Pensieve header: A package to manage equivalence relations.

An Equivalence Relation Object (ERO) is a symbol er with two internal data fields:

* er@complexity: A function that for any index returns a sortable object (perhaps just a number). Lower objects are betters.

* er@down: A list of indices; the *n*th entry is the index of an element with lower complexity or lower index, or 0 if the element is terminal.

Manipulators:

EROMake[er,n, comp]: Initiates an n-element ERO er, with complexity function comp (defaults to Identity).

EROPeek[er,n]: Returns the best equivalent of element #n, and improves er along the way.

EROAdjoin[er,n1⟼n2]: Decrees that n1 is equivalent to n2, update er.

(Alt) In[ ]:=
```
EROMake[er_, n_Integer] := EROMake[er, n, Identity];
EROMake[er_, n_Integer, comp_] := (
  er@complexity = comp;
  (er@down = Unique[er]) = Table[0, n];
  er)
```

(Alt) In[ ]:=
```
EROPeek[er_, n_Integer] :=
  If[(er@down)[[n]] == 0, n, (er@down)[[n]] = EROPeek[er, (er@down)[[n]]]];
```

(Alt) In[ ]:=
```
EROAdjoin[er_, n1_Integer ⟼ n2_Integer] := Module[{m1, m2},
  m1 = EROPeek[er, n1]; m2 = EROPeek[er, n2];
  Switch[Order[{(er@complexity)[m1], m1}, {(er@complexity)[m2], m2}],
   0, m1,
   1, (er@down)[[m2]] = m1; EROPeek[er, m2],
   -1, (er@down)[[m1]] = m2; EROPeek[er, m1]
  ]]
```

(Alt) In[ ]:=
```
Dynamic[EROMake[er0, 10]@down]
```

(Alt) Out[ ]=

(Alt) In[ ]:=
```
er0@down
```

(Alt) Out[ ]=
```
er0[down]
```

(Alt) In[ ]:= `EROPeek[er0, 5]`

(Alt) Out[ ]= 5

(Alt) In[ ]:= `EROAdjoin[er0, 3 ⟼ 7]`

Set: er0[down] in the part assignment is not a symbol.

(Alt) Out[ ]= 7