Project Report | Himanshu Pahwa

Q. How many distinct configurations are possible in this environment?

Since there are 50 possible nodes for each the agent, the prey and the predator hence the total possible distinct configurations of the state (Agent_poisition, Predator_position, Prey_pos) are **125000**.

Q. What states s are easy to determine $U*$ for?

U* is easy to determine for the terminal states since the game ends at the terminal states.
Since our goal is to minimize the utility we can assign a very high number to non favorable terminal states and a 0 to favorable terminal states.

Favorable Terminal State: The state when Agent and Prey are present at the same node in the graph.
For eg: (2,50,2), (3,40,3) and so on.

Non Favorable Terminal State: The state when the Agent and the Predator are present at the same node in the graph.
For eg: (3,3,40), (40,40,5) and so on.

Q. How does $U*(s)$ relate to $U*$ of other states, and the actions the agent can take?

(These expressions come from the Bellman's Equations)

U* (s) relate to U* of other states as follows:

$$U^*(s) = \min_{a \in A(s)} \left[r_{s,a} + \beta \sum_{s'} p^a_{s,s'} U^*(s') \right]$$

And the action the agent takes as follows:

$$\pi^*(s) = \operatorname{argmin}_{a \in A(s)} \left[r_{s,a} + \beta \sum_{s'} p^a_{s,s'} U^*(s') \right]$$

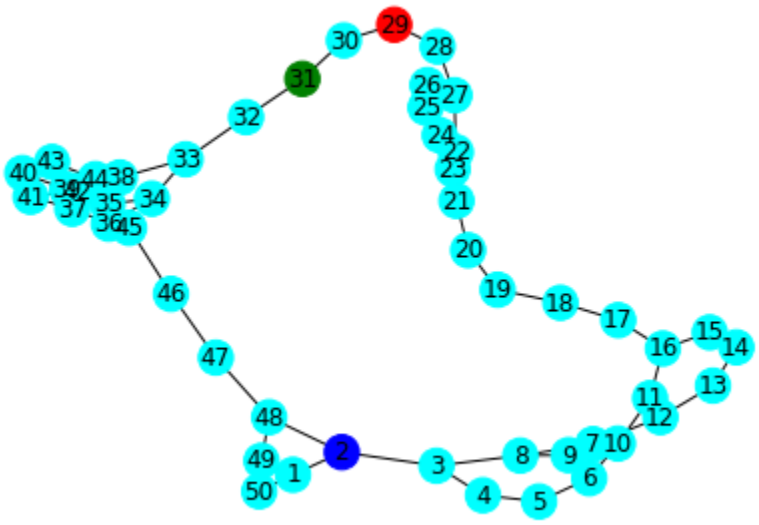
Q. Are there any starting states for which the agent will not be able to capture the prey? What causes this failure?

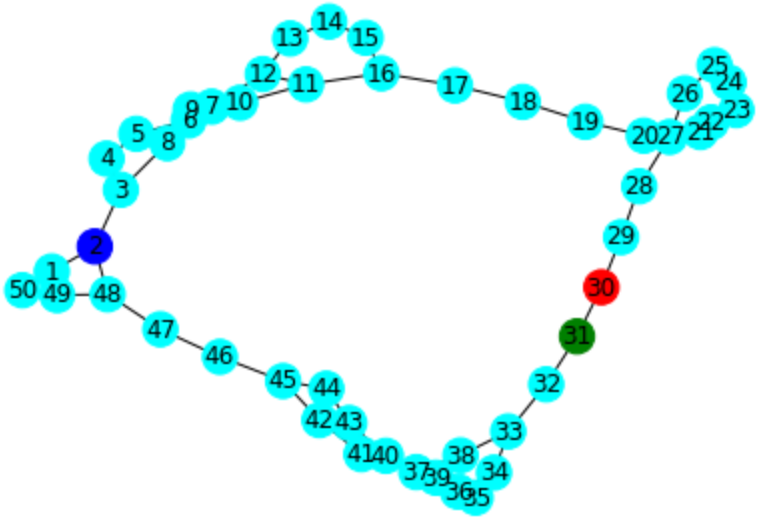
Yes, such starting states do exist.
This happened when the Agent and the predator were spawned in such a way that the agent and the predator have all common neighbors.
This means that whatever step the agent takes it is highly possible that the predator also moves to that node and catches the agent.

