✖

Interview Questions (optional)

3/3 points earned (100%)

Excellent!

Retake

Course Home

✓ 1 / 1 points

1.

Social network connectivity. Given a social network containing n members and a log file containing m timestamps at which times pairs of members formed friendships, design an algorithm to determine the earliest time at which all members are connected (i.e., every member is a friend of a friend of a friend ... of a friend). Assume that the log file is sorted by timestamp and that friendship is an equivalence relation. The running time of your algorithm should be $m \log n$ or better and use extra space proportional to n.

Note: these interview questions are ungraded and purely for your own enrichment. To get a hint, submit a solution.

using the improved version of union find algorithm, call union based on our log file, until we find that the size array after union get to n. This will tell us that this root tree contains n element. Which means these user are all connected

Thank you for your response.

Hint: union−find.

✓ 1 / 1
points

2.

Union-find with specific canonical element. Add a method $\texttt{find}()$ to the union-find data type so that $\texttt{find(i)}$ returns the largest element in the connected component containing i. The operations, $\texttt{union}()$, $\texttt{connected}()$, and $\texttt{find}()$ should all take logarithmic time or better.

For example, if one of the connected components is $\{1, 2, 6, 9\}$, then the $\texttt{find}()$ method should return 9 for each of the four elements in the connected components.

We can keep a third array called 'Biggest'. To start with each each element is equal to its index. Each time we need to combine two connected component p,q and they already are root. We update Biggest[root] with bigger one of Biggest[p] and Biggest[q].
Then when we need to find() we find its root and return Biggest[root]

Thank you for your response.
Hint: maintain an extra array to the weighted quick-union data structure that stores for each root `i` the large element in the connected component containing `i`.

✓ 1 / 1
points

3.

Successor with delete. Given a set of N integers $S = \{0, 1, \ldots, N-1\}$ and a sequence of requests of the following form:

- Remove x from S
- Find the *successor* of x: the smallest y in S such that $y \geq x$.

design a data type so that all operations (except construction) should take logarithmic time or better.

We keep a status array. Initial all 0. Each time we remove x from S we check x-1 and x+1 see if they are already removed from S. If so we connect x to x-1 or x+1 respectively. Also we keep the biggest array from previous problem. If x+1 have not been remove we return x+1 other with we return Bigget[root(x)]+1

Thank you for your response.

Hint: use the modification of the union−find data discussed in the previous question.