

# **Applied Artificial Intelligence(DV2557)**

## **Wumpus World Report**

### **Project group 09**

<b>Damineni Sarath Chandra</b>	<b>19980810 0110</b>
<b>Ganesh Bhumireddy</b>	<b>19980629 0194</b>
<b>Sruthi Annapureddy</b>	<b>19970707 1248</b>

#### **Wumpus World Table:**

<b>(1,4)</b>	<b>(2,4)</b>	<b>(3,4)</b>	<b>(4,4)</b>
<b>(1,3)</b>	<b>(2,3)</b>	<b>(3,3)</b>	<b>(4,3)</b>
<b>(1,2)</b>	<b>(2,2)</b>	<b>(3,2)</b>	<b>(4,2)</b>
<b>(1,1)</b>	<b>(2,1)</b>	<b>(3,1)</b>	<b>(4,1)</b>

- We declare and initialize the `destination_list` ( Which is the object of `RoomLinkedList` - an internal class in `MyAgent.java` ).
- This `destination_list` is a linked list that maintains a list of rooms. This list was stored according to the priority such that if any rooms were proved to have pits those rooms were added at the last of the list.
- Initial `destination_list` looks like this

(2,1)

(3,1)

(4,1)

(1,2)

(2,2)

(3,2)

(4,2)

(1,3)

(2,3)

(3,3)

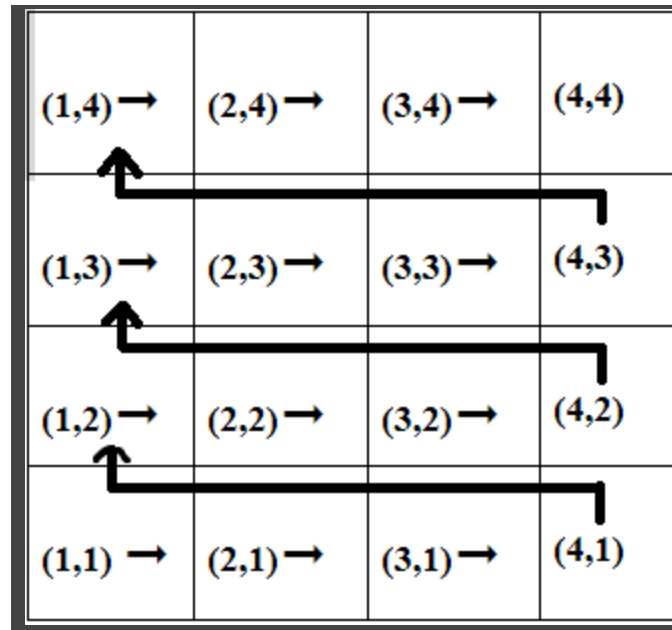
(4,3)

(1,4)

(2,4)

(3,4)

(4,4)



- As, in the initial position player is at position (1,1), (1,1) is not present in the destination\_list. So the present destination is (2,1). After reaching the room (2,1), As (2,1) doesn't have any breeze or stench then destination\_list updates as follows

(3,1)

(4,1)

(1,2)

(2,2)

(3,2)

(4,2)

(1,3)

(2,3)

(3,3)

(4,3)

(1,4)

(2,4)

(3,4)

(4,4)

So, the player will now move to the room (3,1) and this continues until the player doesn't experience any breeze or stench.

- The basic and initial moves of our Wumpus world table is developed in a horizontal direction.
- Destination\_list gets changes when there exists a stench.
- So, when the player experiences a stench then the new destination\_list looks like

(1,2)

(1,3)

(1,4)

(2,1)

(2,2)

(2,3)

(2,4)

(3,1)

(3,2)

(3,3)

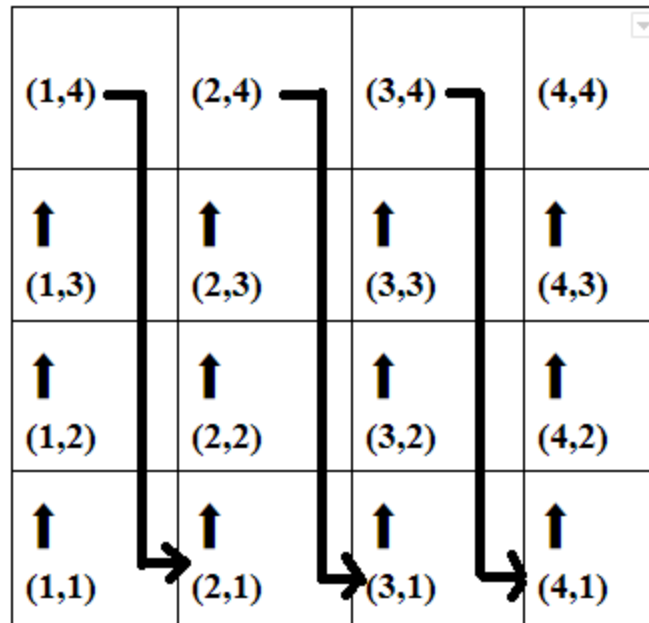
(3,4)

