

AI PROJECT REPORT

WUMPUS WORLD

Submitted by

Sihab Sakib Bayraktar

**PART I**

In this part, we use the BFS Algorithm, known as Breadth First Search algorithm. The method visits and marks all of the key nodes in a network in an efficient and accurate breadthwise manner. This approach explores all nodes nearby to a single node (initial or source point) in a network. Remember that BFS only connects to these nodes one at a time. This algorithm can be implemented by using queue i.e. (FIFO-First in First Out). In this experiment, BFS traverses all the nodes in the graph and keeps dropping them as completed. BFS visits an adjacent unvisited node, marks it as done, and inserts it into a queue. The Time complexity of BFS is O(V + E) when Adjacency List is used and O(V^2) when Adjacency Matrix is used, where V stands for vertices and E stands for edges.

**How BFS Works?**

BFS pseudocode is shown in the below figure. According to that pseudocode,

1. To begin traveling, pick any node at random.
2. Go to the node next to it that hasn't been visited yet.
3. In the boolean array, mark it as visited and show it.
4. Add the node you just visited to the queue.
5. Remove the first node from the queue if there are no nearby nodes.
6. Continue in this manner until the queue is empty.

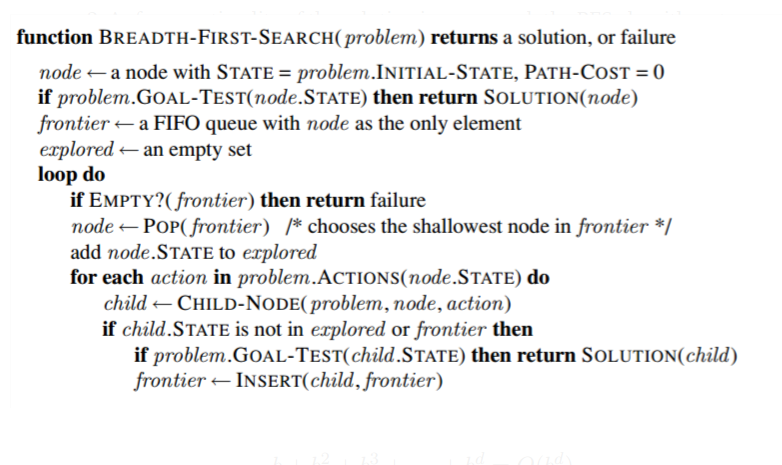


Figure 1. This figure indicates the pseudocode for BFS algorithm.

**Additional Implementation**

For obtaining simplicity and object-oriented manner more, we created some extra classes and Enums. Implementation details can be examined in the common package.

* LinkedList for plan of agent.
* Queue for BFS implementation.
* ArrayList for collection of positions.
* Entity – Enum: Entity is an enum where we used to define elements in this project like Wumpus, Pit, Gold, Senses and Tile conditions.
* Direction – Enum: Direction is an enum where we used to define agents’ position
* Position – Class: Position is a class where we used to define all positions of each element in this project.

An example of 2x2 matrix for Wumpus game transition state system as shown in the below figure.

A whiteboard with writing on it

Description automatically generated with low confidence

**Figure 2.** This figure indicates the problem state transition of 2x2 Wumpus World.

**PART II**

In this part, agent does not know the environment fully. Instead, it has sensors to follow the path reaching the goal (objective). To implement this partial observable environment, we used “Tweety” library, and we used search algorithm implemented in first part. Here, rules of the world was added to knowledge base using the “initRule()” method. Apart from initial rules, there were some rules as well. For explaining those rules, we create some atomic proposition used pre-defined class “Proposition” in “Tweety” library. Those propositions can be explained like this;

* Pij is true if there is a pit in the cell i, j.
* Bij is true if there is a breeze in the cell i, j.
* Wij is true if there is a Wumpus in the cell i, j.
* Vij is true if the cell i, j is visited.
* Gij is true if the cell i, j contains gold.
* Nij is true if the cell i, j. is safe.

In “getAction()” method, firstly we tried to operate senses. And then we tried to add those senses and the related actions into knowledge base. Specifically, if the agent senses the scream which means Wumpus is dead, so we need to remove the proposition Wumpus alive from the knowledge base. In here, if the agent does not sense breeze, next tile is safe, and agent can go there without any danger. And, since the Wumpus is dead, stench is removed from the visited tile and if there is no breeze, we can say that environment of that tile is safe. If the agent receives bump from its sensor, the world size is reduced using the direction of the agent and safe tile that we added into knowledge base is eliminated and agent plan should be cleared. If the agent does not sense bump, agent should move using the “positionUpdate()” method. If the agent feels stench and if the Wumpus is alive, current tile is added to knowledge base. If the breeze occurs, current tile is also added to knowledge base. Also tile in which the agent stays should added as safe and visited tile in knowledge base. If agent does not sense breeze at the same time not sense stench, we added surrounding environment as a safe into knowledge base using “safeEnvironmentAdd()” method. If the agent senses a glitter, which means the agent reaches the goal, so “search()” method is called to go back to initial point. If the agents plan is empty, we have to examine unvisited tiles with the safe ones using “findUnvisitiedSafeTile()” method. In here, “search()” method is called to traverse those unvisited and safe tiles. If the agent plan is empty, which means there is no safe and unvisited tiles, and if the agent has arrow, the “findPossibleWumpus()” method is called to give possible location of Wumpus so that agent can kill the Wumpus. So before the final step, if the agent plan is empty, which means there is no Wumpus and unvisited tiles and no safe locations, the agent should take a risk to reach the gold. For that purpose, firstly, “findUnsafeUnvisitedTile ()” is called then “search()” is called. But here, there is a possibility that can agent die. If the plan is empty before those “if conditions stated above”, which means gold is not accessible, so the agent goes home.