CMSC 671 (AI), Fall 2018
Group Project Description

*Please read this **carefully** (even though it is long) and start thinking about the steps involved in designing and implementing your player. The project requirements are subject to changes and improvements (but hopefully not too many). Good luck, and let the science begin!*

The AI Explorer

For the group project, you will be designing an agent that explores a grid-based world. The agent will find treasure, fight monsters, level up, and eventually find the exit—or die trying.[1] The first phase of the project will be single-player; in Phase II you will encounter a new kind of monster that may, in fact, be another group's agent.

Project Requirements and Grading

There are four graded components of the project: a project design (20%), your implemented system in two phases (30% each), and a final writeup (20%). You will be graded on the thoughtfulness and clarity of your design and presentation, and not primarily on your algorithm's performance. This gives you the freedom to try a risky approach that is interesting from a design perspective but might not work very well. An approach that doesn't work very well, and is *also* naïve, trivial, or not well-motivated, will not receive a good grade.

Don't just hack something together—this is an AI class! You should think about the strategies you want to implement; what AI methods would be appropriate for such strategies; and how those AI methods can be adapted for this purpose.

Project Design (20%)

You must submit a project design via Blackboard. Only one project design should be submitted per group, and all members should have input into the project design. (Remember, submitting something with someone's name on it when they did not in fact work on it is a form of academic dishonesty—and besides, you're going to be working on it; you should want input!) The initial design should be 2-4 pages long, not counting references.

Your project design should contain:

1. The team name and names of all participating team members.
2. A short (≤ 500 words) description of your exploration strategy, written in clear, understandable English.
    - How will you make decisions about where to go while exploring the map?

[1] The AI Dungeon is a member of a class of games referred to as "roguelikes." This example was inspired by both old-school examples (nethack) and modern games, most notably Dream Quest and Strategery.

    - What factors will you take into account when choosing whether to pick up a resource?
    - How will your strategy change with the introduction of dynamic monsters?
    - How will you decide whether to fight or not, and which monsters?
3. A discussion of how you developed the strategies (by playing other roguelikes, by reading articles, by discussing approaches with your team members, by trying things out and experimenting, …).
4. How your planned system draws on ideas from the AI literature.
    - This should be focused on material and concepts that we covered in class.
    - Cite references for the AI concepts you mention (when citing the textbook; include section numbers).
    - If you like, you may also discuss methods that you would/could use but don't expect to implement within the scope of the semester.
5. Your evaluation strategy. How will you test your system to see if it's working? How will you quantify how well it works? The results of your evaluation will be part of the final writeup.
    - Examples of possible evaluation strategies: Have your player play a large number of games; compare play against a human player's decisions; evaluate a set number of games and list problems noted; play against human(s); work with another team to run your players against each other. You should come up with an evaluation plan that *demonstrably* covers a lot of the search space (many cases).
6. A corresponding Python design describing:
    - Main functions (including required functions). (For example, do you have a separate function for calculating path costs? For making a decision whether to move? For deciding whether to interact with a map resource?)
    - Expected inputs and outputs for each function.
    - Behavior of each function.

It is fine (and likely) to make changes to the design after you submit this document.

Game Play

The core idea: your agent will be exploring a partially observable grid world (as before). In addition to terrain types that cost energy to move into, there will be objects on the map that help you (power-ups), and objects that try to hinder you (monsters).

The Big Ideas

There will be a few major changes from the exploration task you have been doing so far:

- **Turn-based decision making:** In all previous work, we have called a single main function named solve. In this work, you will provide a function named step, and we will call it every time your agent makes a move. It will take the entire state space as arguments, and return the next step your player will take ('N', 'S', 'E', or 'W'.)
- **Agent resources:** Your player will start with some amount of strength. Strength will be depleted by moving through rough terrain, used to determine who wins a fight with a monster, and replenished by power-ups.

- **Map objects:** In addition to terrain, your function will take a dictionary containing objects found on the map. These will be: monsters, who have strength of their own and who your agent will sometimes fight; and power-ups, which replenish your strength.

The State Space

The following things will be elements of the state space that is given to the agent. All of these will be maintained by the driver and passed in during each timestep. You may, but do not have to, maintain an internal representation of some parts of the state for your own use.

- The map: a matrix of the terrain, including unknown areas.
- Objects: a dictionary of what's on the map, indexed by location.
- The Boss: a special object; the goal state.
- The agent's current location and strength.

Map Movement

The map will have the same types of terrain as you have been using: paths, sand, mountains, and walls. In addition, we will introduce a square labeled u, for "unknown." In addition to the matrix describing what you know about the map itself, you will have a dictionary of map objects, keyed by location; these are objects on the map.

Each move you make will detract from your strength; moving into more rugged terrain will take more strength. You will lose 1 strength for moving into a path square, 3 strength for moving into a sand square, and 10 strength for moving into a mountain square. If you run out of strength, you will die and be unable to finish. Trying to move into a wall will not move you, but will deplete your strength by 1.

