S2173079 - Ahmed Ibrahim Adem Hamed

AP1 Viva

Genetic-Optimized Deep Reinforcement Learning AI for Playing Single-Player Games

Problem Statement

The Learning Environment

In the advancing age of AI, the next step that humanity is hyped for is robotics. But the task of re-engineering human evolution into these machines is more underestimated than it should be. While the internet has proved to be an excellent learning environment for LLMs, the same cannot be true for robots that have to interact in a physical environment space. Specialized environments are under development, but there is a plethora of pre-existing environments that we can repurpose for this task. Videogames!

Objectives

1. To utilize reinforcement learning to achieve human level performance by an agent in simple single player games with discrete states

2. To use deep reinforcement learning to achieve the same performance but in more complex multi-agent games

Literature Review

1. The Arcade Learning Environment: An Evaluation Platform for General Agents

Bellemare, M. G., Naddaf, Y., Veness, J., & Bowling, M. (2013). The Arcade Learning Environment: An Evaluation Platform for General Agents. Journal of Artificial Intelligence Research, 47, 253–279.

3. Playing Atari with Deep Reinforcement Learning

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing Atari with Deep Reinforcement Learning. arXiv preprint arXiv:1312.5602.

2. Deep Reinforcement Learning for Flappy Bird

Chen, K. (2015). Deep Reinforcement Learning for Flappy Bird. Retrieved from https://cs229.stanford.edu/proj2015/362_report.pdf

4. Gymnasium: A standard Interface for Reinforcement Learning Environments

Towers, M., Kwiatkowski, A., Terry, J., Balis, J. U., De Cola, G., Deleu, T., Goulão, M., Kallinteris, A., Krimmel, M., KG, A., Perez-Vicente, R., Pierré, A., Schulhoff, S., Tai, J. J., Tan, H., & Younis, O. G. (2024). Gymnasium: A Standard Interface for Reinforcement Learning Environments.

Literature Review (cont.)

i. Video PreTraining (VPT): Learning to Act by Watching Unlabeled Online Videos

Baker, B., Akkaya, I., Zhokhov, P., Huizinga, J., Tang, J., Ecoffet, A., Houghton, B., Sampedro, R., & Clune, J. (2022). Video PreTraining (VPT): Learning to Act by Watching Unlabeled Online Videos. arXiv preprint arXiv:2206.11795.

ii. MineRL: A Large-Scale Dataset of Minecraft Demonstrations

Guss, W. H., Houghton, B., Topin, N., Wang, P., Codel, C., Veloso, M., & Salakhutdinov, R. (2019). MineRL: A Large-Scale Dataset of Minecraft Demonstrations. arXiv preprint arXiv:1907.13440.

iii. Fighting Zombies in Minecraft With Deep Reinforcement Learnin

Udagawa, H., Lee, S.-Y., & Narasimhan, T. (2016). Fighting Zombies in Minecraft with Deep Reinforcement Learning. Retrieved from https://cs229.stanford.edu/proj2016/report/UdagawaLeeNarasimhan-FightingZombiesInMinecraftWithDeepReinforcementLearning-report.pdf

Project Requirements

Fully functional environment

The environment is fully functional and can be rendered to the user either during testing or during demonstration

Flatlined rewards

After a certain number of episodes, the change in the reward value flatlines. This means that the agent has found the most optimal solution

Demonstrate optimal solution in every run

For every episode after the training has been complete, the agent picks the path with the least number of steps.

