# Homework 6: Union-Find and MST

For this assignment you will implement the union-find data structure with path compression and weighted union (WQUPC) as we saw in class. Unlike in homework 5, the representation itself is not defined for you, so you'll have to define it. Then you will use your union-find data structure to implement Kruskal's minimum spanning tree (MST) algorithm.

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

This assignment depends on graph and binary heap implementations, like the ones you did in homeworks 4 and 5, respectively. To make sure everyone starts on a firm footing regardless of how their homeworks 4 and 5 went, we are providing you with compiled versions of working solutions. Extract the `hw6-lib.zip` archive in the same directory as `unionfind.rkt`, since it imports them.[1]

## Union-Find

The first part of your job is to complete the implementation of the `UnionFind` class. In particular:

1. Define the necessary field(s) in the `UnionFind` class.
2. Define the constructor (`__init__`) to initialize your fields.
3. Define the `len` method to return the number of objects.
4. Define the `find` method to return the representative of the given object's set.
5. Define the `union` method to union the two given objects' sets.

Calling `UnionFind(n)` returns a new `UnionFind` universe initialized to have `n` objects in disjoint singleton sets numbered 0 to $\mathtt{n} - 1$. Given a universe `uf`, `uf.len()` returns the number of objects (not sets!) in the universe—that is, `len` will always return the number that was passed to the `UnionFind` constructor when that universe was created.

Methods `find` and `union` implement the standard union-find operations:

- The method call `uf.union(n, m)` unions the set containing `n` with the set containing `m`, if they are not already one and the same.
- `uf.find(n)` returns the representative (root) for the set containing `n`.

The `find` method must perform path compression, and because the `union` method calls `find`, it (indirectly) performs path compression as well. The `union`

[1] Don't worry that `unionfind.rkt` imports non-existing `.rkt` files; the compiled versions (which *are* there) are sufficient.

method must set the parent of the root of the smaller set to be the root of the larger set, and must update the weight of the larger set.

For convenience, `uf.same_set?(n, m)` returns whether objects `n` and `m` are in the same set according to union-find universe `uf`.

**Advice** You may find it easier to work your way towards a full implementation of WQUPC via some intermediate steps, as we did in lecture. It's much easier to debug something simple, then debug changes to it, than to debug something complex all at once!

## Kruskal's MST algorithm

Once you have a working union-find, you must implement Kruskal's algorithm as a function `kruskal_mst : WuGraph -> WuGraph`. Given any weighted, undirected graph `g`, `kruskal_mst(g)` returns a graph with the same vertices as `g` and edges forming a minimum spanning forest, using the algorithm as described above.

In order to consider the edges in order by increasing weight, Kruskal's algorithm requires sorting the edges by weight. I've provided a helper function `_get_all_edges_increasing`, which takes a `WuGraph` and returns a vector of its edges in order of increasing weight. Your `kruskal_mst` function should use this function to get the vector of edges to iterate over.

## Honor code

Every homework assignment you hand in must begin with the following definition (taken from the Provost's website[2]; see that for a more detailed explanation of these points):

```
let eight_principles = ["Know your rights.",
     "Acknowledge your sources.",
     "Protect your work.",
     "Avoid suspicion.",
     "Do your own work.",
     "Never falsify a record or permit another person to do so.",
     "Never fabricate data, citations, or experimental results.",
     "Always tell the truth when discussing your work with your instructor."]
```

If the definition is not present, you receive no credit for the assignment.

**Note:** Be careful about formatting the above in your source code! Depending on your pdf reader, directly copy-pasting may not yield valid DSSL2 formatting. To avoid surprises, be sure to test your code *after* copying the above definition.

[2] http://www.northwestern.edu/provost/students/integrity/rules.html

## Deliverables

Your completed `unionfind.rkt`, containing

- a complete definition of the `UnionFind` class with its fields, constructor, and methods,
- a working implementation of Kruskal's algorithm, and
- tests to cover all cases and be confident of your code's correctness.
- the honor code.

Your code will be evaluated for correctness, resource efficiency, thoroughness, code reuse, and style.

## Submission

Your homework must be submitted via Canvas.