River Crossing Problem

Daniel Nguyen

May 2020

1 Introduction

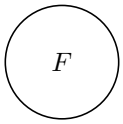
The River Crossing Problem is a fun brain-teaser first introduced to me by the online medieval role-playing game RuneScape.

The problem is as follows: a farmer, a chicken, a wolf, and a bag of grain all sit on one side of a rickety bridge. The farmer needs to take himself and the other three objects to the other side of the bridge. Because of the decrepit nature of the bridge, the bridge can only handle the weight of up to two of the four objects on the bridge at a time. Furthermore, the animals (and the grain!) refuse to cross the bridge without the farmer. As an added difficulty, the wolf cannot be left with the chicken on the side of the bridge opposite the farmer, as the wolf will eat the chicken. Similarly, the chicken cannot be left with the grain on the side of the bridge opposite the farmer, else the chicken will eat the grain. Our goal is to find a list of steps which the farmer can use to move all four objects to the other side of the bridge.

2 Solution

We shall model the problem as a directed graph, where each node represents a possible configuration for the objects on each side of the bridge. Let F represent the farmer, C represent the chicken, W represent the wolf, and G represent the grain. Let $X = \{F, C, W, G\}$.

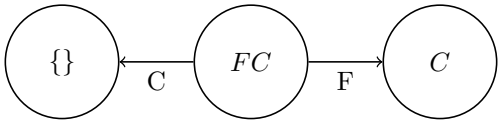
The state of each node is sufficiently represented by the set $V \subseteq X$, where each element of V is an object that is on the starting side of the bridge for a particular configuration. For example:



represents the state where the farmer is on the starting side of the bridge, and the remaining objects are on the other side of the bridge. For brevity, the curly braces and commas separating elements that are usual to set notation shall be omitted.

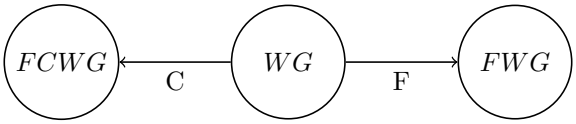
If a node represents the state where a wolf would eat a chicken or a chicken would eat a grain, we shall consider that node to have no outgoing edges. From that state, the farmer cannot make a move such that the end goal is recoverable. We shall call these sink states invalid. We shall call all other nodes valid.

If $F \in V$ for a valid node, then there are $|V|$ transitions out of that node: the farmer may take any of the other objects across the bridge, or he may take no one else across the bridge. We may label the transitions out of this node using elements of V, where the label on each transition represents the second object that the farmer takes with him across the bridge. If the transition is labeled F, then the farmer only takes himself. For example:



shows all the transitions out of the state labeled $\{F, C\}$.

If $F \notin V$ for a valid node, then there are $|X \setminus V|$ transitions. If we label the transitions in this case by elements of $X \setminus V$, we again get the interpretation that a transition is labeled with the object that the farmer is taking across the bridge. The diagram:



shows all the transitions out of the state labeled $\{W, G\}$.

Finally, while the goal node is a valid node, we will restrict that node to having no outgoing edges since it does not make sense to leave the goal state once it has been reached.

The graph given by these rules is drawn below. For clarity, the start node is drawn in blue, the goal node is drawn in green, and all invalid nodes are drawn in red. Also, all edges (x, y) such that (y, x) is also an edge are drawn as one double-arrow edge since both edges will have the same label. Finally, the node represented by $V = \{F\}$ is omitted since it has no outgoing or incoming edges.

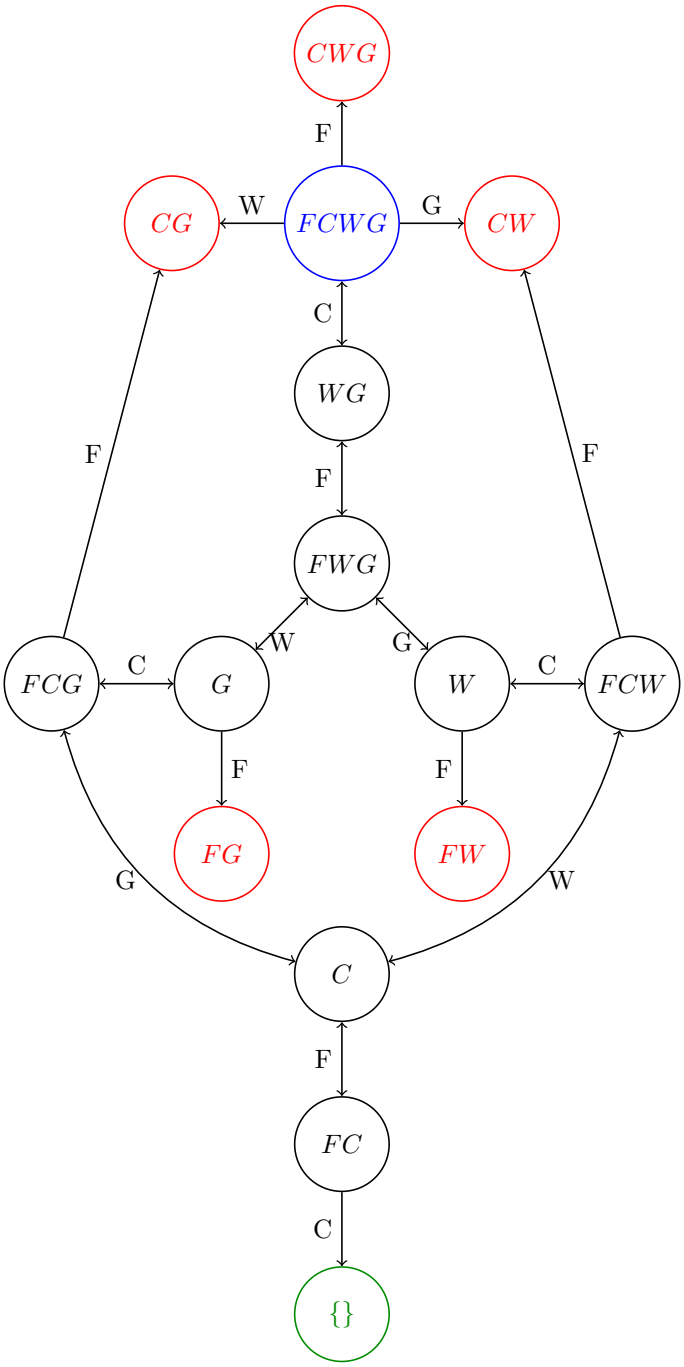


Figure 1: River Crossing Graph

Using a shortest-path algorithm with edge weights of 1 (DFS, BFS, etc.) to find the shorted path from $\{F, C, W, G\}$ to $\{\}$ yields two solutions, each consisting of 7 steps:

$$C, F, G, C, W, F, C$$

and

$$C, F, W, C, G, F, C.$$

All comments, questions, or concerns can be directed to the email address printed to standard out as a result of inputting the URL of the page containing this document into the following Python3 code:

```
import re
url = input()
regex = re.search("(.+).github.io", url)
print(regex.group(1) + "@gmail.com")
```

The LaTeXcode for this document can be found online, here:
https://www.overleaf.com/read/qkkcwrgkysrs.