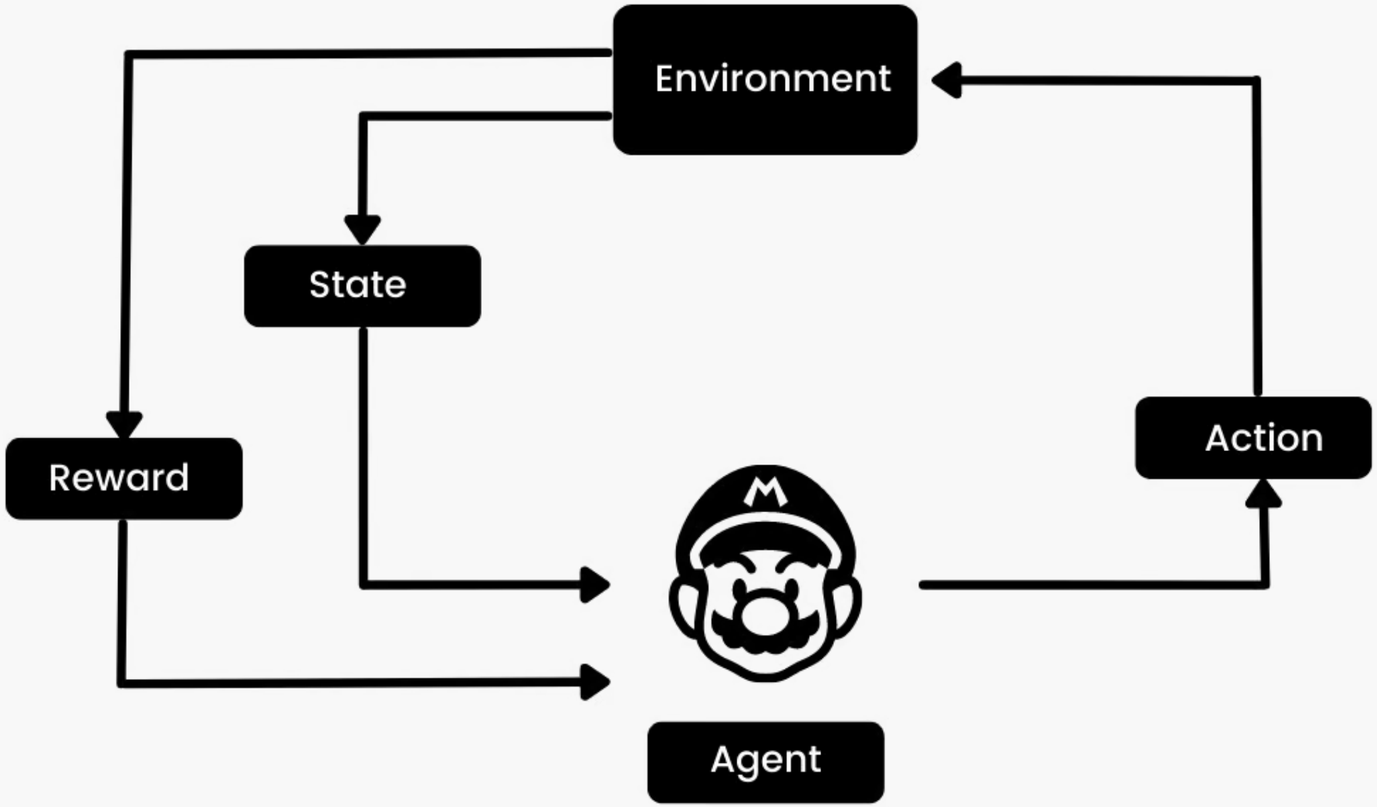
Methodology

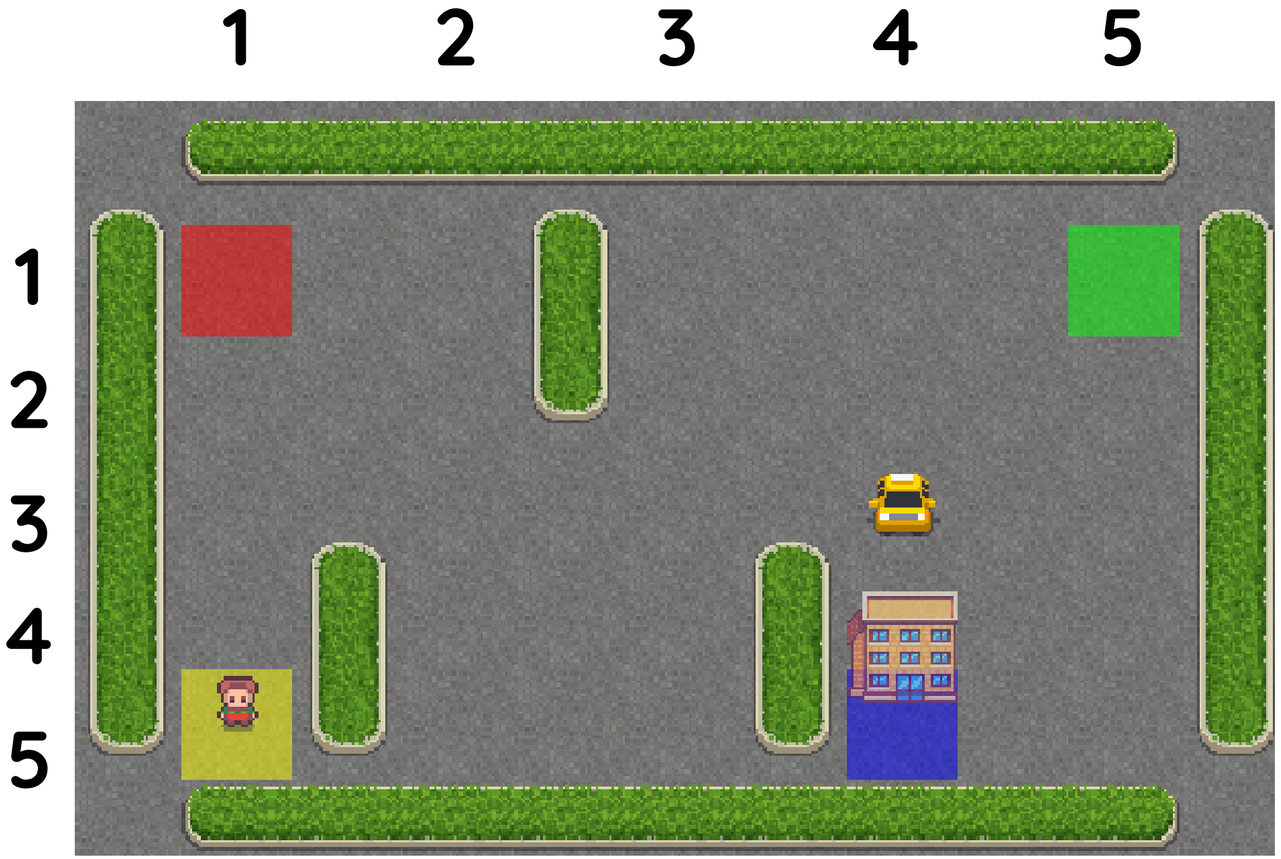
Q-Learning

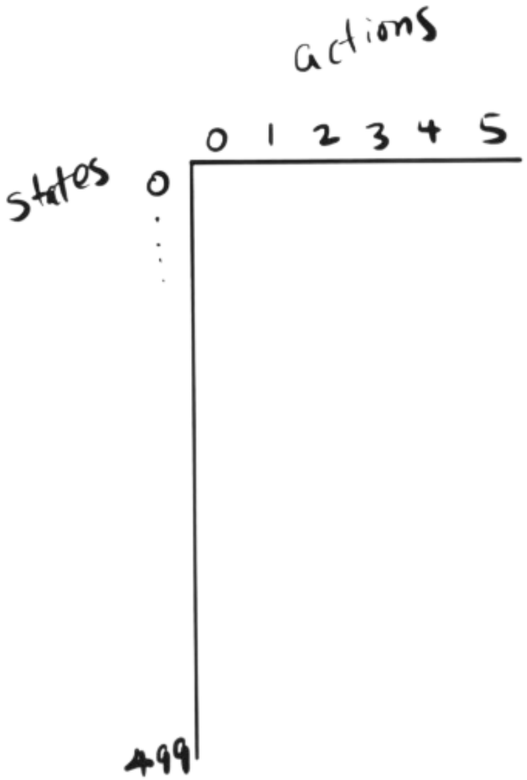
Q-Learning is a model-free, off-policy reinforcement learning (RL) algorithm used to train agents to make optimal decisions in Markov Decision Processes (MDPs). It is based on Q-values (action-value function), which estimate the expected future rewards for taking an action in a given state.

