Review of Highway Tolls

Problem

We are given an undirected and unweighted graph G with N vertices and M edges, and constants $1 \leq A < B$.

Two vertices s and t are fixed but they are unknown to us.

We want to find s and t by calling the following function fewer times:

- For each edge in G, you arbitrarily assign the weight of A or B to turn G into a weighted graph. Then, the function returns the length of a shortest path between s and t on (weighted) G.

Subtask and Solutions

- Overall constraints: $N \leq 90,000$, $M \leq 130,000$

Subtask 1 (5 points)

at most 100 function calls, G is a tree, $N \leq 100$, s is known

- Test every possible t.

Subtask 2 (7 points)

at most 60 function calls, G is a tree, s is known

- Sort the vertices by the distance from s. Then t can be found using binary search

Subtask 3 (6 points)

at most 60 function calls, G is a path

- Binary search

Subtask 4 (33 points)

at most 60 function calls, G is a tree

- One function call with weight of every edge A to find the distance between s and t in unweighted G.
- An edge e on the shortest path between s and t can be found using binary search.
- After removing e, the graph will be separated into two subtrees. Then you can perform the solution of Subtask 2 twice to find s and t separately.
- Centroid decomposition is possible but implementation will be tough.

Subtask 5 (18 points)

at most 52 function calls, $A = 1, B = 2$

- Let S be a subset of V, where V is the set of vertices in G.
- We set the weight of edges between S and $V \setminus S$ to 1. The weights of other edges are set to 2. Then, we can tell whether exactly one of s and t belongs to S by looking at the parity of the answer to the call.
- Thus we can compute s xor t. Using this, we can also find s and t themselves.

Subtask 6 (21 + 10 points)

at most 52 or 50 function calls (21 or 31 points, respectively)

- Solution A: 21 points
  - A vertex v on a shortest path between s and t can be found using binary search.
  - Construct a BFS tree with root v. Then, we can use binary search again to find one of s and t.
  - The other can be found similarly.
- Solution B: 31 points
  - Find an edge e on a shortest path between s and t as in Subtask 4.
  - Let $e = uv$. Without loss of generality, we can assume s, u, v and t appears in this order on this shortest path.
  - Then we can prove that s is strictly closer to u than to v. Similarly, t is closer to v than to u.
  - Thus we have disjoint candidate sets S and T such that s and t are contained in S and T, respectively. At the same time, we can construct BFS trees of vertex sets S and T with roots u and v, respectively. We can suppose a shortest path goes only through e and edges in the BFS trees.
  - Now we can find s and t as in the last part of Subtask 4.