(4,1)

(4,2)

(4,3)

(4,4)



- So, like this destination\_list swaps amongst these two styles every time it experiences a stench.
- destination\_list doesn't contain the rooms that are already visited.
- Every time AI tries to move towards the first room in the destination\_list.
- There are two functionalities where the entire game was built.

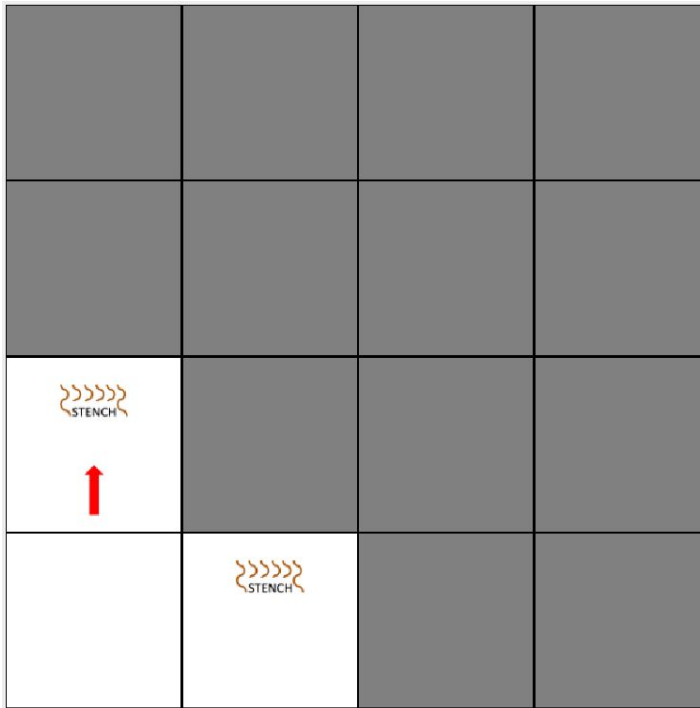
☐ Stench

☐ Breeze

**Stench:** While playing the game, if we found a stench the initial destination\_list will be changed to a new destination\_list. Such that horizontal destination\_list changes to vertical destination\_list and vice versa.

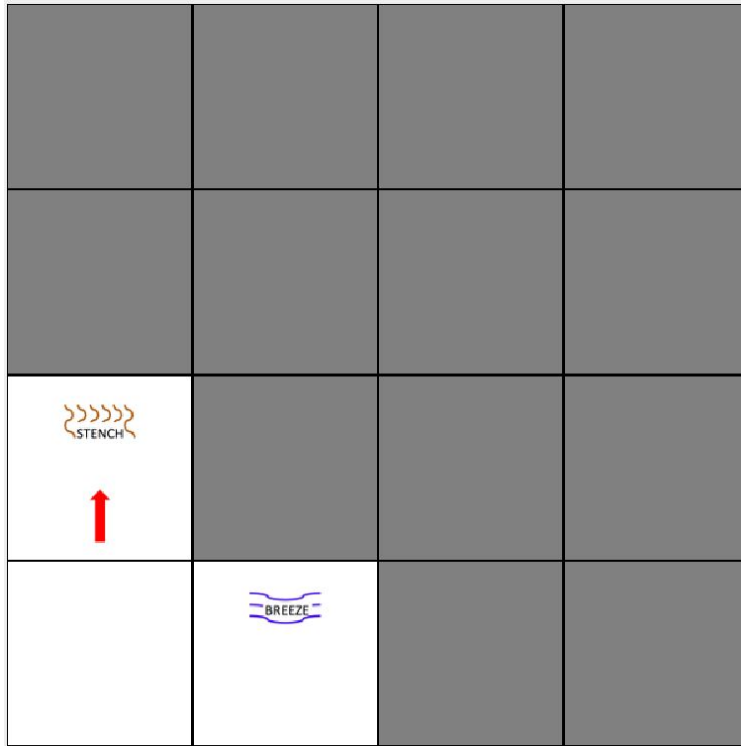
After experiencing the stench AI try to evaluate the position of the wumpus. If AI evaluates the Stench position then AI turns towards the required direction and shoots the arrow and kills the wumpus.