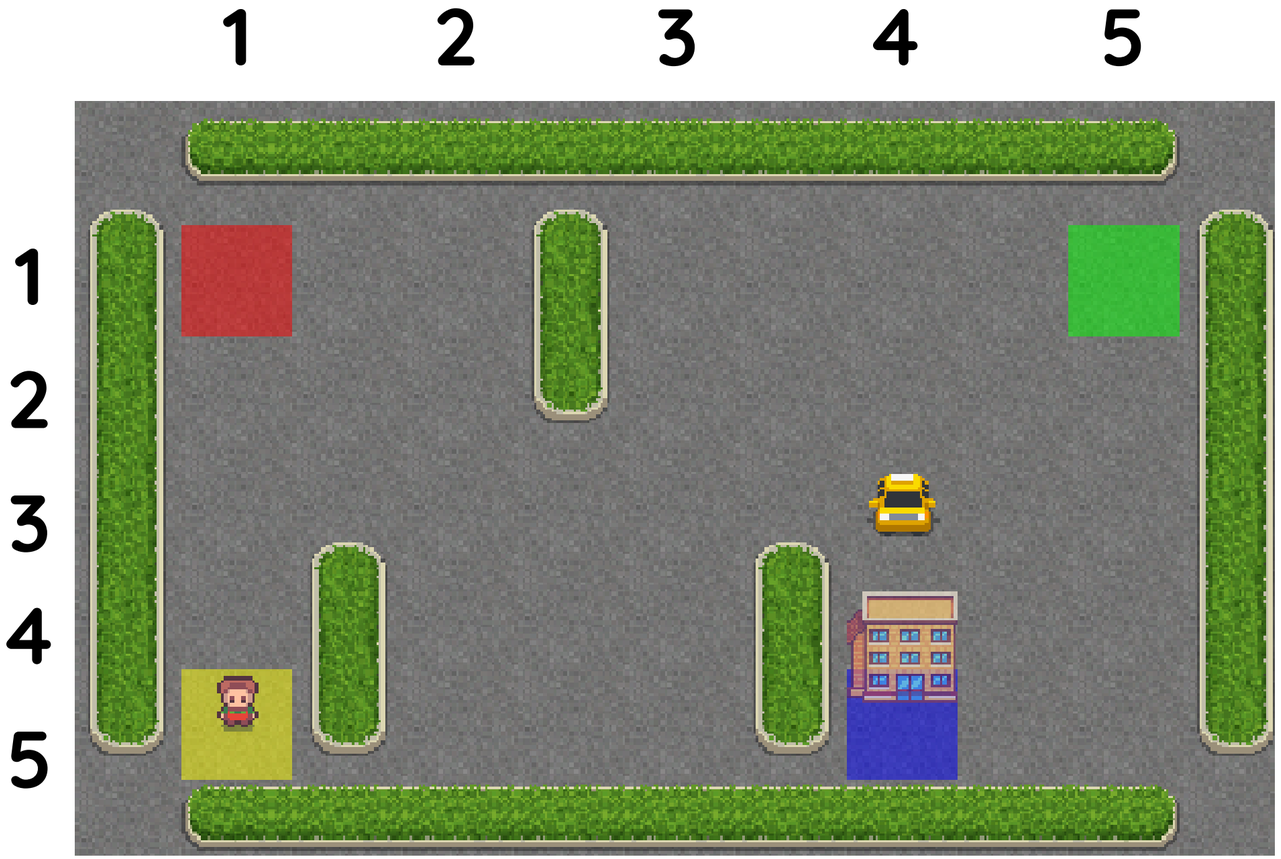
- State (S): Represents the current situation of the agent in the environment.
- Action (A): Choices available to the agent at any given state.
- Reward (R): A scalar value received after taking an action. It guides the agent to learn which actions are beneficial.
- Q-Value (Q(s, a)): Represents the expected cumulative future reward of taking action a in state s and then following an optimal policy. It is stored in a Q-table for lookup during training.

