HW3: Union-Find

Due: Monday, November 23, at 11:59 PM, via Canvas

You may work on your own or with one (1) partner.

For this assignment you will implement the union-find data structure with path compression and weighted union as we saw in class. Unlike in HW2, the representation itself is not defined for you, so you'll have to define it.

In `unionfind.rkt` I've supplied headers for the functions that you'll need to write, along with two suggested helpers and some code to help with testing.

Your task

First you will need to define your representation, the `UnionFind` data type. Each `UnionFind` represents a "universe" with a fixed number of objects identified by natural numbers.

Then you will have to implement five functions:

```
create    : N -> UnionFind              ; O(n)
size      : UnionFind -> N              ; O(1)
union!    : UnionFind N N -> Void       ; amortized O(α(n))
find      : UnionFind N -> N            ; amortized O(α(n))
same-set? : UnionFind N N -> Boolean    ; amortized O(α(n))
```

(Note: `N` is the natural numbers and α is the inverse Ackermann function.)

Calling `(create n)` returns a new `UnionFind` universe (defined by you) initialized to have `n` objects in disjoint singleton sets numbered 0 to `n` – 1. Given a universe `uf`, `(size uf)` returns the number of objects (not sets!) in the universe—that is, `size` will always return the number that was passed to `create` when that universe was initialized.

Functions `union!` and `find` implement the standard union-find operations: The function call `(union! uf n m)` unions the set containing `n` with the set containing `m`, if they are not already one and the same. `(find uf n)` returns the representative (root) object name for the set containing `n`. The `find` function must perform path compression, and the `union!` function must set the parent of the root of the smaller set to be the root of the larger set. For convenience, `(same-set? uf n m)` returns whether objects `n` and `m` are in the same set according to union-find universe `uf`.

Deliverables

The provided file `unionfind.rkt` (`http://goo.gl/12dFgn`), containing 1) a definition of your `UnionFind` data type, and 2) complete, working definitions of the five functions specified above. Thorough testing is strongly recommended but will not be graded.