**Problem Statement 2 – Trading Model**

Algorithm

Since the given graph has a tree structure, there is one only one way to go from any one node to another without repeating any node. For each pair ui and vi in each query, find the path joining both these vertices. This can be done with depth first search. All the nodes whose path from ui doesn’t include any edge of the path joining ui and vi would not be reachable from vi since vi will have to cross the initial position of ui to reach there. Hence, more such nodes with ui implies more will be the strength of stocks with ui. Thus, ui will try to minimize the edges whose path consists of some edge that is present in the path joining ui and vi. Similarly, vi try to maximize its strength by doing the same. As a result, ui and vi will go till there exists a single edge joining both. After this, stock strength of ui and vi will be sum of strength of all stocks in graph (tree structure) with root ui and vi respectively.

Choice of Data Structures

An array of vectors is used to make the graph. Vector of index i contains all indices that have an edge connected to i. Choosing array of vectors instead of array of arrays increases both average space and time performance. Strength of each stock is taken as input and stored in an array. An array of type bool (possible values true/false) is used to check if a stock is visited or not, while finding paths.

Time and Space Complexity

For every query, the most time expensive task is to calculate sum of strength of every stock for both people which includes considering every stock once. Thus, time complexity is O(n) for each query and thus, overall time complexity is O(n\*q).

A local vector “temp” is declared in the recursive function (“dfs”) which finds the sum of strengths of all the stocks in a graph (tree structure). Maximum number of variables created in a recursive call may be equal to n+(n-1)+(n-2) … + 1 which gives space complexity as O(n\*n).