Editorial: Towns

The graph given in this task is an edge-weighted tree T), in which the degree of each internal node is at least three. Let n denote the number of leaves of T). The goal is to find the centers of the tree and determine if any center is also a leaf median (an internal node such that after removing the node, every component contains at most $n/2$ leaves. In this task, the tree is not given, and the contestants must solve the task by using limited number of queries for distances between leaves. An algorithm using $7n/2$ queries is sketched as follows. The details are in the next sections. * First, find the centers by at most $2n-3$ queries; and then * for each center (at most two centers), determine if it is also a leaf median by using no more than $3n/2$ queries.

Radius and centers

The diameter of a tree can be found with $2n - 3$ queries as follows. * Farthest to-Farthest: Pick an arbitrary vertex v and find a vertex s farthest to r. Find a vertex t farthest to s. It can be shown that $d(s,t)$ is a diameter of the tree. * Any center must be one the vs-path. * Then the vertices on the vs-path with its distance to s closest to $d(s,t)/2$ are centers.

Determining if a center is also a median

Let v,s be the two leaves in the above process and m be a center on the vs-path with $d(s,m) = r$. We need to determine if each component of $T - m$ has at most $n/2$. Let S be the set of all leaves.

First we compute the multiset $B = \{(d(u,s) + d(s,v) - d(u,v))/2 \mid \forall u \in S\}$. Each of the different values in B identifies a unique internal vertex on the svpath. Let m^0 be the internal vertex on the sv-path with $d(s,m^0) = \alpha$, where α is the median of the multiset B. If we root T at the sv-path, the leaf median must be in the subtree rooted at m^0. Therefore if r 6= α (i.e., m 6= m^0), then m is not a median. Note that there are two medians in B if |B| is even. Otherwise it remains to solve the "giant component" problem: checking if there is a component in T-m with more than n/2 leaves.

Let $X = \{u \in S \mid d(u,s) + d(s,v) - d(u,v) = 2r\}$ which is the set of the leaves branching from sv-path at m. For $x_1,x_2 \in X$, we have that x_1,x_2 are in the same component of T-m iff $d(s,x_1) + d(s,x_2) - d(x_1,x_2) > 2r$. Then, we can solve the giant component problem by algorithms for the following problem: There are n color balls and we want to determine if there are more than n/2 balls of the same color.

The cost is counted by the number of performed queries R(u,v) which returns Equal/NotEqual according to if u and v are of the same color.

It was shown that d3n/2e − 2 queries are necessary and sufficient in Fischer and Salzberg (1982), Solution to problem 81–5, J. Algorithms, pp. 376–379. The following is another similar approach. Initially each element is itself a set. At each iteration, we arbitrarily pair up the survival sets and compare their representatives. If they are equal, then join the two sets into their union. Otherwise, mark both dead. In the case that the number of alive sets is odd, mark anyone dead. Repeating this process, eventually either there is exactly one survival set or all sets are dead. It can be shown that if there is a majority originally, then it must be the survival one. So the remaining work is to compare the survival representative with the representatives of all DEAD sets.

The number of comparisons: Let A_i denote the number of alive sets in the i-th iteration ($A_0 = n$). The number of comparisons to obtain the only survival set is $1/2(A_0 + A_1 + A_2 ...)$. In the second stage the number of comparisons is the number of dead sets. Let D_i denote the number of sets die at iteration i. Then, we have $D_1 = A_0 - 2A_1$. Similarly $D_i = A_{i-1} - 2A_i$. Therefore, the total number of dead sets is $(A_0 - 2A_1)+(A_1 - 2A_2)+... = A_0 - (A_1 + A_2 + ...)$. In summary, the number of comparisons is $(3/2)A_0 - (1/2)(A_1 + A_2 + ...) < 3n/2$.