Week 2 Challenge Problem
Graded Quiz • 1h

Due Nov 4, 2:59 PM CST

✓ **Congratulations! You passed!**

TO PASS 80% or higher

Keep Learning

GRADE
100%

Week 2 Challenge Problem

LATEST SUBMISSION GRADE
100%

1. **Modify the implementation of DisjointSets::find(int i) below so that it performs path compression.**

   5 / 5 points

   If the up-tree array d.s is loaded with a disjoint set such that each element points to the next element until the last element which holds a -1, then the last element is the root and its index in the array is the name of the set.

   After calling find() on one of the elements in the set, the find function should (1) return the name of the disjoint set (the index of its root element) and (2) set that element in the up-tree array and all of its ancestors to point directly to the root.

```cpp
#include <iostream>
#include <vector>

//vector<int> traversal_path;


class DisjointSets {
public:
  int s[256];

  DisjointSets() { for (int i = 0; i < 256; i++) s[i] = -1; }

  int find(int i);
};

/* Modify the find() method below
 * to implement path compression
 * so that element i and all of
 * its ancestors in the up-tree
 * point to directly to the root
 * after find() completes.
 */

/*
int DisjointSets::find(int i) {
  if ( s[i] < 0 ) {
    return i;
  } else {
    return find(s[i]);
  }
}
*/

int DisjointSets::find( int i)
{

 static std::vector<int> traversal_path;

 if( s[i] < 0 )
 {
   for( auto iter = traversal_path.begin() ; iter != traversal_path.end() ; iter++)
   {
     // path compression from i to root node
     int node = *iter;
     s[ node ] = i;

   }

   // return the name(i.e., array index) of Disjoint set
   return i;
 }
 else
 {
   //add currnet node into traversal path
   traversal_path.push_back( i );
   return find( s[i] );
 }

}
//end of function int DisjointSet::find( int i)




int main() {
  DisjointSets d;

  d.s[1] = 3;
  d.s[3] = 5;
  d.s[5] = 7;
  d.s[7] = -1;

  std::cout << "d.find(3) = " << d.find(3) << std::endl;
  std::cout << "d.s(1) = " << d.s[1] << std::endl;
  std::cout << "d.s(3) = " << d.s[3] << std::endl;
  std::cout << "d.s(5) = " << d.s[5] << std::endl;
  std::cout << "d.s(7) = " << d.s[7] << std::endl;

  return 0;
}
```

Run

Reset

✓ **Correct**

```
Tested disjoint set (14, 30, 45, 47, 53, 66, 72, 87, 93, 104, 115, 127, 136, 143, 155).
d.find(30) = 155, expected 155.
After find ran, expected all of these to be set to 155:
d.s[30] = 155.
d.s[45] = 155.
d.s[47] = 155.
d.s[53] = 155.
d.s[66] = 155.
```

```
d.s[72] = 155.
d.s[87] = 155.
d.s[93] = 155.
d.s[104] = 155.
d.s[115] = 155.
d.s[127] = 155.
d.s[136] = 155.
d.s[143] = 155.
After find ran, expected this to be 30:
d.s[14] = 30.
```