

## 1 Elementary-cycle sizes

This is a repeat of a problem from the last homework:

Give a linear-time algorithm for the following:

- input: a planar embedded graph  $G$ , and a spanning tree  $T$  of  $G$
- output: a table giving, for each nontree edge  $uv$ , the length of the simple  $u$ -to- $v$  path in  $T$  between  $u$  and  $v$ .

## 2 Vertex separators in trees

A *vertex separator* (or *edge separator*) for a graph is a set  $S$  of vertices (or edges) whose deletion breaks the graph into pieces, each of which is “small” in comparison to the original graph.

Your goal will be to prove the following lemma.

**Lemma 2.1** *Let  $T$  be a rooted tree. Let  $\hat{w}(\cdot)$  be an assignment of non-negative weights to vertices such that the weight of each vertex is at least the sum of the weights of its children. Let  $W$  be the weight of the root. Assume  $W > 0$ . Let  $\alpha$  be a positive number less than 1. Then there is a linear-time algorithm to find a vertex  $v_0$  such that  $\hat{w}(v_0) > \alpha W$  and every child  $v$  of  $v_0$  satisfies  $\hat{w}(v) \leq \alpha W$ .*

You will use a procedure of the following form:

```
define  $f(v)$ :  
1   if (...something...),  
2       return  $f(u)$   
3   else return  $v$ 
```

Then prove the following corollary.

**Corollary 2.2 (Tree Vertex Separator)** *Let  $T$  be a tree, and let  $w(\cdot)$  be an assignment of nonnegative weights to vertices. Let  $W$  be the sum of the weights. There is a linear-time algorithm to find a vertex  $\hat{v}$  such that every component in  $T - \{\hat{v}\}$  has total weight at most  $W/2$ .*

## 3 Edge separators in degree-three trees

Use Lemma 2.1 to prove the following lemma.

**Lemma 3.1 (Tree Edge Separator)** *Let  $T$  be an unrooted tree having degree at most three, and let  $w(\cdot)$  be an assignment of weights to edges. Let  $W$  be the sum of weights. There is a linear-time algorithm to find an edge  $\hat{e}$  such that every component in  $T - \{\hat{e}\}$  has total weight at most  $2W/3$ .*

## 4 Elementary-cycle separators in planar graphs

Prove the following lemma.

**Lemma 4.1 (Elementary-cycle separator)** *Let  $G$  be a planar embedded graph, and let  $f_\infty$  be a face of  $G$ . Suppose every face of  $G$  consists of three edges. Let  $T$  be a spanning tree of  $G$ . Let  $w(\cdot)$  be an assignment of weights to faces such that each face has weight at most  $W/4$ , where  $W$  is the sum of the weights. There is a linear-time algorithm to find a nontree edge  $\hat{e}$  such that*

- the set of faces enclosed by  $C_{\hat{e}}$  has weight at most  $3W/4$ , and
- the set of faces not enclosed by  $C_{\hat{e}}$  has weight at most  $3W/4$ ,

where  $C_{\hat{e}}$  denotes the elementary cycle of  $\hat{e}$  with respect to  $T$ .

## 5 EXPOSE( $v$ ) for dynamic trees that support descendent searches

In lecture, I gave pseudocode for EXPOSE( $v$ ) and analyzed it for dynamic trees without dashed nodes. Give pseudocode for the case of dynamic trees with dashed nodes. Describe briefly how the analysis should be adapted to this case.

## 6 Rudimentary dynamic tree

In this problem, we discuss a data structure that is considerably simpler than dynamic trees but that nevertheless supports some interesting tree operations.

Let  $T$  be an embedded planar graph that is a tree. Note that  $T$  has a single face, which is a permutation cycle  $(d_0 d_1 \dots d_{r-1})$  on the set of darts of  $T$ . (Recall that a permutation cycle is written starting at an arbitrary element of the cycle.) We represent this cycle by a BST whose nodes are the darts, and the BST order is  $d_0 d_1 \dots d_{r-1}$ .

Using this representation, we can represent a subgraph of a graph, as long as the subgraph is a forest; there is one BST per tree in the forest.

Show how to support the following operations each in  $O(\log n)$  time, where  $n$  is the total number of edges.

- REMOVE( $e$ ): remove an edge  $e$  from the forest.
- ADD( $e$ ): add an edge to the forest

Note: we have identified nodes of the BSTs with darts, so

## 7 Ancestor queries

Show how to support the following operation in the data structure of Problem 6:

- $\text{ANCESTOR}(e_1, e_2, e_3)$ : given three edges, return *true* if they all appear in the same tree, and  $e_2$  lies on the simple  $e_1$ -to- $e_3$  path.

## 8 Edge-separator queries

Show how to support the following operation in the data structure of Problem 6:

- $\text{SEPARATOR}(e)$ : given an edge  $e$  in the forest, return an edge  $\hat{e}$  in the tree  $T$  containing  $e$  such that each connected component of  $T - \hat{e}$  has at most  $2|E(T)|/3$  edges.