Eg 1:



As Stenches are present at (1,2) and (2,1) AI confirms that wumpus is present at (2,2) so AI changes its direction towards the room (2,2) and shoots the wumpus.

Eg 2: In the above example, the player experiences a stench at (1,2). So, wumpus may be present at (1,3) and (2,2). But, if wumpus is at present at (2,2) then the player experiences a stench at (2,1). But, the player doesn't experienced a stench at (2,1) so AI confirms wumpus at (1,3). Then AI shoots the arrow and kills the wumpus.



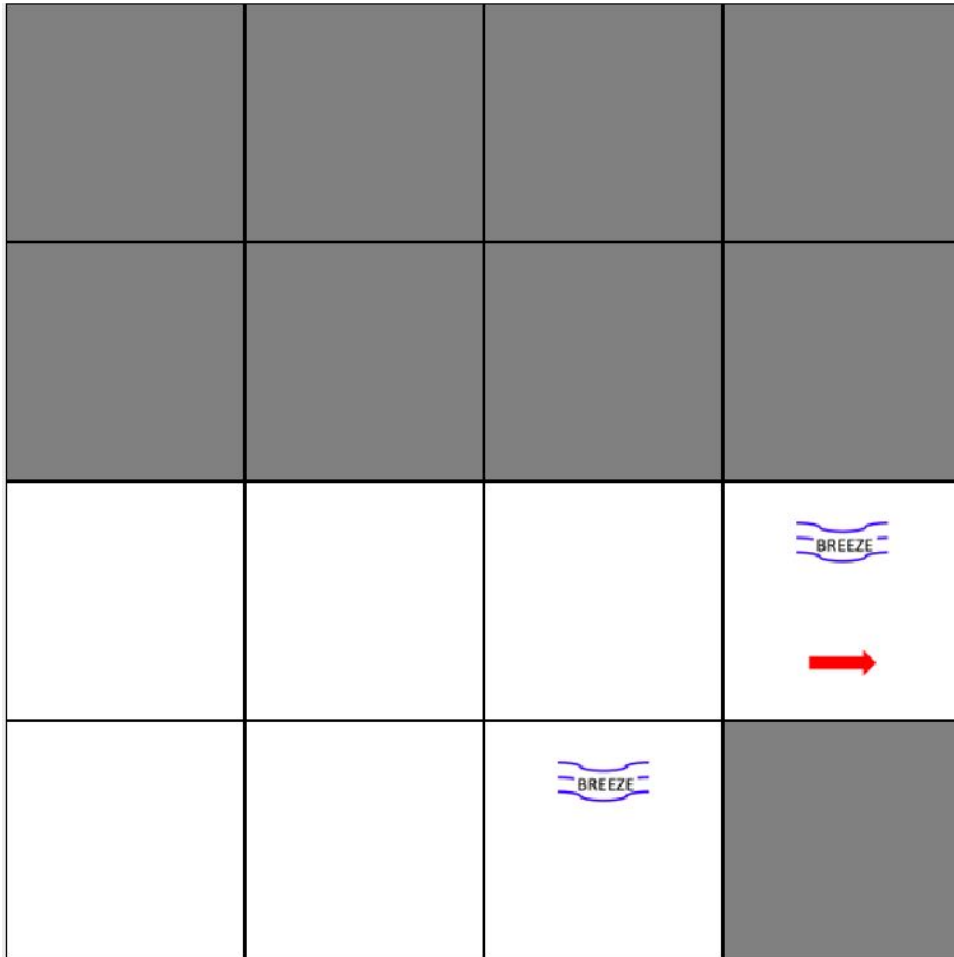
If AI fails to evaluate the position of the wumpus then go back to the previous room and start travels towards the new destination obtained from `destination_list`.

Consider, while playing, we found stench in the horizontal direction of the table then the direction changes to the vertical direction. If we find another stench in at present moving direction then as mentioned, target changes in the opposite direction.

**Breeze:** When the AI player feels the breeze then, at first, player searches for any safe room exists in already visited rooms for this we maintain a `visited_room` linked list that stores a list of all visited rooms. If there exists a safe room then the player moves to it and checks for the next target. Incase no safe room exists then the player has to take the risk so that irrespective of the neighboring room.

