***Class Project (A Red-Sea-Crossing Problem) Report***

***ZHANG Yubo (19079784D)\_***

**1, Problem description:**

Moses is bringing seven other people with him from Egypt to Israel which are separated by Red Sea. Besides Moses, there are Pharaoh and two families: (1) Ahab(husband), Jezebel(wife)and a servant and (2) Ananias(husband), Sapphira(wife)and a servant. Further details below:

•They ride on a boat that accommodates only two people traveling between Egypt and Israel.

•Only Moses, Pharaoh, Ahab, and Ananias can operate the boat.

•If Moses is not present, Pharaoh will beat up all others with him.

•If Ahab is absent from his wife and servant, Ananias will beat them up.

•If Ananias is absent from his wife and servant, Ahab will beat them up.

The problem is to bring them from Egypt to Israel without having anyone beaten up in a minimum number of trips.

**2, Data abstraction:**

1. *Description of the states, nodes, links and shortest-path:*

This program defines the state of boat and 8 people by a 9-tuple of Boolean values—(E)east and (W)west, which indicates the current locations of the boat and the six people. The 9 Boolean values are ordered in this way: the boat, 'Moses', 'Pharaoh', 'Ahab', 'Jezebel', 'servant of Ahab', 'Ananias', 'Sapphira', 'servant of Ananias'.

This program models each state by a node and connect a pair of legal nodes by a link, if the two legal states can reach to one another directly according to the legal states and legal move requirement.

At last the problem is equal to find a shortest path (with the least links) from node ‘EEEEEEEEE’ to node ‘WWWWWWWWW’ using a graph (in the last page) with nodes and links.

1. *A graph of section B’s adjacency list is at the end of this report*
2. *Data types required:*

list\string\set\dictionary\adjacency list\ undirected graph like:

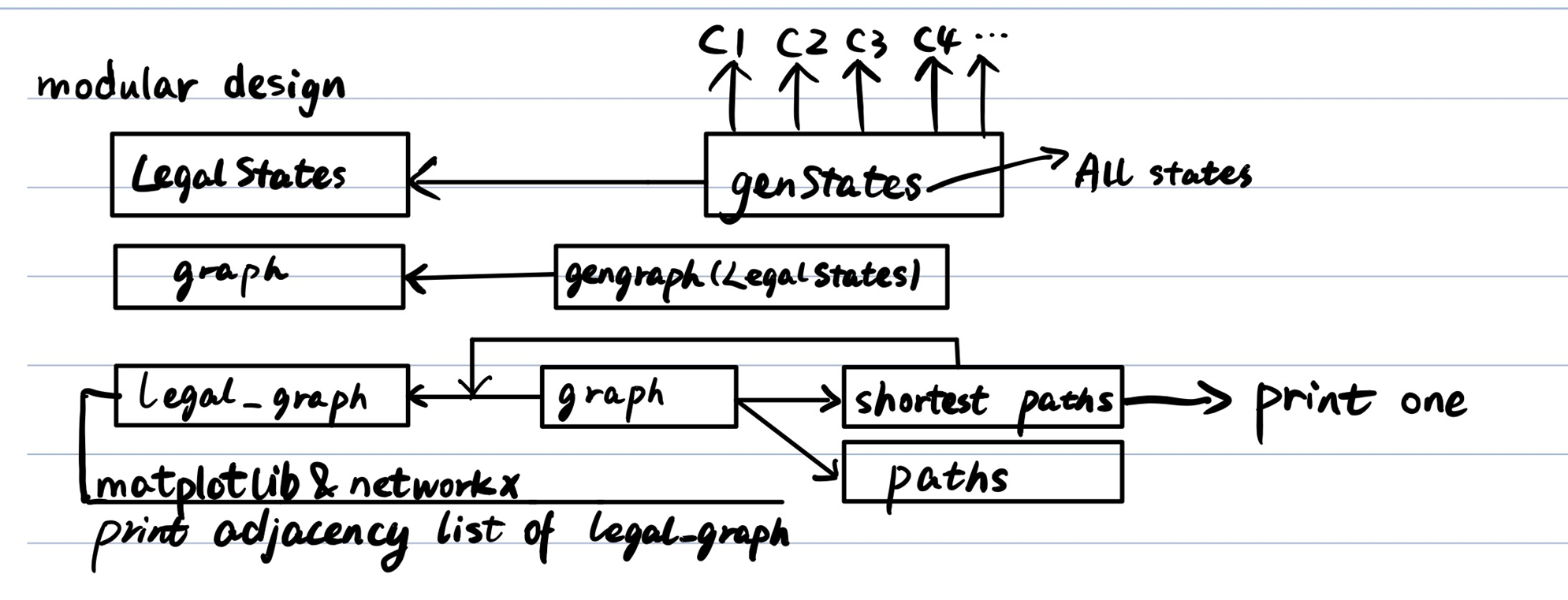
A Boolean value for the location A list of 9 Boolean values for each state. A graph for the relationship among the states. A list of legal states as a path

A list of all paths and A list of all shortest paths

**3, The algorithm needed to solve the graph problem**

Recursive algorithm: This program finds paths and a shortest path (all E to all W) from an adjacency list (graph) using Recursive algorithm, and then filter all shortest paths by the length from all paths

**4, A modular design of the program**



|  |  |  |  |
| --- | --- | --- | --- |
| Key Functions | Inputs | Outputs | Comments |
| genStates() | None | A list of all possible states | Generate a list of all possible states |
| format\_print() | A function | states | To print six states per line |
| isAStateLegal()/c1()/c2()/c3()/  c4()/……/ other\_states() | A state | Yes or no | Check whether the state meets the requirements according to the given conditions. |
| genGraph() | A list of all legal states | Graph: An adjacency list for all legal states | Generate a graph of all legal states |
| nextStates() | A legal state and legal states list | A list of neighboring nodes for a legal state | Generate a list of neighboring states of a legal state |
| neighborNode() | 2 states | Yes or no | Check whether 2 states are neighbors |
| find\_all\_paths() | Graph | All solution paths (a list of lists with states) | Generate all paths for (E,E,E,E,E,E,E,E,E)–>(W,W,W,W,W,W,W,W,W) |
| findShortestPath() | Graph | A solution path (a list of states) | Generate a shortest path for (E,E,E,E,E,E,E,E,E)–>(W,W,W,W,W,W,W,W,W) |
| printShortestPath() | A solution path | A human-readable path | Print a human-readable solution |

**5, A Python implementation of the data types**

A Python Boolean value for the location

A Python list of 9 Boolean values for each state.

A Python list of all legal states

A Python list of legal states as a path

A Python list of all paths

A Python list of all shortest paths

Especially, Python dictionary can be used to form the adjacency list. The key is a legal state and the value is a list of other legal states that have link with this state.

And we can convey this adjacency list to a graph (in the last page) using Python networkx and matplotlib wheels.