System Design

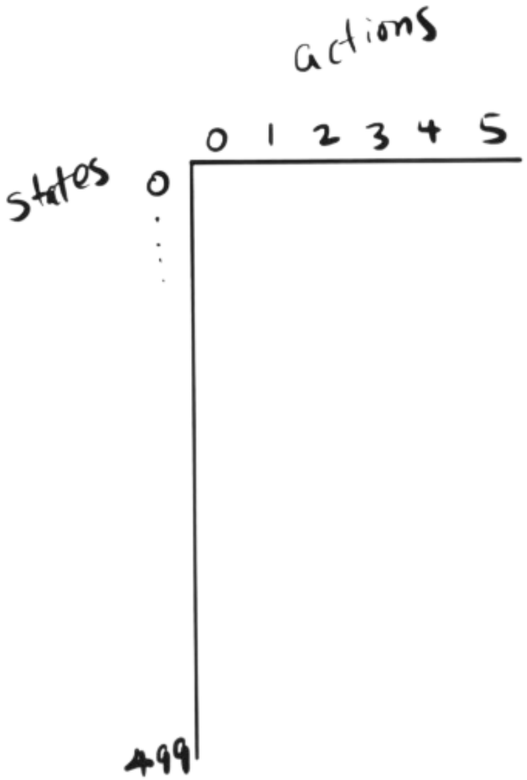


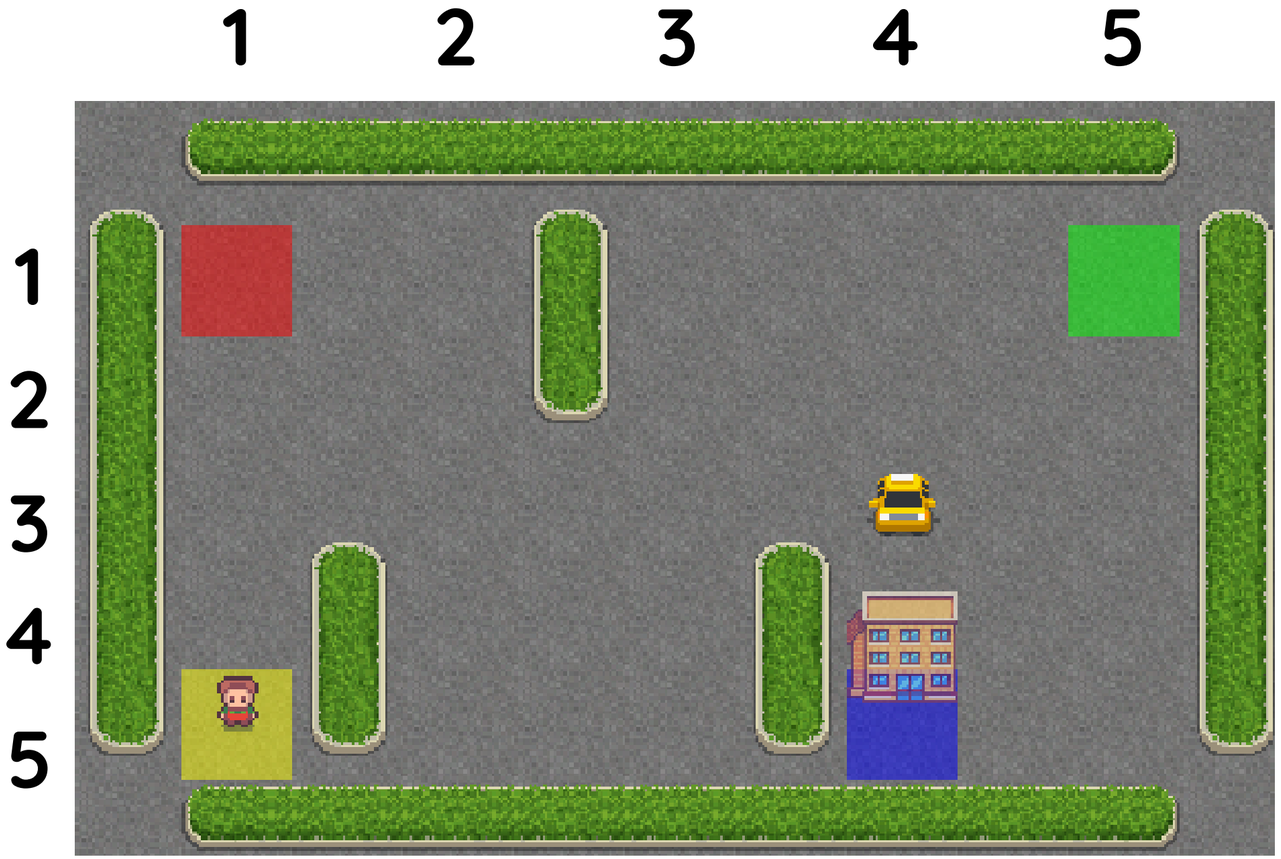

1
2
3
4
5
1
2
3
4
5


actions
0 1 2 3 4 5