Suppose, the Agent is spawned on node 15 (with neighbors 14,16) and the predator is spawned on node 17 (with neighbors 18,16, 14).
Whatever choice of action the agent makes, the predator will have a 60% chance (because distracted) to move to the agent's node.

This was handled automatically after we performed experimentation on how to assign utilities and rewards.

After solving that, the only other possibilities that we observed, where the agent was unable to catch the prey, were states like (2,31,31),(2,29,31),(2,30,31). To try to find the reason , we had to visualize these states.





We observed that the prey and the pred were spawned in a set of nodes which were of degree 2. So it was less probable for them to scatter quickly. And for every case of this kind of failure, the agent was at 2. There must be very conflicting utilities with these states. These cases only occurred for no steps set as 100. With a higher number of steps, these cases turned into successes as well.

Therefore our success rate goes to 100% if the number of steps is increased.

Q. Find the state with the largest possible finite value of U* , and give a visualization of it.

The states with largest possible values of U* are the non favorable terminal states, that is when the Agent and the Predator occupy the same node. Few more states which will have a very large value of utility will be when the agent is one action away to get to a non favorable terminal state.

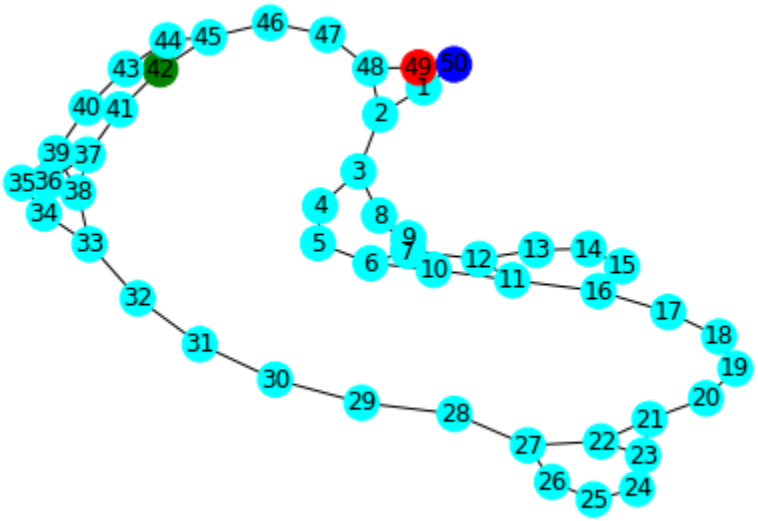
For our unfavorable terminal states, we assigned a very high finite utility. And for our favorable states we assigned zero utility. Assuming that unfavorable states are referred to as infinite. We'll find the states with the second highest utility value.

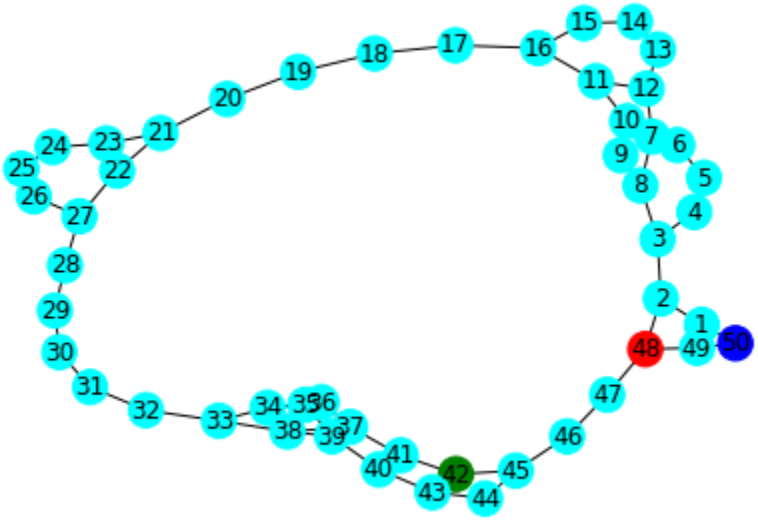
```
3645.0334977477155, (50, 42, 48): 8719.302887055846, (50, 42, 49): 10000, (50, 42, 50): 10000, (50, 43, 1): 10000, (50, 43, 2): 6676.8061191397355, (50, 43, 3): 2932.523537447231, (50,
3, 5): 62.12186528458624, (50, 43, 6): 61.52135433883444, (50, 43, 7): 61.74769370640803, (50, 43, 8): 1316.0557899060068, (50, 43, 9): 61.78390657004826, (50, 43, 10):
61.35074063556174, (50, 43, 12): 61.39261914610407, (50, 43, 13): 61.34213465383988, (50, 43, 14): 61.32958812034005, (50, 43, 15): 61.31832290497619, (50, 43, 16): 61.32631744241939,
```

