Kick Start 2018 - Round C

Analysis: Planet Distance

Planet Distance: Analysis

The problem statement explains that we have an undirected connected graph with N nodes and N edges, and exactly one cycle in it. Our task here is to first find the nodes that are part of the cycle, and then find the minimum distance from each of the other nodes to this cycle. From the input, we can form an adjacency list and use this to solve the problem.

Small dataset

We can perform a DFS from every node, keeping track of visited and parent nodes, to get the nodes that are a part of the cycle. During the DFS for node i, if the first visited node we encounter (that is not the parent), is the node i, then node i is part of the cycle.
Time complexity for this method is $O(N^2)$.

Once we have identified all the nodes in the cycle, we can do a BFS from each of these cyclic nodes, keeping track of the number of edges we have travelled so far, to get the minimum distance to all the other nodes. We initially mark all the cyclic nodes as visited, to ensure that we dont traverse the cycle during the BFS.

The overall time complexity is $O(N^2)$.

Large dataset

We can identify which nodes are part of the cycle in many ways. One way is to do a DFS, while keeping track of the parent node. If you encounter a node that is visited, and is not the parent of the previous node, it means that this node is part of the cycle. You can then backtrack using the parent nodes to get all the nodes of the cycle. The time complexity is O(N).

The second way is to recursively remove all vertices of degree 1. This can be done efficiently by storing a map of vertices to their degrees.

Initially, traverse the map and store all the vertices with degree = 1 in a queue. Traverse the queue as long as it is not empty. For each node in the queue, mark it as visited, and iterate through all the nodes that are connected to it (using the adjacency list), and decrement the degree of each of those nodes by one in the map. Add all nodes whose degree becomes equal to one to the queue. At the end of this algorithm, all the nodes that are unvisited are part of the cycle.
The time complexity for this method is O(N).

Once we have identified all the nodes in the cycle, we can do a BFS from each of these cyclic nodes, keeping track of the number of edges we have travelled so far, to get the minimum distance to all the other nodes. We initially mark all the cyclic nodes as visited, to ensure that we dont traverse the cycle during the BFS.

The overall time complexity is O(N).