states
0
499



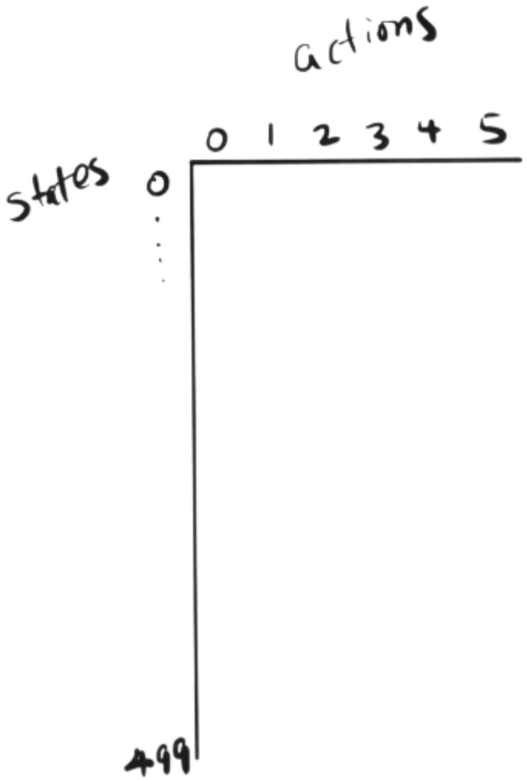
Taxi = 5 x 5 = 25

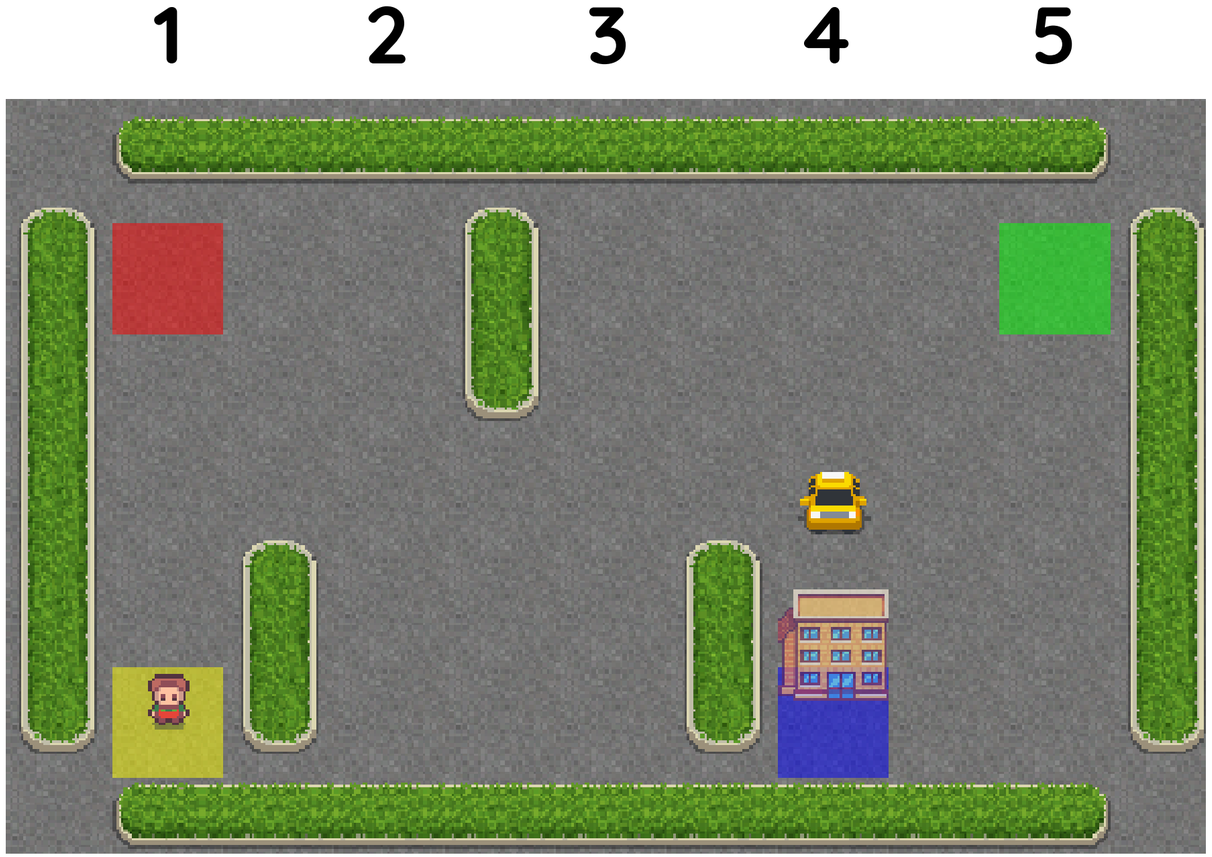




Taxi = 5 x 5 = 25