Here, we can see how utilities propagate outwards from unfavorable states. According to our observation, the next worst utility value can be seen in the states where the agent is two steps away from the unfavorable state and is likely to go 1 step closer to

the terminal state. (we assigned utility to the terminal, and 1 step away from the terminal states).

Here's a visualization of the above:





Q. Simulate the performance of an agent based on U∗ , and compare its performance (in terms of steps to capture the prey) to Agent 1 and Agent 2 in Project 2. How do they compare?

We compared the performance of Agent U* with Agent 1 and Agent 2 in terms of the estimated minimum number of steps they take to successfully catch the prey. We've shown the average number of steps it takes for each agent to catch the prey.

Agent 1

Agent 2

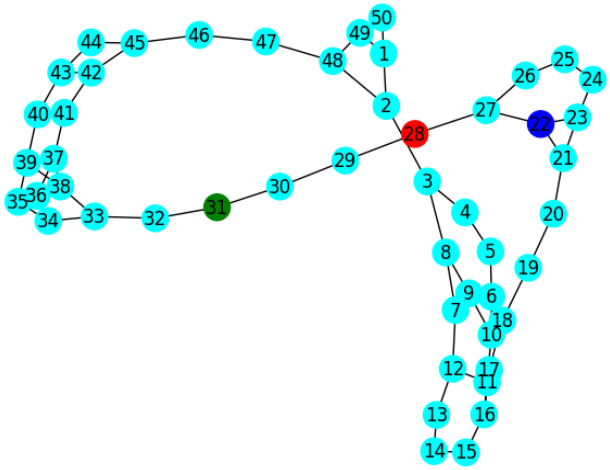
```
agent_two_remix(1,31,30,150)
[55]  ✓ 0.8s
...  0 : (1, 31, 30)
     1 : (2, 30, 31)
     2 : (48, 29, 32)
     3 : (47, 29, 33)
     4 : (46, 28, 34)
     5 : (45, 29, 33)
     6 : (44, 29, 34)
     7 : (45, 28, 36)
     8 : (44, 28, 37)
     9 : (44, 28, 41)
     10 : (44, 28, 40)
     11 : (45, 28, 41)
     12 : (44, 27, 40)
     13 : (45, 27, 43)
     14 : (46, 26, 44)
     15 : (47, 27, 43)
     16 : (48, 22, 40)
     17 : (2, 21, 41)
     18 : (3, 21, 40)
     19 : (8, 20, 43)
     20 : (9, 20, 44)
     21 : (10, 19, 43)
     22 : (11, 18, 40)
     23 : (16, 17, 43)

     'Success'
```

