bits of ell across a single edge e_i. This corresponds to removing a pair of e_i from the crossing sequence. Eventually it reduces to nothing.
- But now we've seen enough to get the full algorithm. Compute the crossing sequence(s). Do the reduction(s). Test if the reduced string(s) are equal or empty.
- Any path of length k has at most nk edge crossings so reductions take O(nk) time. The triangulation takes O(n log n) time.
- Theorem: Let P be a polygon with holes with n vertices, and let alpha and beta be polygonal paths in P with k edges total. We can determine whether alpha and beta are homotopic in P in O(n log n + nk) time.

Variantions

- We'll briefly discuss some variations on the problem and algorithm.

Cycles

- Nearly the same algorithm can test if two polygonal cycles are freely homotopic.
- The big change is that the crossing sequence is better thought of as a *cyclic string*. Cyclic strings form an equivalence class where for any two strings w and x, cyclic strings w dot x and x dot w are equivalent. One is *cyclicly reduced* if no character appears twice in a row; in particular, the "first" and "last" characters must be different.
- We can still reduce a cycle string of length x and test if two reduced strings are identical in O(x) time.
- Theorem: Let P be a polygon with holes with n vertices, and let alpha and beta be polygonal cycles in P with k edges total. We can determine whether alpha and beta are freely homotopic in P in O(n log n + nk) time.

Few Holes

- Now suppose the number of holes h is much smaller than n.
- We replace each hole in P with a single *sentinel point* as follows:
- Let S = {s_1, s_2, …, s_h} denote a set of h points where each point s_i lies in the corresponding hole P_i.
- Lemma: A loop in P is contractible in P if and only if it is contractible in R^2 \ S.
- So now we take a large rectangle R that surrounds the whole polygon P and triangulate R \ S in O(h log h) time. A path of length k has at most O(hk) crossings in this setting.
- Theorem: Let P be a polygon with h holes with n vertices, and let alpha and beta be polygonal paths (or cycles) in P with k edges total. We can determine whether alpha and beta are (freely) homotopic in P in O(n + h log h + hk) time.