I collaborated with: W3023339

Problem
Until recently, there was no known method for computing the diameter of a graph that didn't first compute the shortest path between all pairs of nodes. When graphs are dense, all-pairs shortest paths is fairly expensive, so some people have explored quicker algorithms which *estimate* the diameter of the graph. Develop a linear-time algorithm that, given a graph G, returns a diameter estimate that is always within a factor of 1/2 of the true diameter. That is, if the true diameter is $diam(G)$ then you should return a value k where $diam(G)/2 \leq k \leq diam(G)$.
Solution

Algorithm 1 Diameter approximation of tree T using vertex r

```
function DIAMETER(G,r)
    Mark all v ∈ V as unvisited
    Let T be an empty graph
    Add r to T; mark r as visited; r.level ← 0
    k ← 0
    while There are visited vertices do
        current ← some visited vertex having minimum level
        k ← current.level if current.level > maxLevel
        Mark current as explored
        for unvisited neighbors v of current do
            Add {current, v} to T
            Mark v as visited
            v.level ← current.level + 1
        end for
    end while
return k
end function
```

Claim: The level k shows the distance from a vertex r to the vertex farthest away from it t is $diam(G)/2 \leq k \leq diam(G)$.
All edges of G not in T connect vertices at consecutive levels (or at the same level) of T, so k represents the number of hops from r to the node farthest away form it. For the lower bound of k, If there is more than one node in T at level k then the diameter of our graph is $2k$ or $diam(G)/2 = k$. Now for the upper bound, if we got lucky and picked an r such that the shortest path from it to another vertex t happend to be the longest path in G then $k = diam(G)$.Thus, $diam(G)/2 \leq k \leq diam(G)$.