Assignment 2

Trivial Pursuit

Deadline: July 15, 11:55pm.
Perfect score: 100.

Assignment Instructions:

Teams: Assignments should be completed by teams of up to three students. You can work on this assignment individually if you prefer. No additional credit will be given for students that complete an assignment individually. Students are responsible for staying engaged with their team in a timely and effective manner, and will face penalties if they fail to do so. Students may change teams between projects. All team members are responsible for their team's submissions and adherence to academic integrity policy.

Make sure to write the name and RUID of every member of your group on your submitted report.

Submission Rules: Projects should be submitted to Canvas as a zip file. The zip file should include all code written for the assignment, as well as your project report in PDF form. Do not submit your report as a Word document or raw text. Each team of students should submit only a single copy of their solutions and indicate all team members on their submission. Failure to follow these rules will result in lower grade for this assignment.

TA Meetings: Each team will sign up for an appointment via the course Canvas site and videoconference with a course TA before submitting their project. Each team will be allotted 10 minutes, with the meeting counting for 10% of the project grade. The TA will verify the team's progress and approach to the assigned tasks, and direct students to resources, as needed. As stated above, students should have completed the early steps of the project before their progress report meeting. Some teams may be asked to meet again post-project to discuss their submission, at the discretion of the TAs. It is essential that you reply promptly to TA inquiries and adhere to your team's designated meeting times.

Please write who you had your TA meeting with in your final report.

Precision: Try to be precise, especially when writing proofs. Computer science is at its heart a branch of mathematics, and I expect the same level of rigor from your proofs that I would in an upper-level math class. Each step in a valid should be an application of a known mathematical identity. Handwaving, direct assertion, or arguing that something is true because it would make sense, is not a valid proof, and will be graded as a zero. For non-proof questions, write as if you are trying to convince a very skeptical reader.

Collusion, Plagiarism, etc.: Each team must prepare its solutions independently from other teams, i.e., without using common notes, code or worksheets with other students or trying to solve problems in collaboration with other teams. You must indicate any external sources you have used in the preparation of your solution. **Do not plagiarize online sources** and in general make sure you do not violate Rutgers' academic integrity policy, which can be found here: `https://global.rutgers.edu/academic-integrity-rutgers`. Failure to follow these rules may result in failure in the course.

Some cases of academic misconduct occur because the student gets over-rushed and desperate. Other times, the student is simply at a loss for how to proceed. To avoid this situation, always start with the actual project guidance and class session content in completing assignments – these resources are provided for a reason. Second, make sure that you stick with the recommended timeline for your work, and reach out to me and the TAs when you are unsure about how to proceed. Assigned projects are to ensure that you are following the provided material and doing the required work for the course, and, therefore, closely parallel the course content.

Assignment Description

The purpose of this project is to give you experience modeling the knowledge you have about a problem, how that knowledge changes over time, and using that knowledge to inform how you make decisions and act in an environment.In this project you'll build an environment, an agent that has to interact with that environment in order to accomplish a goal, and you'll collect data to analyze the performance of your agent in pursuit of that goal.

1 Implementation

1.1 The environment

The environment for this project is a graph of nodes. Some of these nodes will be connected, allowing transit between them. Movement between unconnected nodes is impossible.

To generate the environment, generate a graph with 40 nodes (consider them numbered 1 to 40). First, connect them in a large loop, (node 1 connects to node 2, 2 to 3, etc, 39 to 40, node 40 to node 1). Then add 10 more edges between randomly selected nodes, making sure a) an edge doesn't already exist between those two nodes, and b) no node has degree more than 3.

A) How can you do this efficiently?

1.2 The Target

Somewhere in this environment is a target that you want to capture. The target occupies a node, and each timestep the target moves to one of the neighbors of its current node, executing a random walk through the environment. The target always moves, and cannot stay in the same place.

1.3 The Agents

For this project, you'll build a number of agents that are attempting to catch the target. Each agent has different capabilities, in terms of what they can sense and what they can do. The agents and the target alternate taking actions. Build the agents, collect data, and analyze their performance.

Total Knowledge, Limited Movement These agents have knowledge of the whole environment (in particular,where the target is at every timestep). But the agent starts at a random point, and can move at most one step each timestep in pursuit of the target. The target is captured if the agent ends its round in the target's node

- **Agent 0** sits at a fixed node and does not move. If the target enters the agent's cell, the target is captured

  B) Give a rough argument why (with probability 1), this lazy agent will still catch the target.

- **Agent 1** always selects a move to reduce the distance to the target as fast as possible, using an appropriate search algorithm. If there are multiple such moves available, Agent 1 chooses one of them at random.
- **Agent 2** is your own design. Try to beat Agent 1. You can only move on edge at a time

Partial Knowledge, Free Movement These agents do not know where the target is, but do know how the target moves. On their turn, these agents can examine any node in the environment to check to see if the target is there. They maintain a belief state (probabilistic knowledge base) of where the target is, updating it based on a) the result of examining a node, and b) knowing how the target moves. The target is captured if the agent examines the target's node.

- **Agent 3** examines the same node every time, waiting for the target to show up there.

  C) Note, the results you get for this agent should not surprise you. Why?

- **Agent 4** always examines the node with the highest probability of containing the target. If there are multiple such nodes, Agent 4 chooses from them at random.
- **Agent 5** is your own design. Try to beat Agent 4.

Partial Knowledge, Limited Movement These agents do not know where the target is, and can only move between adjacent nodes each timestep. However, they have the ability to examine any node in the system to see if the target is there. In its turn, the agent can choose to examine any node to see if the target is there, update its belief state based on that information, and then choose to move to an adjacent node. The target is captured if the agent enters the target's node.

- **Agent 6** always examines the node with the highest probability of containing the target, updates its belief state based on the result, and then moves to reduce the distance between its current position and the new node with highest probability of containing the target.

  > D) Does this agent ever successfully capture the target?

- **Agent 7** is your own design. Try to beat Agent 6.

2 Analysis

Along with the above implementation, you need to analyze the performance and the results of your code in a final **lab report**. The report should include the following:

- Discussion of design and algorithm choices you made, including but not limited to
  - Answers to the gray boxed questions above.
  - How you modeled and updated the belief state for the partial information agents. Make sure your probabilistic updating is correct!
  - Description of your Agents 2, 5, 7, and how you implemented it.
- Data comparing the performance of each agent to its peers. Do the results make sense - why or why not?
  - For this, you'll need to repeatedly run each agent (on different random environments) and average howmany steps the agent took to capture the target.

    > E) How should you handle situations where the agent doesn't seem to be catching the target?

  - Note that the more trials you run, the more accurate your data will be.
- Do your results make sense and agree with your intuition?
- How did your Agents 2, 5, 7 stack up against the others? Why did it succeed, or why did it fail?
- If you had more time and resources, what's the most advanced Agent 7 you could imagine? Would that strategy be provably optimal?