**Breeze case 1:** If the player feels the breeze then our AI searches for the safe room in the visited rooms. If safe room found then

Eg:






In this example, the player feels the breeze at the room (4,2) so the player searches for the safe room in the visited rooms.

The nearest safe room available in the room (3,2) so the player comes back to room (3,2) and continues the process again.

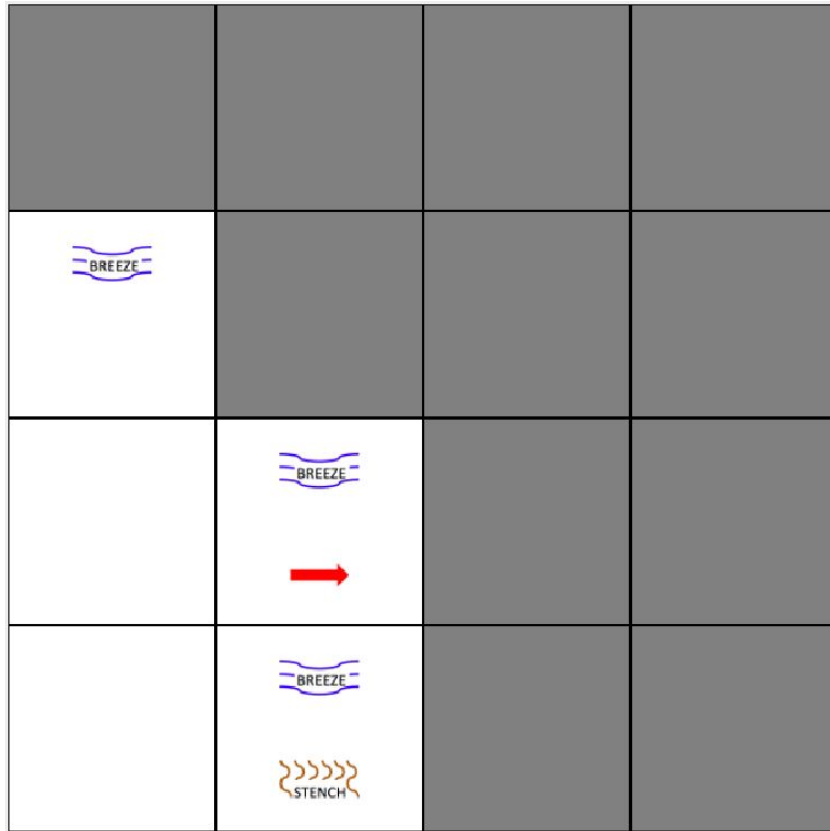
So the next step looks like









**Breeze case 2:** Even though after searching for the safe room, if the player doesn't find any safe room then the ultimate next step has to take risk-based upon probabilities assigned to each room.

Eg:



Such that in the above condition there is no safe room available after the player experiences the breeze at the room (2,2). So the player has to take the risk. Now, the player has to move to the room (2,3) or (3,2).

But, the probability of having the pit at (2,3) is more than the probability of having pit at (3,2) because the player already experienced breezes at both (1,3) and (2,2). So the player moves to the room (3,2) rather than (2,3).

### Assigning Scores to the rooms

We created a two-dimensional array of objects for the class Rooms. For each room we assign a score for those rooms and its valid neighbors.

- All unvisited rooms are given with a score of 0.
- If the room doesn't have any breeze and stench then all the valid neighbors of that room are given score 5.
- If the room has a breeze, stench then all its valid neighbors who have score other than 5 is incremented with score 2.
- While finding pits, all the breezes will be checked whether only one room in its valid neighbors to have a score other than 5. Then that room will be confirmed to have a pit and added to the pit\_list linked list and also added at the last of destination\_list linked list.

## **Who has done What**

Sarath Chandra Damineni: Implementing breeze and stench functionality, Implementing various linked lists to obtain visited nodes, pits found, breezes found. Assigning scores and using them for finding pits, stenches. Steps to implement when no safe room found when experiencing a breeze.

Ganesh Bhumireddy: Implementing breeze and stench functionality, finding paths between rooms and deciding which path is better for the current condition, functionality to find a safe room after experiencing a breeze.

Sruthi Annapureddy: Implementing breeze and Stench functionality, finding pits and stenches based upon the scores of each room, functionality to manage the `destination_list` linked list.

All three of us developed the logic by discussing and analyzing the better way to solve each condition and then we put that in to code.