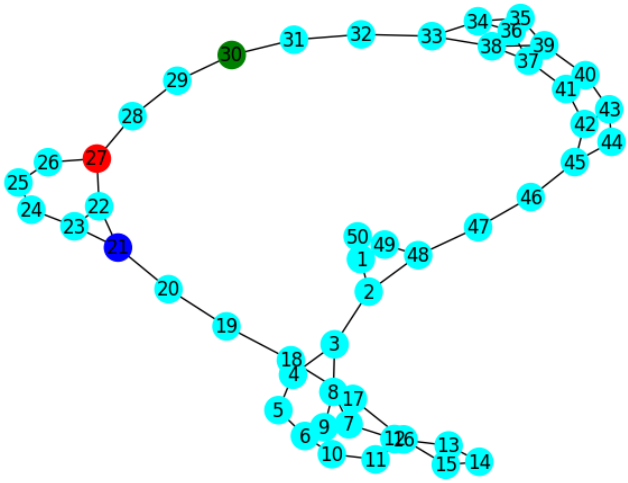
U* Agent	Agent 1	Agent 2
15	57	24

- We calculated the steps above for one of the worst starting states where both prey and predator were in the same direction from the agent.
  e.g. (Agent:1 Prey:31 Pred:30)

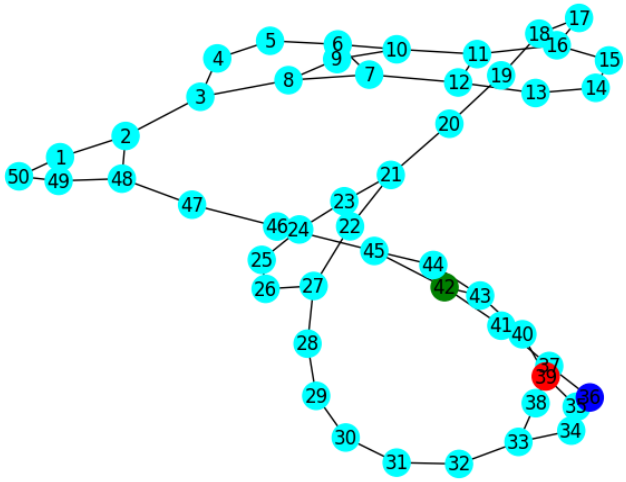
- These states were the ones where the U* Agent and Agent 1 made different choices. U* Agent already knew the best policy at any given state, while agent 1 didn't. Agent 1 moves according to the proximity from the prey and the predator. U* the agent took that best action even if, in some cases, it meant moving a bit away from the prey. Agent 1 on the other hand had conflicted reactions to the states and kept wiggling, resulting in rapid increase in the number of steps with no actual progress. This is where the efficiency gap was observed the most.



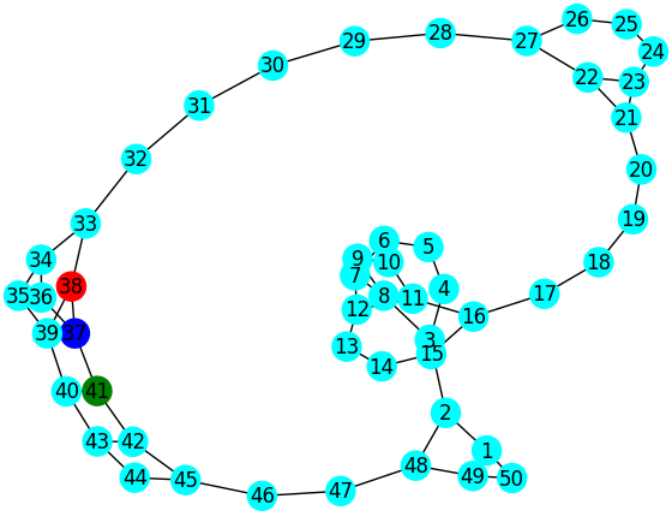
U* Agent - state 1 to state 2



- Blue denotes Agent, Red denotes the predator and Green denotes the prey. Here we can observe that the prey and the predator are in the same direction from the agent. The agent decides to take a step away even though the distance from the prey is increasing as well. This is because it knows what the best policy is of being in that state, i.e. the best action. Now let's take a look at what agent 1 does in a similar situation.

