1. Algorithms for flat world
   (a) Planar graph: graph embedded in plane, can assume all edges are straight lines
   (b) One outer face, $F$, directed, non-negative edge lengths $l(u \rightarrow v)$
   (c) Two structures: lengths, and combinatorial edge structure. Lengths are not related to embedding.
   (d) Assume all faces triangulated for simplicity
   (e) Goal: $O(n)$ distance queries $(u, v)$, $u$ on outer face, $v$ anywhere
   (f) Alternate view: implicit representations of all shortest path trees
   (g) $O(n \log n)$ algorithm to Klein, 2005. Generalized to embedded graphs by Cabello, Chambers, and Erickson 2012.
   (h) Will mostly follow Klein’s presentation.

2. Properties of shortest path trees
   (a) Let outermost face in clockwise order be $r_0 \ldots r_k$.
   (b) Ensure $r_0$ can reach all vertices by adding infinite length edges.
   (c) Add edges $r_1 \rightarrow r_0, r_2 \rightarrow r_1, \ldots$ with infinite length.
   (d) Each $r_i$ has shortest path tree $T_i$ rooted at it.
   (e) (due to Gary): perturbing edge lengths by $1/\text{poly} n$ ensures $T_i$ is unique.
   (f) Left face of an edge $u \rightarrow v$: edge to the left when $u \rightarrow v$ is oriented upwards
   (g) Dual graph: each face becomes vertex, edge maps to edge from left face to right face
   (h) Property of planar graphs: removal of a spanning tree $T_i$ gives spanning tree in dual graph $T^*_i$.

3. Algorithm
   (a) Build shortest-path tree rooted at $r_0$
   (b) Unrelaxed edge $u \rightarrow v$: $d(u) + l(u \rightarrow v) < d(v)$.
   (c) Reduced length: $\hat{l}(u \rightarrow v) = d(u) + l(u \rightarrow v) - d(v)$. Edge unrelaxed if reduced length is negative.
   (d) For $i = 1 \ldots k$:
      i. Remove parent edge entering $r_i$
      ii. Add $r_i \rightarrow r_{i-1}$ to tree
      iii. Keep on relaxing unrelaxed edges.
   (e) Need to find unrelaxed edges, and update $T, T^*$, and $\hat{l}$ quickly.
   (f) Maintain edges using dynamic trees.
   (g) Modification operation: replace $w \rightarrow v$ with $u \rightarrow v$. 

(h) Distances of vertices in \( u \)'s subtree decrease by \( \Delta = \hat{l}(u \rightarrow v) \).

(i) Need to know when an edge becomes unrelaxed in dual tree.

(j) Change lengths \( \hat{l} \): edges into subtree of \( v \) increase by \( \Delta \), while edges leaving decrease by \( \Delta \).

(k) Let faces to left/right of \( u \rightarrow v \) be \( x \) and \( y \).

(l) Paths from \( x \) to \( y \) along \( T^* \) precisely the list of edges leaving \( v \)'s subtree.

(m) Directions of edges given by direction of edges on paths.

(n) Bulk add to a subsection once path becomes a BST. \( O(\log n) \) per update.

4. Bound on number of relaxations

(a) Can show that for some order of relaxing unrelaxed edges, each edge is relaxed at most once.

(b) Show instead: the indices \( i \) where \( u \rightarrow v \in T_i \) is contiguous on the outer face (could wrap around cyclically).

(c) Need to prove: once shortest paths from \( s_i \) to \( u \) and \( s_j \) to \( u \) intersect, they will be same.

(d) Sketch: suppose they first intersect at \( z \), then one of the two subsequent paths from \( z \) to \( u \) must be shorter.

(e) The shortest paths from \( s_i \) to \( v \) move clockwise as \( s_i \) moves clockwise.

(f) \( u \rightarrow v \) is one of the possible last edges, it gets used for a contiguous section of the is.

(g) Leafmost unrelaxed edge: unrelaxed edge with no unrelaxed descendants in \( T_i^* \).

(h) Can show that this edge is not in \( T_{i+1} \).