Friends

6 solutions to 6 subtasks

	Solution1	Solution2	Solution3	Solution4	Solution5	Solution6
Subtask1	AC	WA	WA	WA	WA	AC
Subtask2	TLE	AC	WA	WA	WA	AC
Subtask3	TLE	WA	AC	WA	WA	AC
Subtask4	TLE	WA	WA	AC	WA	AC
Subtask5	TLE	WA	WA	WA	AC	AC
Subtask6	TLE	WA	WA	WA	WA	AC

Table 1-1: The result of each solution applying to each subtask.
Note: AC=Accepted, WA=Wrong Answer, TLE=Time Limit Exceeded.

Solution1

In subtask1, N is at most 10. So just apply backtracking for every person – to chose or not to chose. The complexity is $O(N * 2^N)$, Accepted.
The sizes of N in other subtasks are too large to apply this solution, resulting in Time Limit Exceeded with this complexity.

Solution2

In subtask2, there're all 'MyFriendsAreYourFriends' relations, forming a graph with no edge. That is equivalent to chose all persons, with complexity of $O(N)$.
For other subtasks, there're not only this kind of relations, so this solution does not work and will result in Wrong Answer.

Solution3

In subtask3, there're all 'WeAreYourFriends' relations, forming a complete graph. Since every pair of two persons is connected by an edge, the answer to this problem is equivalent to choose the maximum confidence among all people, with complexity of $O(N)$.
For other subtasks, there're not only this kind of relations, so this solution does not work and will result in Wrong Answer.

Solution4

In subtask4, all relations are 'IamYourFriend', forming a tree. So we apply the DP-in-tree method.
Define $dp[i][j]$ as the maximum sum for the i^{th} node with status j, where $j = 0$ stands for not choosing this node and $j = 1$ stands for choosing this node. Then:

(1) If the i^{th} node is leaf, then

- $dp[i][j] = 0$, for $j = 0$.
- $dp[i][j] = confidence[i]$, for $j = 1$.

(2) Otherwise,

- $dp[i][j] = max(dp[k][0],\ dp[k][1])$, for $j = 0$ and for all k, where k is i's child.
- $dp[i][j] = dp[k][0]$, for $j = 1$ and for all k, where k is i's child.

The final answer is $max(dp[root][0], dp[root][1])$, where root stands for the root of this tree.
For other subtasks, there're not only 'IamYourFriend' relations, so this solution will not work and will result in Wrong Answer.

Solution5

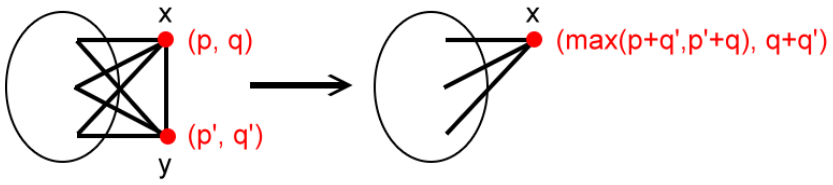
Since there're only 'MyFriendsAreYourFriends' and 'IamYourFriend' relations, the resulting graph contains no odd cycle. That is, we obtain a bipartite graph. With all confidence equals to 1, the problem becomes finding maximum independent set in a bipartite graph. As we know, a set is independent if and only if its complement is a vertex cover. If the complement of independent set is not a vertex cover, then there exists at least one edge with end points u and v, which is included in the independent set, conflicting with the definition of independent set. Trivially, a maximum independent set is the complement of minimum vertex cover.
According to Konig's theorem[1], in any bipartite graph, the number of edges in a maximum matching is equal to the number of vertices in a minimum vertex cover. Thus, we can apply the augmenting path algorithm to find out maximum cardinality matching in a bipartite graph, with complexity of $O(NE)$ or $O(\sqrt{N}E)$, depending on different implementations, where E is the number of edges.
Let k be the result of maximum cardinality bipartite matching, the answer to this problem equals to N - k, since maximum independent set is the complement

of minimum vertex cover.
As for partitioning the graph into bipartite, we apply *dfs* to mark out the odd points and the even points, and then put all odd points one side, even points the other side.
In subtask 2 and 4, the relations are 'MyFriendsAreYourFriends' and 'IamYourFriend'. However, the confidence value in those two subtasks are not all equals to 1. As a result, this solution can only solve subtask5 correctly, and Wrong Answer for other subtasks.

Solution6

By using Greedy method to eliminate each person in the reverse order of building process, we will finally get the (p, q) pair for the last person. The answer will be $max(p, q)$ of the last person.
Here we briefly introduce this method. Initially, we maintain two values $p(x)$ and $q(x)$ for each person x, where $p(x) = confidence[x]$ and $q(x) = 0$. Physically, p stands for 'choose' and q stands for 'not choose'.
To simplify the notation, we call p for $p(x)$, q for $q(x)$, p' for $p(y)$, q' for $q(y)$.

(1) WeAreYourFriends



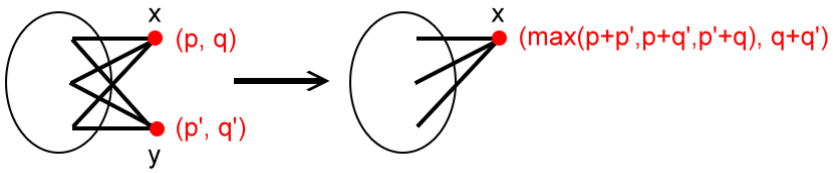
To eliminate y, we can either choose x or chose y, or neither of both.

(a) Choose x: $p = p + q'$.

(b) Choose y: $p = p' + q$.

(c) Neither: $q = q + q'$. So $p = max(p + q', p' + q)$, $q = q + q'$.

(2) MyFriendsAreYourFriends



To eliminate y, we can choose x, choose y or choose both, or neither of both.

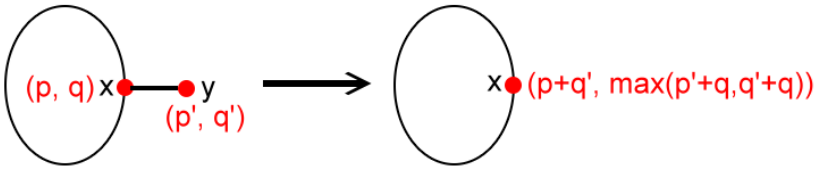
(a) Choose x: $p = p + q'$.

(b) Choose y: $p = p' + q$.

(c) Choose both: $p = p + p'$.

(d) Neither: $q = q + q'$. So $p = max(p + q', p' + q, p + p')$, $q = q + q'$.

(3) IamYourFriend



To eliminate y, we can choose x, or either choose y or choose neither.

(a) Choose x: $p = p + q'$.

(b) Choose y: $q = p' + q$.

(c) Neither: $q = q + q'$. So $p = p + q'$, $q = max(p' + q, q + q')$.

The complexity of this scheme is $O(N)$. This method is capable of solving all subtasks.

Reference

[1] *Konig's theorem*
http://en.wikipedia.org/wiki/K%C3%B6nig%27s_theorem_(graph_theory)