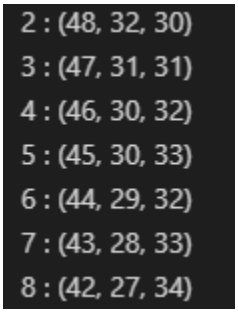


Agent 1 - state 1 to state 2

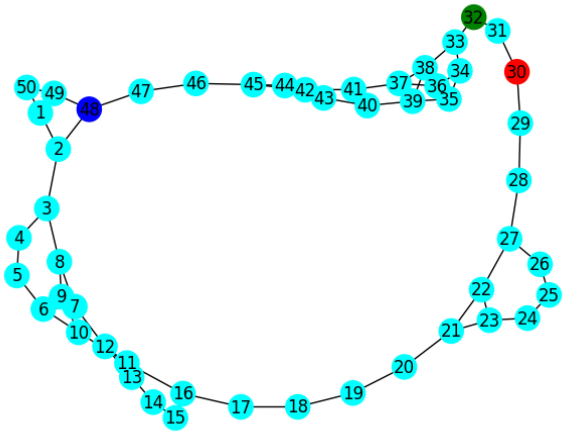


- Here, we can see that in situations like this, the actions of the agents are chosen at random and there is no best action assigned to each state. This leads to ambiguity while choosing actions causing an increase in the number of steps if at all successful.

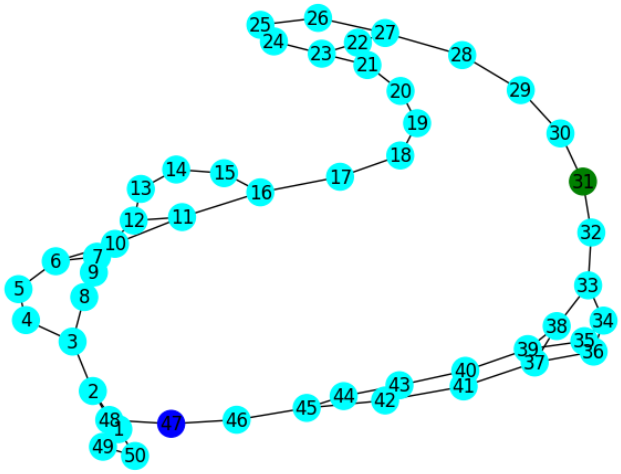
- **Agent 2** works similar to Agent 1 if the predator is close i.e within a radius of 5 nodes, otherwise it follows the prey.



Agent 2 moving rapidly toward the prey



Agent 2 From state 1 to state 2



- The agent was run several times, and average no. of steps were observed.

- For states that we consider as favorable states, the average number of steps were very low for U* agent(3-8) while for agent 1 and agent 2 we observed a relatively higher number of steps.

- We have visualized the movement of U* Agent and Agent 1 in cases where they make different choices. Since we had multiple graphs as paths of these agents, too many to accommodate in the report, we have recorded their video clips showing the difference in their respective movements.
- VIDEO EXAMPLE IN THE FOLDER

Build a model to predict the value of U ∗ (s) from the state s. Call this model V

Q) How do you represent the states as input for your model?

- For our neural network, our input structure is a 2-d array with dimensions [125000,3] where we have 125000 data points and 3 features for each data point namely, agent position, prey position and predator position.

Q) What kind of features might be relevant?

- Apart from agent position,prey position,predator position, other features that we were experimenting with were distance of prey from agent and distance of predator from the agent.

Q) What kind of model are you taking V to be? How do you train it?

- We're taking V to be a neural network. After a lot of experimentation we perfected the specifications of our model.
- We decided to use 2 hidden layers with activation function Leaky Relu and the output layer with the linear activation function.
- We also experimented with tanh and sigmoid functions for hidden layers but they did not stack up well. Sigmoid function is generally used for classification problems but we had a regression problem because utility values are continuous real values.
- Apart from this, we're using 35 nodes in each of our hidden layers and 1 node in our output layer. To train our model we assign weights with mean 0 and small variance and our initial bias as zero.
- We took out error function to be the cross entropy function
- We did batch gradient descent to make our weights optimal, randomly batches of 500 were chosen and gradient descent was applied to those many inputs rather than on all the inputs.

