Computational Thinking and Program Design

Individual Class Project

11027241 (Jason)

I. The Problem :

The problem is to bring a man, an ape, a bull, a rhino, a goat, and a cabbage all from the east side of a river to the east side in a smallest number of boat trips. There are several details about this problem.

1. The boat can take at most two entities.
2. Only the man and the ape can row the boat.
3. The ape cannot stay alone with the bull without the man.
4. The ape cannot stay alone with the goat without the man.
5. The bull cannot stay alone with the rhino without the man.
6. The goal cannot stay alone with the cabbage without the man.

II. Data Abstraction:

System States:

The goal of this data abstraction step is to transform the human-readable problem to a machine-readable form. We define the state of this system by a 7-tuple that including man, ape, bull, rhino, goat cabbage and boat of Boolean values—(E)ast and (W)est. There are total of 2^7 equals 128 states.

Links:

We model each state by a node and connect a pair of nodes by a link, if the two states can reach to one another directly.

As condition 3 (E, W, W, *, *, *) and (W, E, E, *, *, *) are illegal, where * means “don’t care”.

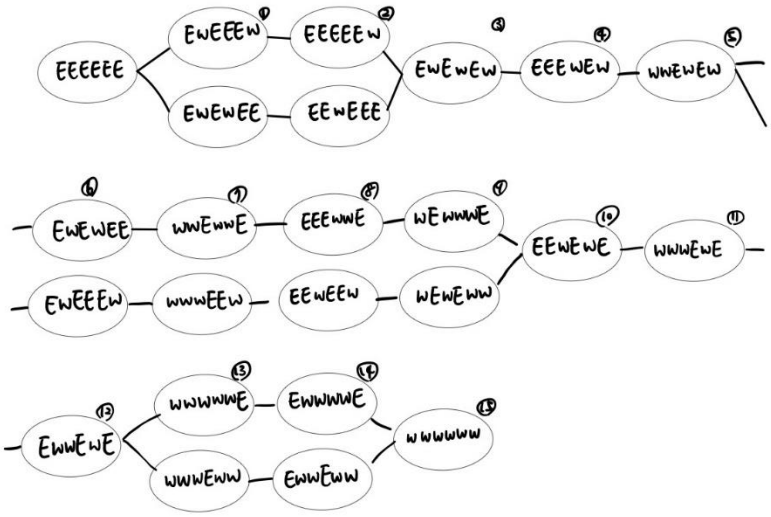
As condition 4(E, W, *, *, W, *) and (W, E, *, *, E, *) are illegal.

As condition 5(E, *, W, W, *, *) and (W, *, E, E, *, *) are illegal

As condition 6(E, *, *, *, W, W) and (W, *, *, *, E, E) are illegal.

A shortest-path problem:

By these legal states as nodes and linking two states which can reach each other directly, we come up a graph for this problem.



This are all possible paths I can image

III. Algorithm Design

This project needs a shortest-path algorithm to solve the shortest-path problem. The input to the algorithm is start node(‘EEEEEEE’) and the end node(‘WWWWWWW’), also the graph needs to input the algorithm. The output is a list of nodes from the start node to the end node, and the length of path is the shortest path to solves this problem.

IV. Program Design:

Function	Inputs	Outputs
main ()	None	Print out the solution to this problem.
genStates ()	None	Return a set of all possible states
genGraph ()	A set of all possible states	Return a graph consisting of only legal states
isAStateLagel ()	A state	If the state is legal in this problem
isNeighbor ()	Two states	If the two states are legal to be the neighbor, then

		return true
findShortestPath ()	A graph, a start node and a end node	List of nodes
printPath ()	A list of nodes	Print the steps of solve this problem

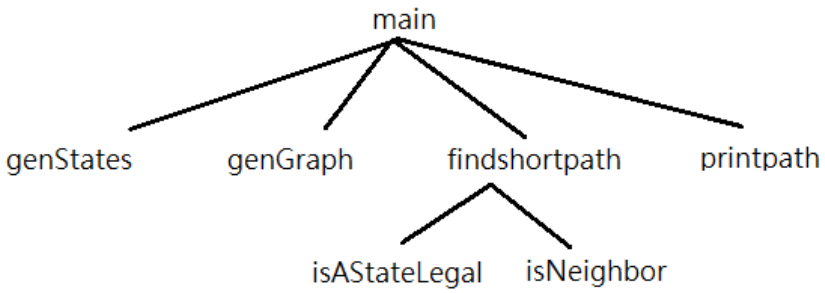


Figure 1. The relationship among these functions.

Function	Inputs	Outputs
addNode ()	A graph and a node which may or may not be in the graph	If the node is already in the graph will be updated by including this node and return True
getNodes ()	A graph	Return the set of nodes in the graph
isLinked ()	A graph and two nodes in the graph	If there is a link between the two nodes return True, otherwise return False
addLinks ()	A graph and two nodes (n1 and n2) which may or may not be in the graph	If any node is not in the graph, add the node(s) to the graph. The graph will be updated by including a undirected link from n1 to n2 and return True.

V. Python Implementation:

- I implement the Boolean by strings ‘E and ‘W’ for its readability.
- In this project, I use seven characters to create a string it show each state.
- Design function (isAStateLagel) to judge every state is legal or not.
- Another one function (isNeighbor) can filter two input states are illogical.
- Use findShortestPath to find out the path, and print all the steps.