player = 5

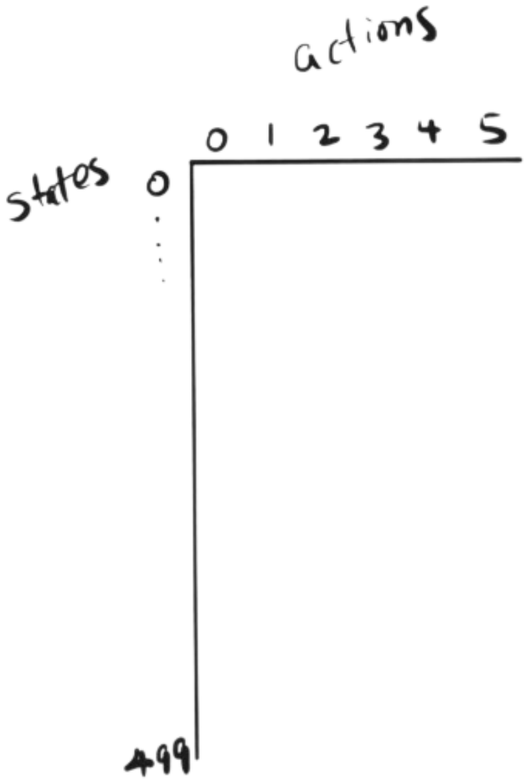


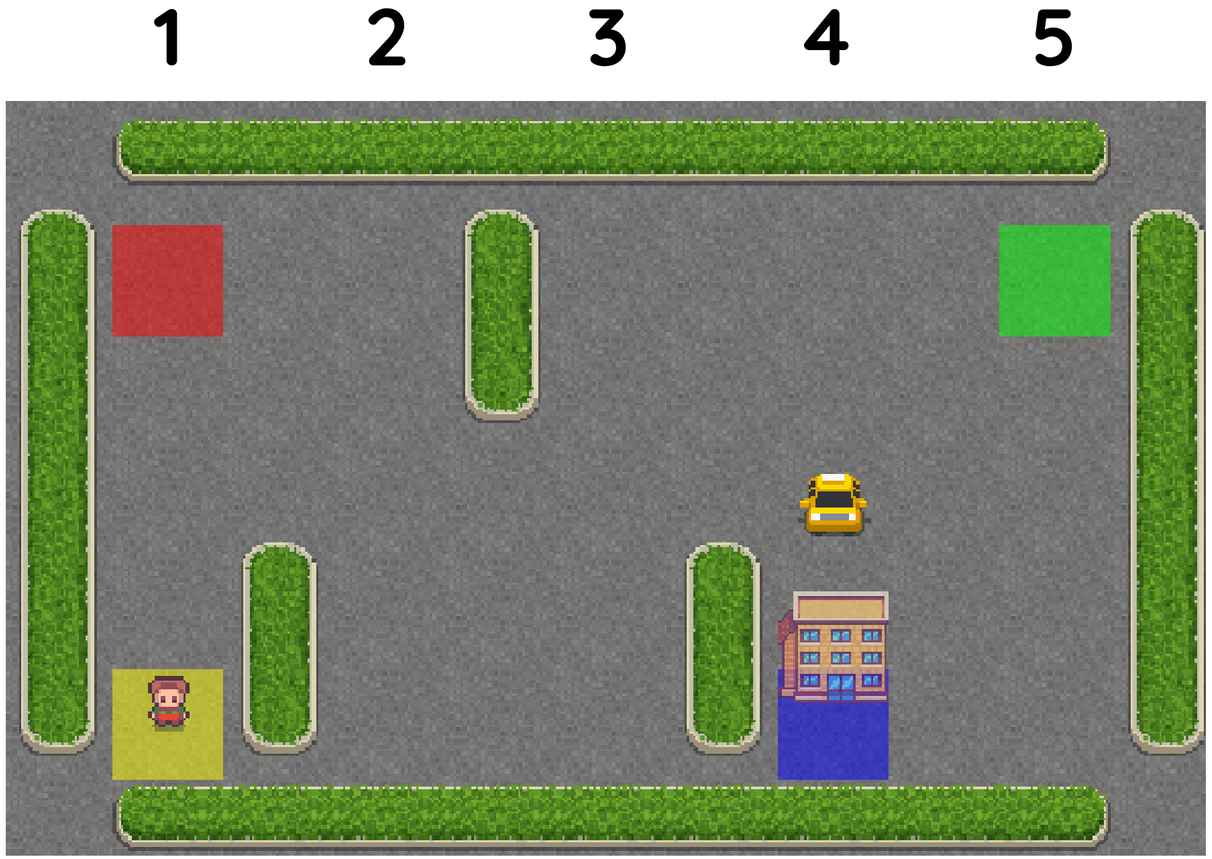


Taxi = 5 x 5 = 25

player = 5