Q) Is overfitting an issue here?

- Yes, overfitting was an issue here.

Q)How accurate is V ?

- The accuracy of the neural network was about 50 % but it was able to predict the utility values in the same range (meaning that in order if say utility for A was less than utility of B then the model was not able to predict accurately the values of utility but in the predicted utility also the utility of A was less than utility of B, hence our agent was able to perform

- Since we were not able to minimize our loss completely but were to get utility values on which our V agent is able to perform, hence the success rate of our agent V came out to be 90%

Q) Once you have a model V , simulate an agent based on the values V instead of the values U* . How does its performance stack against the U* agent?

- Since we were not able to minimize our loss completely but were to get utility values on which our V agent is able to perform, hence the success rate of our agent V came out to be 90%
- Whereas U* agent was working on optimal utilities hence it's success rate (100%) was much higher than V agent.

<u>U Partial Agent</u>

Upartial(Sagent, Spredator, p) = Summation(on Sprey){ **p**Sprey U^* (Sagent, Spredator,Sprey)}.

Each state represents a tuple of Agent position, predator position and a vector of probabilities depicting where the prey might be at that time step. The probability vector is updated at every time step according to the information received by surveying a node or moving to a particular node. The probability updates are done according to guidelines mentioned in project 2.
The above formula helps us to estimate the partial utility of a state so that when our agent moves it can take the best possible action without actually knowing the prey position.
U* was the optimal utility we got after running the value iteration algorithm on every state. Since U* was optimal and we are multiplying the

Simulate an agent based on Upartial, in the partial prey info environment case from Project 2, using the values of $U*$ from above.

The U partial agent takes all possible actions into consideration and then by fixing every action and the predator position, U partial is calculated for that state given the current p vector containing probabilities of prey being at all the nodes.

The agent then takes the action from which it obtains minimum partial utility.
The above two processes go on till either the agent catches the prey or the predator catches the agent.

The U partial agent performs exceedingly well with a success rate of about 99.8% for 1000 different starting positions on a fixed graph.

Q)How does it compare to Agent 3 and 4 from Project 2?

- In terms of success rate, Agent U* beat agent 3 by a margin of around 20% and beat agent 4 by a margin of around 10%.
- In terms of the average expected number of steps, the agent UPartial outperformed both agent 3 and 4 in the partial environment setting where the prey is unknown.
- Given below are snippets of the average number of iterations it took each agent to reach success.
-

Agent 3	**Agent 4**	**Agent UPartial**
60	31	24

Agent 3 :

```
agent_three(1,31,30,100)
```

[133] ✓ 0.1s

```
Output exceeds the size limit. Open the full output data in a text editor
0 : (1, 31, 30)
1 : (50, 32, 29)
2 : (1, 32, 30)
3 : (50, 31, 31)
4 : (1, 30, 32)
5 : (50, 31, 33)
6 : (1, 31, 32)
7 : (50, 32, 33)
8 : (1, 33, 38)
9 : (50, 32, 33)
10 : (1, 33, 38)
11 : (50, 32, 37)
12 : (1, 31, 41)
13 : (50, 31, 42)
14 : (1, 30, 45)
15 : (50, 30, 42)
16 : (1, 30, 41)
17 : (50, 31, 40)
18 : (1, 32, 41)
19 : (50, 33, 40)
20 : (1, 38, 41)
21 : (50, 39, 42)
22 : (1, 39, 43)
23 : (50, 35, 42)
24 : (1, 35, 41)
...
56 : (37, 30, 49)
57 : (38, 29, 48)
58 : (33, 29, 49)
59 : (32, 30, 48)

'Success'
```

