Solution Description - Tourists

Subtask 1 (n,m,q ≤ 200):

We can use naive solution to solve this subtask: meaning we simulate all queries for all tourists separately. For each tourist we keep his current opinion and current city. When we process the question query we simply read the value from the appropriate cell. When processing event query we add the event value to all tourists who are currently in a given city. When processing travel queries, we run the DFS/BFS to calculate the distance between the cities and update all the values appropriately.

*This solutions runs in O(n*m*q)*

Subtask 2 (n,m,q ≤ 2000):

For this subtask we preprocess distances between each pair of the cities using DFS/BFS instead of calculating it separately for each query.

*This solutions runs in O(n^2+m*q)*

Subtask 3 (m,q ≤ 2000):

Here we use LCA to calculate the distance between each pair of cities for each tourist in each query.

*This solutions runs in O(n*log n +m*q*log n)*

*n*log n for LCA preprocessing*

Subtask 4 (no ‘e’ queries):

We use segment tree to keep the current city of each tourist and their opinion. When keeping the current city in any node, we have the current city of all childs for a given node or -1 if there are many towns tourists are in a given segment. When processing travel queries we will go to all the nodes that we will go to in a normal segment tree query, bu if any of those nodes will contain -1 (for the current city), we will go to their children, until they will contain values different than -1 and then perform travel operations on those nodes: meaning we change the current city and tourist opinion in a give node. Although in any single query we may go to *m log m* nodes in total we will update no more than *~ 4 * m log m* nodes. This is because the number of updated nodes is dependent on the number of tourist segments (we define it as a range <i,j> where all tourists between number i and j are in the same city). After each query all the tourist segments in the travel query range are merged into a single segment. When processing a question query, we will sum up values from the given tourist node and all his parent nodes. Assuming we use LCA to calculate the distance and we cache the last call to calculate the distance:

*solution runs in O(n*log n + (m+q)*log m)*

Subtask 5 (no further restrictions):

We will keep the array containing the sum of all the invent values for each city from the beginning. When we update a tourist node traveling from city A to city B, we will add value corresponding to city A and deduce the value corresponding to city B. When answering to the question queries we simply add the value corresponding to the current city of a given tourist and add it to the value from segment tree (as described in the last subtask)

*solution runs in O(n*log n + (m+q)*log m)*