Assignment 3

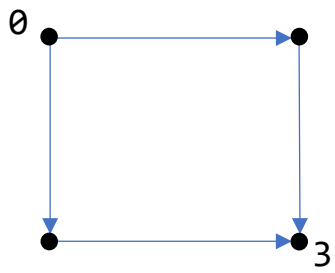
Create a Prolog knowledge base named `graphs.pl` to submit your work for this assignment. The purpose of this assignment is to gain practice implementing Prolog search so that you can traverse a graph structure in general. This work could be extended to knowledge ***graphs***.

---

1) Represent a graph using predicate `arc(X, Y)` where `X` and `Y` are node labels.

Node labels can just be numbers, so for a graph with n nodes, use labels `0` to `n-1`. Make the edges in the graph starting as arcs. For example, an arc from node `0` to node `1` is the Prolog fact `arc(0, 1)` but not `arc(1, 0)`.

Make statements in your knowledge base to have two disjoint paths from node 0 to node 3. In other words, assign directions on the edges of a square that flow arcs from one corner to the opposite corner.



2) Write a recursive predicate `connected(X, Y)` to test if there exists a directed path from one node X to another node Y. The predicate should evaluate to true or false.

The connected predicate should work for any directed graph in general. So, if the set of facts for which arcs exist changed, your implementation should evaluate to true if it is possible to recurse on a sequence of arcs from X to the other final node Y. It should evaluate to false if there is no such directed path.

3) Change your graph from directed to simple edges, and update your version of `connected` so that functionality still works as expected.

Your graph should now have arcs in both directions to represent edges correctly. For example, if the two nodes 1 and 0 have an edge between them, then there should be Prolog

facts `arc(0, 1)` and `arc(1, 0)`. To keep your sanity, order your arc statements in the same way we ordered coordinate pairs at the beginning of the semester (by ascending first element, and second element breaks ties).

You will need some extra parameters to update your connected predicate to deal with controlling recursion properly. Use any technique you can imagine to get the recursion under control. Full marks for full functionality that would work for any graph in general.

Rubric

10 marks total

- [1 mark] directed graph representation
- [2 marks] implement `connected` to work on directed graph
- [1 mark] simple graph representation
- [2 marks] implement `connected` to work on simple graph
- [4 marks] implement `connected` to work on simple graphs in general

Submit `graphs.pl` before Dec 7 end of day.