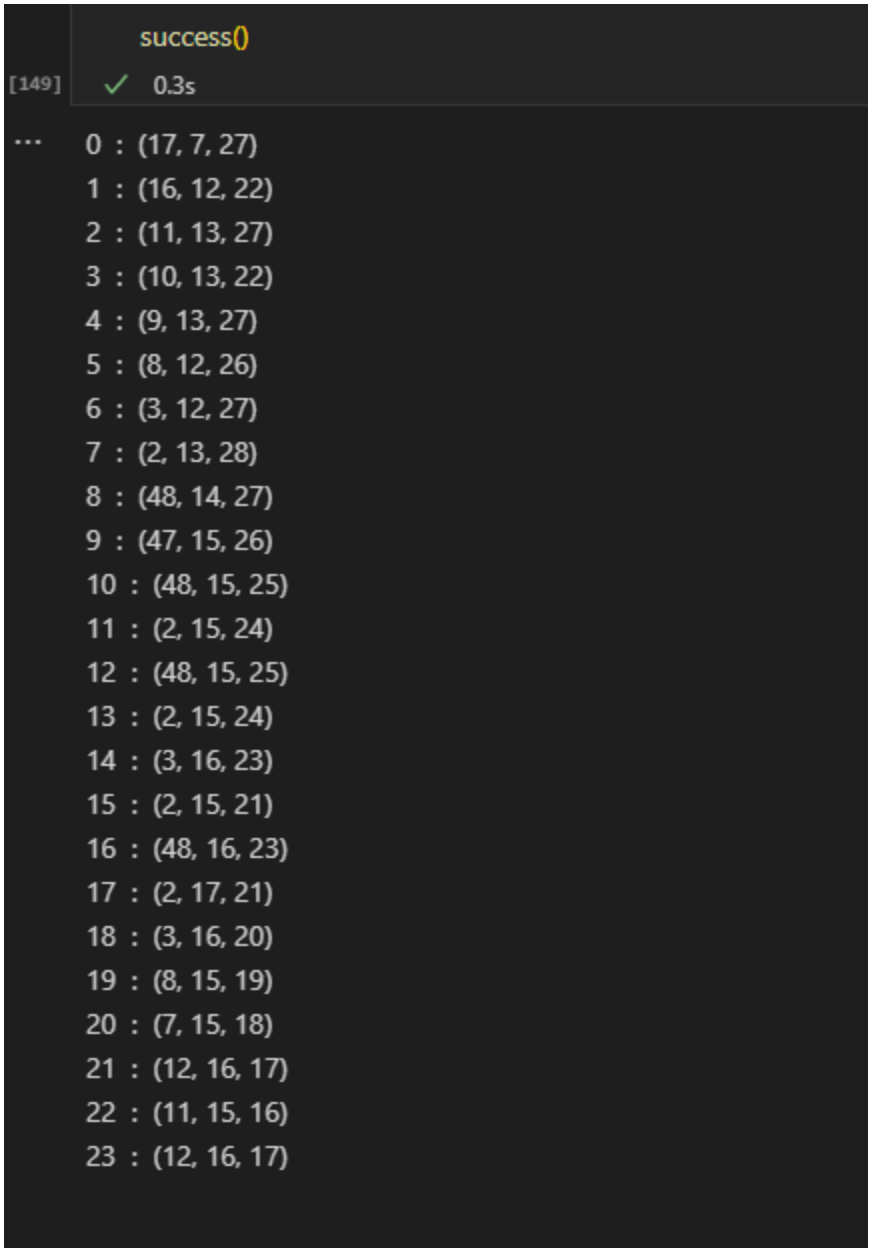
Agent 4:

```
agent_four(1,31,30,100)
✓ 0.9s

Output exceeds the size limit. Open the full output data in a text editor
0 : (1, 31, 30)
1 : (2, 32, 29)
2 : (3, 32, 30)
3 : (2, 33, 29)
4 : (3, 32, 28)
5 : (8, 31, 27)
6 : (7, 32, 28)
7 : (12, 32, 27)
8 : (11, 32, 28)
9 : (12, 33, 27)
10 : (7, 38, 28)
11 : (12, 38, 27)
12 : (11, 39, 28)
13 : (16, 39, 27)
14 : (17, 40, 28)
15 : (18, 43, 27)
16 : (17, 42, 28)
17 : (16, 41, 27)
18 : (11, 42, 22)
19 : (12, 42, 21)
20 : (7, 45, 20)
21 : (8, 44, 19)
22 : (3, 43, 18)
23 : (2, 40, 17)
24 : (48, 41, 16)
...
27 : (47, 42, 17)
28 : (46, 43, 16)
29 : (45, 40, 17)
30 : (42, 40, 16)

'Success'
```

Agent UPartial:



Q) Do you think this partial information agent is optimal?

