**Assignment One**

**Part One**

The objective of the game is for the agents within the in environment is to find the gold coin within the environment. When then the gold coin is found the game ends. Within the environment also is asset of Rocks that block the moves of the agent’s movement if it is in the location the agent is trying to move to.

The task environment is nondeterministic as the agent will not know if it has found the coin, hit a rock or if the location is empty. But once the environment has been created, it is discrete as it will have the coin and rocks in set location that will not move as the agent explores the environment.

For this assignment, the three agents used are:

* Table Based Agent: This follows a hard coded set of movements within a list when in the environment

P: Depends on how well the agent follows the and if it reaches the goal state

E: The size of the environment has a large impact of performance of set actions

A: Can lose correct location if it is prevented from moving due to an obstacle. Can only follow moves provided

S: It can receive its location form the environment. It can detect if an object is within the location its moving to.

* Random Reflex agent: This agent performs a randomly selected moves from the list of moves available to it.

P: Depend on the agent has reached the goal state and how many moves it took to get there.

E: The scale of the environment can affect its performance greatly for time reasons  
A: It uses a randomly selected move each turn that is unaffected by its location

S: Can observe if an object is when a location it’s moving to.

* Model Agent: This agent keeps a percept history of the moves taken within the environment and tries to improve its performance from it.

P: How it improves based on the percept history it has built up  
E: Probability will be needed in the non-deterministic environment of the game due to changing locations each game

A: allows the agent to move (up, down, left or right) within an environment

S: Allows the agent to collect a perception history of an environment to improve future actions.

Regarding the TableBasedAgent in this environment, it won’t perform great as it will follow a set of moves from the table provided but the starting location of the agent is random within the environment. It would also not find objects efficiently to the objects non- deterministic state each game. It will struggle to work in large environments as more data will need to be fed into the table passed to the TableBasedAgent

The RandomReflexAgent will perform better as it doesn’t depend on the starting location to find objects. If the environment is large, it could take large amounts of time to find an object that is it has an unlimited number of moves.

For the environments non-deterministic state, the ModelAgent will struggle to finds objects that change location each in each game instance. If the ModelAgent was in a deterministic environment, it would build up to a point that it would be able to find the coin in the minimum amount of moves possible.

Performance over 1000 instances with 20 moves each in an 8x8 environment:

A black background with white text

Description automatically generated

Performance over 1000 instances with 20 moves each in an 10x10 environment:

A black background with white text

Description automatically generated

It shows that each agents performance dropped significantly when the environment size was increase. With the RandomReflexAgent performance the best by far in this non-deterministic environment.

**Part Two**

The problem statement for this environment is to go through all the possible routes on the search and try to find the most efficient route to the goal state using a selection of different algorithms to find the best path. The goal state is to simply find the gold coin within the search problem. The initial state of the agent will start at (0,0) the coin and rocks will be given random locations to simulate the non-deterministic environment.

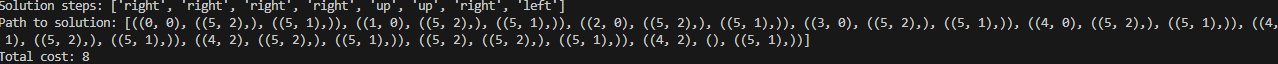
The actions will be added to the agents list of moves within the actions function and it will add them to be usable within the bounds of the environment. The result uses the actions added, to move within the bounds of the environment and will remove the coin from the environment if the agent moves to its location.

The goal test will check if the gold coin has removed from the environment to stop the search. It will also check that the state is being passed to the function correctly and that it isn’t empty.

The path cost will add random values to the total cost depending on the action taken in each move.

**Uniformed search algorithms:**

1. Breadth -first search: This will allow the algorithm to search through each of the successor’s nodes until it reaches the goal state. As the environment size is increased, it will take more time to find the goal state as it will need to ass more nodes to memory. Testing a 12x12 environment could in the short term take a non-deterministic amount of polynomial time to search through if the goal state is at the last search node. It seems to find the goal state within the assignment environment quite easily, it cost range seem to have average range of 9 search movements to find the golf coin within the environment mainly using right and up as the most use search moves in the algorithm used



2. Depth -first search: This find the deepest path of in the environment and will remove that path from the search nodes using LIFO. This can have issues if the environment size is very large and has complex goal states to achieve. However, only each node being used will be stored in memory rather then the whole environment being stored. It could use a depth check limit if the depth of each search is too large for computational resources to search through. It will allow the agent to possibly move through the environment more efficient and will take up less computational resources which is a massive benefit for the environment. It seems to find a large range of total costs with no consistency in term of the cost used, I can get 2 moves of 25 for example.

26 steps:

A screenshot of a computer code

Description automatically generated

5 steps:  
  
A black background with white text

Description automatically generated

3. Iterative Deepening Search: This seems to have a straightforward approach to the search. It seems to have gradual progression in a more reserved manner for example (0,0) -> (0,1) -> (0,2).

This will take the longest amount of time to find the goal state but it will not be as resource consuming as the 2 other algorithms. This will find the gold coin within the search but again will taken more time in the scenario that the coin is say in location (9,9) in a 10x10 environment.

A black background with white text

Description automatically generated

**informed search algorithms:**

1. Uniform Cost Search: This algorithm seems to search the the node path with the lowest cost in the search and gradually increase this until the goal state is reached. This seems like a fair approach it is efficient for the agent to use if the environment is large. It will compare all of the best paths however, this may be difficult within the environments non-deterministic case.

The highest path cost I could see was 14 steps:  
  
A black background with white text

Description automatically generated

**Conclusion**

I would like to add improvement the Game environment, I could add multiple agents to the environment and see to gets the coin first. I could randomly generate more coins randomly within the environment. There could be more and different types of obstacles to add more difficulty.