destination = 4



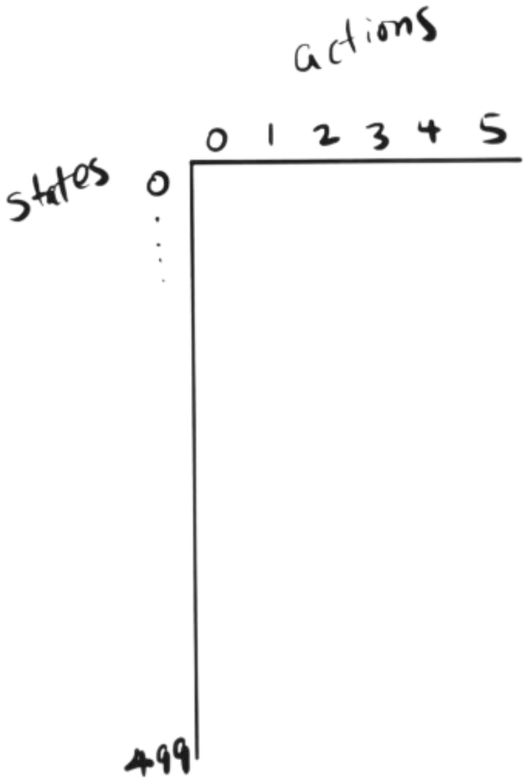


Taxi = 5 x 5 = 25

player = 5

destination = 4

Total states = 500



Thank you