# CS-171 Wumpus World Final AI Report

**Team name________Primadonna______________________________________________**

**Member #1 (name/id) Haochen Zhou 23567813 Member #2 (name/id) Zhuoyi Wu 17547497**

## I. In about 1/2 page of text, describe what you did to make your Final AI agent "smart."

Judge the safety: The first thing for our agent can do is that it can determine which room in the cave is safe or not by obtaining information from its each move. At first, our agent can observe stench and breeze in each room to determine the safety of it. If the agent cannot observe both stench and breeze in a room, it will know that the directly adjacent squares of the room will be safe and our agent will be allowed to move there. In addition, in a room, if our agent cannot observe a breeze (Breeze=False), it will store the directly adjacent room to "not Pit" in its knowledge base. If the agent cannot observe a stench (Stench=False) in a room, it will store the directly adjacent squares of the room to "not Wumpus" in its knowledge base. Then if when a room is in both "Not Pit Set" and "Not Wumpus Set" (find the intersection of not Pit and not Wumpus), the room will be added into "All_safe Set". Our agent thus can know more "safe" room in the cave to go to.

Search algorithm: The search algorithm we use is Depth First Search. The depth is increasing while the search is being processed and more safe nodes are judged, till all safe nodes found are visited or the gold are found. This algorithm can make agent visit as many rooms as possible. Meanwhile, once the agent bumps the wall, it will remember where the bound is, and so it will not bump the same wall next time.

Kill Wumpus: If agent detects stench but no breeze at the starting point (if there is breeze the agent will climb out directly to avoid any risk), it will shoot an arrow to the right; if it hears the scream, it will know the Wumpus has been killed and will move to the right in next step. Otherwise, it will know the position of Wumpus. After recording the position of Wumpus (row 2 col 1) and adding other nearby nodes into Not_Wum_Set, the agent will move to the right, too. If agent detects a stench in two different rooms except the starting point, it will know where the Wumpus is and kill it by shooting an arrow. Knowing the position of the Wumpus or killing it can enhance the chance of finding the gold.

## II. In about 1/4 page of text, describe problems you encountered and how you solved them.

First of all, I have no idea about how to determine more safe rooms that allow our agent to go. Then, after learning the idea of propositional logic in class, I was going to use a sentence "no stench in a room" to infer "not pit" and "no breeze in a room" to infer "not Wumpus" in their directly adjacent rooms. Finally, if a room has both "not Pit" and "not Wumpus", then we can infer anther sentence that "the room is safe". Thus, our agent can go more safe rooms.

Besides, backtracking is also a main problem for me that I encountered in the lab. At first, I want to use a recursion that enables our agent to backtrack if there are no more safe rooms surrounding it. However, it is more complex than I think and it is hard to achieve. Thus, finally I try to use the dictionary recording each step's and its previous step's information (that is, {key: one note visited, value: its father node in search process}) in order. And then the agent can go back according to the dictionary.

## III. In about 1/4 page of text, provide suggestions for improving this project.

One suggestion to improve the project I wrote is to find the shortest path to come back after the gold is found. What the agent currently do is to record the father node of each node visited. After the gold is found, it will check if there is any directly adjacent node which is closer to start point and has been visited, if yes, go to this node. If not, go back to its father node till reach the starting point.

One way which may be applicable is to use Dijkstra algorithm, csto find the shortest path among all safe nodes that are already known. However, since each turn also costs 1 point, it is not easy to calculate the path cost from one node to another by code, That's why we have not achieved it yet.

Another suggestion for this project is to write a complete UI using TK or other UI libraries. It is more vivid to show each step in UI like a chess game when users run python3 main.pyc -d.