1. (a) Proceed greedily by recursion. When we are at a tree rooted at \( r \) with children \( c_1, c_2, \ldots, c_k \), gives the solid edge to the the \( c_1 \) (wlog suppose they are sorted in terms of size).

Let's prove the \( \log n \) bound by induction on the tree structure. The bound is obvious for a leaf node, as \( \log 1 = 0 \).

Now suppose as before we have a root \( r \) and children \( c_1, c_2, \ldots, c_k \) sorted by size (so, \( |c_1| \geq |c_2| \geq \ldots \geq |c_k| \)). Note that \( |c_1| \leq n \), so the access cost for any element in \( c_1 \) is at most \( \log n \) (we gave \( c_1 \) as solid edge). For all the other subtrees, note that \( |c_i| \leq \frac{n}{2} \), so the access cost in them is at most \( \log n - 1 \). Therefore in \( r \) the cost is at most \( \log n \) as desired.

(b) We use a potential function argument. Let \( \Phi = \) the number of nodes in the tree where \( G \) and \( B \) color different edges to its children solid.

We know that \( AC_{G,\sigma_i} = C_{G,\sigma_i} + \Delta \Phi \), and we want to show that \( AC_{G,\sigma_i} \leq 2C_{B,\sigma_i} \).

Let's look at the nodes and edges along the path from \( x \) to the root. Let \( S \) be the set of these nodes/edges which are dashed in the running of \( G \) and also dashed in \( B \), and \( T \) those which are dashed in \( G \) but not in \( B \). Then \( C_{G,\sigma_i} = |S| + |T| \). The cost of \( B \) is at least \( C_{B,\sigma_i} \geq |S| \).

Next, we claim that \( \Delta \Phi \leq |S| - |T| \). Before flipping edges, note that the edges in \( T \) were exactly the ones counted by the potential function. Now, the edges in \( T \) are not counted but the ones in \( S \) might be, which gives the desired result. So,

\[
AC_{G,\sigma_i} = C_{G,\sigma_i} + \Delta \Phi = |S| + |T| + |S| - |T| = 2|S| \leq 2C_{B,\sigma_i}
\]

as desired.

Now let's reason about the relabeling that \( B \) does on \( \sigma_i \). Notice that each relabel \( B \) does can only make \( \Delta \Phi \leq 1 \). So \( AC_{G,\sigma_i} \leq C_{B,\sigma_i} \leq 2C_{B,\sigma_i} \) as desired.

Finally, note that \( \Phi_0 = 0 \) as \( T = T \), and \( \Phi \) can never be negative. So, by telescoping we are done.

(c) Note that the static access algorithm incurs cost at most \( n + m \log_2 n \): we change at most one edge per node to set the solid and dashed edges into the proper structure, and then every access costs at most \( \log_2 n \). Since we proved \( G \) was 2–competitive, it must have cost at most 2 times any other algorithm. In particular, it must have cost at most \( 2n + 2m \log_2 n \) as desired.

2. Construct a suffix tree for \( S \) and \( S^R \), the reverse of the string. This is our \( O(n) \) preprocessing.

We can augment each suffix tree with the longest path to a leaf at every node. Note that since each leaf represents an end of a suffix, this augmentation tells us for each edge the first occurrence of the substring ending at that edge.
4. We show that we can do each of the $O(\log n)$ rounds of the surrogate algorithm with $O(n/\log n)$ processors and time $O(\log^2 n)$. Doing over all rounds and then doing a map with $O(n/\log n)$ processors and time $O(\log n)$ gives the desired bounds.

As shown in the previous problem, we can sort the array in time $O(\log^2 n)$. The only step left is assigning values in $\{0,1,\ldots,n-1\}$ to the first two surrogates (or, in the case of the
first iteration, the first two characters). We use the prescan algorithm from lecture, by first mapping to each element in the list a 1 if it is not equal to the element to the left of it, and 0 otherwise. Then the map and the scan both satisfy the bounds ($O(n/\log n)$ processors and $O(\log n)$ time).