HackerRank For Work

Spring_2018_INFO6205_Se... 35 minutes

Question - 1

SCORE: **15 points**

Quick Union

If we are going to apply quick-find algorithm to solve the dynamic connectivity problem until all components are connected, how many times of array operations is necessary.

- (●) N^2
- ( ) NlogN
- ( ) N
- ( ) logN

Question - 2

SCORE: **15 points**

Weighted Quick-Union

What's the worst case of find(), connected() and union() method applying weighted quick-union algorithm with N sites?

- ( ) N
- ( ) NlogN
- ( ) N^2
- (●) logN

Question - 3

SCORE: **15 points**

Quick Find, Quick Union

Statement 1 : Quick-Find union operation is too expensive
Statement 2 : Trees formed in Quick-Union are always flat
Statement 3 : Find / connected operation can be N-array access in Quick-Union, hence it is too expensive
Statement 4 : It takes O(N) array accesses to process one union operation on N objects in Quick-Find
Which statements are true ?

- ( ) All of these
- ( ) Statement 1 and 3
- ( ) Statement 1 and 2
- ( ) Statement 3 and 4
- ( ) Statement 1, 2 and 4
- ( ) Statement 1, 2 and 3

- (●) Statement 1, 3 and 4
- ( ) None of these

Question - 4

SCORE: **55 points**

Is it a valid tree?

Union Find can be used to determine if the input can be valid or not to build a tree. Given n(the number of nodes) and a list of edges (each edge is a pair of directly connected nodes), please implement the functions to help judge if its valid to build a tree. (Recall the definition of a tree mentioned last lecture)
For example:
Given n = 5 and edges = [[0, 1], [0, 2], [0, 3], [1, 4]], return true.
Given n = 5 and edges = [[0, 1], [1, 2], [2, 3], [1, 3], [1, 4]], return false.
(1,1) (0,0) are not edges, so the input will not include this type.
In this question, you will not required to implement the function to solve but you are required to implement the structure of union find based on this.