1 Introduction

1.1 The problem

Given a planar graph $G = (V, E)$ and a set of source vertices $S$, we wish to support distance queries $d(s, v)$ where $s \in S$ and $v \in V$. In particular, let’s construct shortest-path trees rooted at each $s \in S$.

1.2 A naive algorithm

What if we just run an SSSP algorithm on each node of $S$? From a presentation 2 months ago, SSSP can be done in $O(n)$ time on planar graphs, so time $O(|S| \cdot n)$.

It turns out that if all vertices of $S$ are on the same face, then there exists an algorithm which takes $O(n \log n)$ time (no dependence on the size of $S$!)

2 Review of planar graphs

The dual of a planar graph $G$ is a graph $G^*$ whose vertices are the faces of $G$ (i.e. the regions bounded by cycles) and whose edges are the edges of $G$, where each edge connects the two faces that it separates.

Interdigitating trees lemma: If we have any spanning tree $T$ of $G$, then the unused edges make a spanning tree in $G^*$.

3 The algorithm

The overall idea: Compute one shortest-path tree and modify it to get the shortest-path tree rooted at an adjacent vertex on the face by continuously moving the root between them.

3.1 Tension

Given a spanning tree $T$ rooted at $r$, define the tension of an edge $uv$ to be $t(uv) = d(r, v) - l(uv) - d(r, u)$, so if all edges have tension 0, then $T$ must be a shortest-path tree.
If we start with a shortest-path tree rooted at \( r_i \) and want to move to a shortest-path tree rooted at \( r_{i+1} \), we first color the nodes so that the children of \( r_{i+1} \) in the \( r_i \) shortest-path tree are blue and the remaining nodes are red. Then we want to repeatedly find the largest tension of any edge from a blue node to a red node, add that edge to the tree, and update tensions (see example).

### 3.2 Dynamic trees

To do this efficiently, we keep separate dynamic trees to represent the original, or primal, graph, and in the dual graph. In the primal graph, each node also stores its distance from the root, and in the dual graph, each edge stores its current tension. By the interdigitating trees lemma, finding the maximum tension edge is equivalent to finding the largest edge value on some path in the dual graph, which dynamic trees can do in \( O(\log n) \) time.

### 4 Time analysis

We claim that each edge is modified at most twice. Lemma: The set of all roots whose shortest-path trees contain a given edge are in an interval along the face. Proof by contradiction.

Combined with \( O(\log n) \) time per dynamic tree operation, gives \( O(m \log n) = O(n \log n) \). We also have \( O(|S| \log n) \) work with the MAXPATH calls, which is also \( O(n \log n) \).

Can use techniques from persistent data structures to get an \( O(n \log n) \) space data structure that supports all distance queries in \( O(\log n) \) time.