- When we say whether the partial information agent is optimal or not, we need to make sure what our point of reference is. Since we are looking at the expected minimum number of steps to catch the prey, we can say that no, Partial U* agent is not optimal.

- But we observe that it gives a 100 percent success rate. Rarely (1 out of 1000 times) there is a case or two where the agent, predator and prey are spawned in a location where it is impossible for the agent to escape the predator which

returns a success rate of 99.7 to 99.9. Other than this the agent is always successful.

- Here, at every step the agent calculates the partial utilities of its neighboring states and sums them up over the possible actions. It then takes action for which the partial U is minimum.

- It goes on to do so until it finds the prey.

Build a model Vpartial to predict the value Upartial for these partial information states. Use as the training data states (including belief states) that actually occur during simulations of the Upartial agent.

Q) How do you represent the states sagent, spredator, p as input for your model? What kind of features might be relevant?

- We're taking an input of dimensions 125000 X 51 in which the first feature for every datapoint is the distance of agent to predator and the rest 50 features are shortest distances of the agent to every other node because we're unsure where the prey is and there is some probability of prey being on every node.
- We are calculating the distances using the bfs function as you can see in the code.

Q) What kind of model are you taking Vpartial to be? How do you train it?

- Our VPartial is very similar to our V model. The main difference is how we take inputs for our model. Here we consider the shortest distances of the agent from all probable positions of the prey in an ordinal manner along with the distance between agent and the predator.
- We decided to use 2 hidden layers with activation function Leaky Relu and the output layer with the linear activation function.
- We also experimented with tanh and sigmoid functions for hidden layers but they did not stack up well. Sigmoid function is generally used for classification problems but we had a regression problem because utility values are continuous real values.
- Apart from this, we're using 35 nodes in each of our hidden layers and 1 node in our output layer. To train our model we assign weights with mean 0 and small variance and our initial bias as zero.
- We took out error function to be the cross entropy function
- We did batch gradient descent to make our weights optimal, randomly batches of 500 were chosen and gradient descent was applied to those many inputs rather than on all the inputs.

Q) Is overfitting an issue here? What can you do about it?

- Yes, overfitting is coming out to be an issue here. We can use regularization to deal with overfitting here. Alternatively dropout layers can be added in between every hidden layer so that some random features are dropped in each forward pass which will eventually prevent overfitting of our data.

Q) How accurate is Vpartial? How can you judge this?

- The model accuracy here, just as the V model wasn't very good but it was ordinal which gave us a high success rate for the agent irrespective of a huge loss. To judge the accuracy you can compare the true values to predicted values.

Q) Is Vpartial more or less accurate than simply substituting V into equation (1)?

- Vpartial is less accurate than simply because our loss is on the higher side and the predictions are less accurate compared to the V partial substituted.

Q) Simulate an agent based on Vpartial. How does it stack against the Upartial agent?

- Our Partial agent gave a success rate of 96% approx. It was close to UPartial but not as accurate as the Upartial agent. We used the same strategy for both the agents but the Estimated Partial Agent (100%) worked better than the predicted V partial Agent.

<u>Bonus:</u>

Bonus 1: Build a model to predict the actual utility of the Upartial agent. Where are you getting your data, and what kind of model are you using? How accurate is your model, and how does its predictions compare to the estimated Upartial values?

- To predict the actual utility of the Upartial agent, we could build a neural network where the input values are the distance of agent from the predator, average of utility of all 50 states corresponding to that agent and pred position. AFter some experimentation, we were able to declare these features as the most efficient for training this model. Our output was the

Bonus 2: Build an optimal partial information agent.

- To build an optimal partial information agent, we will survey nodes within a radius of nodes at every time step