Shortest Path in a Polygon

Alex Grinman *agrinman@mit.edu*

1 Problem and Definitons

Problem: Given two points S, T in a polygon, find the shortest path from $S \to T$ *inside* in the polygon. Intuitively we can see the solution to these problems as a tight string connecting two pins S and T sticking into the 2D polygon.

1.1 Simple Polygon Graphs

Definition 1.1. A *simple polygon* is flat 2D shape consisting of straight, non-intersecting segments ("sides") that are join pair-wise to form a closed path.

Definition 1.2. $\pi(s,t)$ is the euclidian shortest path from $s \to t$ inside a simple polygon P where s and t are both points inside P. $|\pi(s,t)|$ is the geodesic distance between s and t.

Lemma 1.3. *$\pi(s,t)$ is unique in a simple polygon P.*

Proof. Suppose, this is not true. Then there exists at least two different shortest paths $\pi_1(s,t)$ and $\pi_2(s,t)$. Since these paths are different there must be two disjoint subpaths $s_1 \in \pi_1$ and $s_2 \in \pi_2$, that start and end at the same nodes. Therefore s_1 and s_2 together form a convex shape inside the interior of P. We can find a shorter path by collapsing this convex part of the path. This contradictions that π_1, π_2 were the shortest paths.

2 Funnel Algorithm on Simple Polygons

Let P be a simply polygon with n vertices. Let S be the source vertex and T be the target vertex.

2.1 Algorithm Overview

1. compute the triangulation of the polygon – $O(n \log n)$
2. compute the dual graph of the triangulation – $O(n)$
3. use DFS on the dual graph starting at the triangle containing S to the triangle contain T. – $O(n)$
4. add the diagonals (chords of P) found (crossed) in the DFS from S to T, and for each diagonal calculate the funnel that represents the shortest path from S to each of the endpoints. – $O(n)$

2.2 Triangulation

Definition 2.1. The triangulation of a simply polygon P is defined as the decomposition P into a set of triangles such that the union of these triangles is P.

Theorem 2.2. *Every simple polygon of n vertices has triangle decomposition of exactly $n-2$ triangles.*

Proof. Left as an exercise (hint: simple induction by adding chords).

2.2.1 Trianglulation Algorithm

Very rough idea: "line sweep" approach. Sweep a horizontal line from top to bottom and stop every time we hit a vertex. Depending on the type of vertex and current state, add/remove into a BST. Algorithm runs in $O(n \log n)$. (Siedel Triangulation Algorithm - not covered).

2.3 Dual Triangle Graph and its Shortest path via DFS

The decomposition of P into triangles creates a dual graph, namely, a graph of triangles, where each triangle can be thought of as a node. Two triangle nodes are connected if they share an edge. We can see that the shortest path of triangle nodes from $S \to T$ (the triangles that respectively contain S and T) can be found by representing each the triangle decomposition as tree. Therefore we can solve the shortest path (of triangle nodes) by simply doing DFS on the dual tree.

2.4 Funnel

Definition 2.3. A funnel is the concept used to find the next vertex in the shortest path. Suppose we have a point s in the polygon P and and a diagonal (chord) $d = \overline{ab}$ of P. A funnel is defined as the region between $\pi(s,a)$ and $\pi(s,b)$ and $\overline{ab}$. The following is true about a funnel:

- $\pi(s,a)$ and $\pi(s,b)$ share the same initial chain $\pi(s,c)$ where c is called the *apex*, the point where $\pi(s,a)$ and $\pi(s,b)$ diverge.

This defines the substructure of our algorithm explained next.

2.5 Funnel Algorithm (iteration over DFS diagonals)

Idea: Once we have the shortest path of triangle nodes from the previous step, we need to actually convert that path into a path of points inside the polygon. We can observe that two consecutive triangles in the triangle node shortest path share an edge $d = \overline{ab}$, which is a diagonal (chord) in the polygon. These are exactly the diagonals we will iterate over creating a funnel each time with our current shortest path apex.

This algorithm computes the funnel from S to every diagonal in the triangle DFS path from S to T. It iteratively advances, increasing the funnel, keeping track of the current funnel and apex in each step. The idea is keep two $reflex$ chains from the apex to the endpoints of the current diagonal, where a reflex chains are the shortest paths from the source point to the endpoints of the current diagonal. The goal is two maintain these two shortest paths until we get to the target diagonal (and the target point). We always add the shortest line segment to the shortest path (i.e if a straight line is possible through the interior, then we add that instead

General step for next diagonal $d_i = \overline{ac}$, given apex r, reflex chains $\pi(r,a)$ and $\pi(r,b)$:

1. add new triangle $\overline{abc}$
2. work from b back down the reflex chain to find a possibly find vertex tangent to b.

   if tangent found at vertex t, discard all nodes in $\pi(t,b)$, and add new segment $\pi(t,c)$

   if *not* found, append newsegment $\pi(r,c) = \pi(r,b) \cup \overline{bc}$
3. go to next diagonal

2.5.1 Complexity Analysis of a funnel step

Note that each node get's discarded at most 1 time, hence this algorithm runs in total $O(n)$. Hence, overall complexity $O(n \log n)$

References

[1] Geometry Lab. *Calculating the shortest path in a simple polygon - the Funnel Algorithm.* `http://web.informatik.uni-bonn.de/I/GeomLab/ShortestPathLP/`

[2] Stanford CS268. 1992. *Shortest Path Problems.* Jim Stewart. `https://graphics.stanford.edu/courses/cs268-09-winter/notes/handout7.pdf`

[3] University of Waterloo CS 860. Fall 2014. *Basic geometric shortest path algorithms – shortest paths in 2D.* Anna Lubiw. `https://cs.uwaterloo.ca/~alubiw/CS860/Lecture2-class.pdf`