Interview Questions: Union–Find (ungraded) | Coursera

Clip source: Interview Questions: Union–Find (ungraded) | Coursera

Interview Questions: Union–Find (ungraded)

Practice Quiz

Congratulations! You passed!

Grade received 100%

To pass 1% or higher

Interview Questions: Union–Find (ungraded)

Total points 3

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 mlogn 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.

1 / 1 point

Use a union-find data structure with each site representing a social network member. Add unions between sites in time order of friendships being formed. After each union is added, check the number of connected components within the

union-find data structure. If only one, all members are connected. Must keep track of number of unique components. Decreases when a union occurs between different components.

Correct

Hint: union–find.

2.

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

For example, if one of the connected components is {1,2,6,9}, then the find() method should return 9 for each of the four elements in the connected components.
1 / 1 point
public Integer find(Integer[] component) { return Collections.max(Arrays.asList(ArrayUtils.toObject(component))); }

Correct

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.

3.

Successor with delete. Given a set of n integers S={0,1,...,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≥x.

design a data type so that all operations (except construction) take logarithmic time or better in the worst case.
1 / 1 point
class SuccessorWithDelete { private QuickFindUF uf; public SuccessorWithDelete(int N) { uf = new QuickFindUF(N); } public void

remove(int x) { uf.union(x, x+1); } public int successor(int x) { return uf.find(x); } }

Correct

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