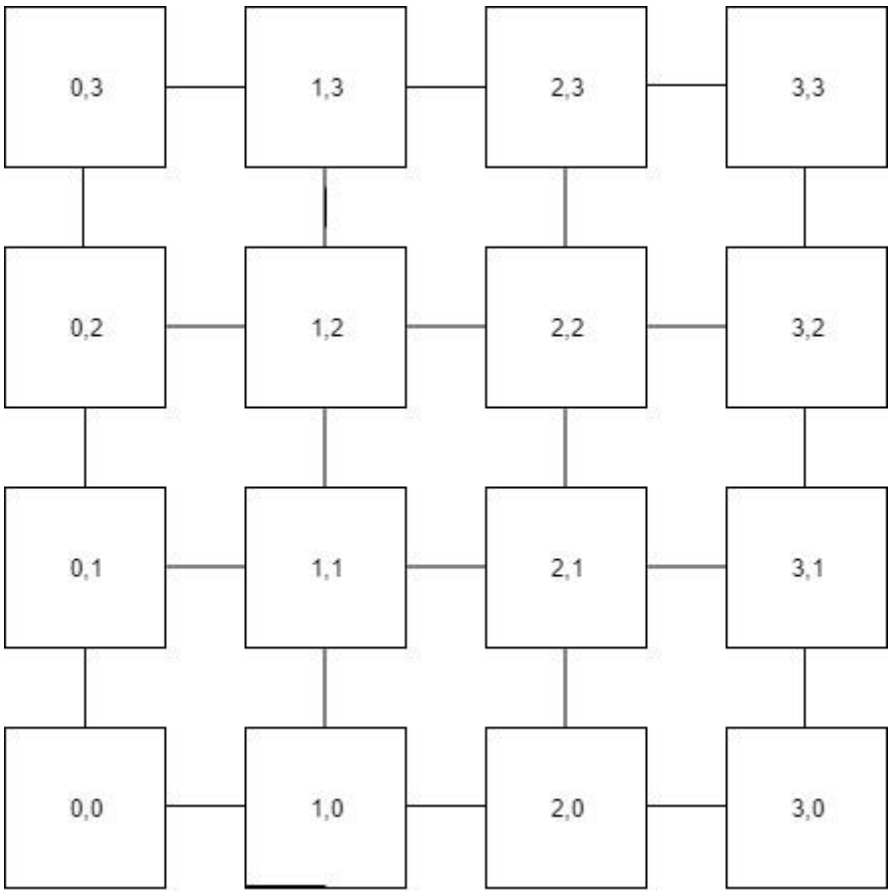
Wumpus World using A* Algorithm

A* Algorithm is used to implement the Wumpus World agent, and this is how we implemented it:

- First, we will construct a graph which has 16 nodes representing the 16 rooms of the Wumpus world.



Every node in this graph is an instance of the Node class which has various instance variables like:'

1. C1, C2, C3 and C4 are the **adjacent nodes**.
2. The Percepts are **Wumpus, Stench, Pit and Breeze**.
3. **Cost** is the numeric value assigned based on the observations.

- Firstly, we assign each of nodes present in the graph to a cost = 10. If the current room in which the player is present in, does not contain any of the above-mentioned percepts like Stench and Breeze then the adjacent rooms will also be assigned a cost = 10.

- If the current room, has a percept of Stench, this implies that, in one of the adjacent rooms, Wumpus is present, and hence we assign a cost of 50,000 to all the adjacent rooms which are unvisited.
- If the current room, has a percept of Breeze, this implies that, in one of the adjacent rooms, Pit is present, and hence we assign a cost of 1,000 to the adjacent rooms which are unvisited.
- If the current room has a pit, we will assign a cost of 10,000 to it.
- When we have a Stench in the first room, we will shoot an arrow to the right, If the Wumpus is dead we will assign a cost = 10 to the adjacent rooms or if the Wumpus is alive we will assign a score of 50000 to the room above and a score = 10 to the room which is on the right to the current room.
- Post, we will use a **updateGraph(),** this function acts a Cost function and Heuristic function as it assigns costs to the rooms that are visited and costs to the rooms that are yet to visit based on the percepts.
- In this step, we will calculate the shortest paths to all the caves from the current cave and we will select the unvisited node which is nearest to the current cave and make a move towards it. This step is performed by the **minPath** () which uses various other functions like **update_Caves_left**(), **extract_min()** and **find_dest_cave().**

1. The distance from the start node to the end node is calculated by adding the costs of all the nodes from the start node to the end node excluding the cost of the start node.

2. By calculating the shortest paths, we will most likely avoid risks like pits and Wumpus while moving towards the unvisited nodes.