Objects

Objects come in two types: power-ups (such as healing potions) and monsters. They will be passed in every turn in a dictionary, keyed on the tuple (x, y) (the location), and containing the following values: a strength, and a label (such as "medkit" or "skeleton"). Power-ups will always have a strength of zero. Moving into a square containing a power-up consumes it, and increases your agent's strength. Moving into a square containing a monster represents choosing to fight it.

There is one special type of object: the Boss. The Boss always sits on the exit, and you must fight it before moving into that square and winning the game.

Fighting

Your starting strength will be a function of map size. When you choose to fight a monster, your strengths will be compared. The higher the delta between strengths, the more likely a win is. (For example, if both agents have a strength of 5, each is equally likely to win, while if your strength is 10 and the monster's is 7, you are 85% likely to win.) If you lose a fight, you will die; if you win a fight, your strength will be completely restored and the total amount will increase—you have become stronger! Defeating stronger monsters will increase your strength more, but of course those fights are harder to win.

Phase I (30%)

Your Phase I player will *not* be graded primarily based on your agent's success (although obviously, an agent that never finds the exit isn't great). We will grade based on the clarity and elegance of your code, the correctness of your approach, and how well your strategy captures appropriate AI concepts in trying to solve the problem, as well as a short description of what you implemented.

Your submission should not include any I/O in the required functions. Helper functions that perform I/O can be included but should not be called from the required functions.

In the first phase of the project, all monsters will stay where they are until vanquished.

Functions

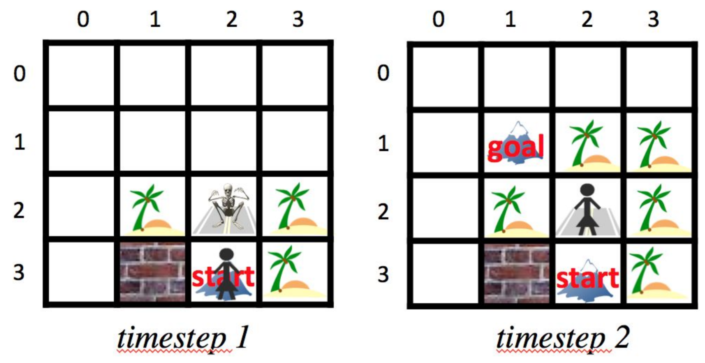
You must implement the following functions:

step(location <tuple>**, strength** <integer>**, map** <matrix>**, objects** <dictionary>**)**: This will take in the complete state and return one of four movement values: 'N', 'S', 'E', or 'W'.

Examples

Timestep 1: step((3,2), 16, [[u,u,u,u], [u,u,u,u], [u,s,p,s], [u,w,m,s]], {(2,2):"skeleton",12})

Timestep 2: step((2,2), 20, [[u,u,u,u], [u,m,s,s], [u,s,p,s], [u,w,m,s]], {})



The step call for the two timesteps shown above.
The Boss is not annotated here - coming soon!

Phase II (30%)

For Phase II, we will add dynamic monsters—they won't remain where you left them, and they may choose to fight you rather than waiting for you!

Details coming soon.

Writeup (20%)

Each team must submit a 6-8 page (3000-4000 words, not counting references and citations) project report. It should cover both your Phase I and Phase II, and describe your approach, your experience in designing and implementing the approach, and the evaluated performance of your system.

1. **A discussion of your final, implemented game strategies** and how you developed them. This discussion should have a section on your Phase I player and a separate section on your Phase II player, and can discuss what changes you made in between.
2. **A discussion of how your system draws on i from the AI literature.** This discussion should be focused on concepts *from class.* If you use other AI sources, please be very clear about what they are and consider checking with me to make sure they are suitably AI-"ish."
    - You may want to talk about better methods that you weren't able to implement. If you ended up with something relatively simple, but had some ambitious/interesting ideas that you weren't able to get working in the scope of the semester, you should talk about it here.
3. **References for the AI concepts you mention** (it's OK to cite the textbook, but please include specific section numbers).
4. **Experimental evaluation of your system.** How did you test it to see if it works? How well did it do? This section should discuss what worked well, and not so well, about your player's strategies; and whether the player behaves as expected. Both qualitative/anecdotal and quantitative data should be included in this discussion; this is where graphs, multiple-trial experiments, etc. go.

A Note on Cooperation

Teams should not share solutions and must follow the usual rules about sharing code, e.g., you should never have copies of or be using someone else's code or ideas except for occasional help debugging is okay. In this case, that means "Someone who is not on your team" – teams can and should be working together closely.

However, you may find that you have some good ideas about utility functions (tracing your player's behavior, analyzing game state, printing hypotheses, testing rules for equivalence...). You may think that some of this code could be useful for other students. You're probably right, and that's great! Please feel free to send the TA and professor any code that you develop that might be generally useful; we will vet it against the academic integrity policy and post it into a code repository. (If you're having trouble with class participation, this is a great way to contribute.)

Remember that if you're using a development repository such as GitHub, your code must not be publicly accessible!