Deleting Vertices and Relations in the `Posets` Julia Package

Deleting Vertices

Deleting vertices from a poset is somewhat different from deleting vertices in a graph. When a vertex is deleted from a graph, the vertex and all edges incident with that vertex are removed. Similarly, when a vertex is removed from a poset, the relations between all the remaining verices remain unchanged. However, this is not the same as simply deleting a vertex and its edges from the poset's Hasse diagram.

Consider the poset on the left in Figure 1. Mathematically, deleting vertex 2 from this poset results in the poset in the middle of the figure. In the Hasse diagram we have an edge from 1 upward to 3 because $1 \prec 3$ in the original poset, and so we still have $1 \prec 3$ after the deletion. Similiarly, the relation $4 \prec 3$ is preserved.

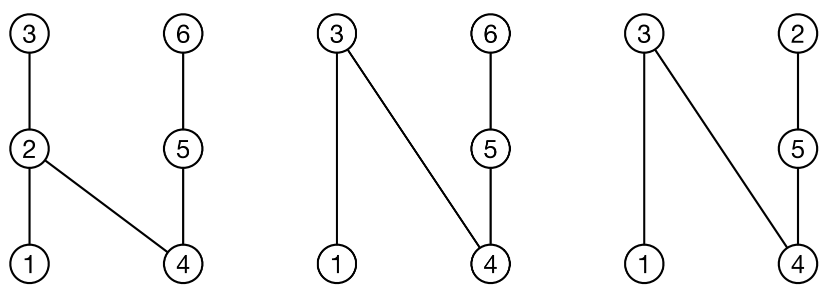

Figure 1: Vertex deletion in `Posets`.

However, the Julia `Posets` package is based on the `Graphs` package. A key convention of graphs (and hence of posets) in these packages is that the vertex set is always of the form $\{1, 2, \ldots, n\}$. If vertex n is deleted, no special action needs to be taken. But if a vertex k (with $k < n$) is removed, then the name n is no longer valid by the naming convention. In this case, the vertex formally named n is renamed k (the label of the deleted vertex). As a result, the result in `Posets` of deleting vertex 2 in Figure 1 is the poset on the right.

This is illustrated in Julia as follows:

```
julia> p = chain(3) + chain(3);
julia> add_relation!(p, 4, 2);
julia> rem_vertex!(p, 2);
julia> collect(relations(p))
5-element Vector{Relation{Int64}}:
 Relation 1 < 3
 Relation 4 < 2
 Relation 4 < 3
 Relation 4 < 5
 Relation 5 < 2
```

Deleting Relations

Deleting a relation from a poset is complicated. The simplest case is the removal of a relation $a \prec b$ where b is a cover of a. In this case, it is possible just to remove the single relation $a \prec b$ and make no changes to any other relation. This is illustrated in Figure 2 in which we delete the cover relation $2 \prec$ from the linear order $1 \prec 2 \prec 3 \prec 4$.

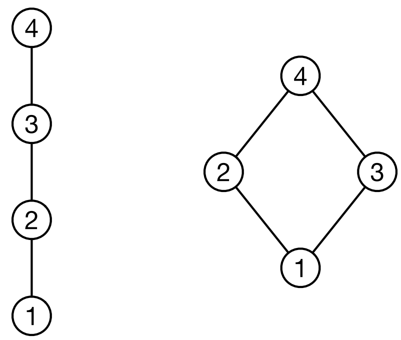

Figure 2: Deleting a cover relation, $2 \prec 3$, from a poset.

The following Juila code[1] implements the action of deleting $2 \prec 3$ from a 4-element chain:

```
julia> p = chain(4);
julia> g = DiGraph(p.d); # This makes a copy of the graph
julia> rem_edge!(g, 3, 4);
julia> q = Poset(g);
julia> collect(relations(q))
5-element Vector{Relation{Int64}}:
 Relation 1 < 2
 Relation 1 < 3
 Relation 1 < 4
 Relation 2 < 3
 Relation 2 < 4
```

It is an easy exercise that deleting a single cover relation from a poset yields a poset.

The situation is different for non-cover relations. For example, consider the linear order $1 \prec 2 \prec 3 \prec 4 \prec 5$. Suppose we wish to delete the relation $2 \prec 4$. If we only delete that one relation, we would still have $2 \prec 3$ and $3 \prec 4$, so omitting $2 \prec 4$ would result in a violation of transitivity.

More generally, suppose we wish to remove the relation $a \prec b$ from a poset. If there is an element x with $a \prec x \prec b$ we cannot delete just $a \prec b$ and keep both $a \prec x$ and $x \prec b$. There is no *a priori* reason to prefer one of $a \prec x$ or $x \prec b$ for deletion (or retention). Hence, it is a design decision that when removing a relation $a \prec b$ we also remove all relations $a \prec x$ and $x \prec b$ for all x that are between a and b. Again, one can prove that the result of this is a poset.

[1] Users should not directly access the internals of a `Poset` object. This is just to illustrate.