B551 Assignment 0: Searching and Python

Fall 2020
Due: Tuesday September 15, 11:59:59PM Eastern (New York) time
Resubmissions or late submissions due: Tuesday September 17, 11:59:59PM Eastern (New York) time
(See *Deadlines* below for details.)

This assignment will give you practice with posing AI problems as search and an opportunity to dust off your coding skills. Because it's important for everyone to get up-to-speed on Python, for this particular assignment you must work individually. Future assignments will allow you to work in groups. Please read the instructions below carefully; we cannot accept any submissions that do not follow the instructions given here. Most importantly: please **start early,** and ask questions on Piazza or in office hours.

If you don't know Python, never fear – this is your chance to learn! But you'll have to spend some significant amount of out of class time to do it. Students in past years have recommended the Python CodeAcademy website, `http://www.codeacademy.com/learn/python`, and Google's Python Class, `https://developers.google.com/edu/python/`. There are also a wide variety of tutorials available online. If you're already proficient in one particular language, you might look for tutorials with names like "How to code in Python: A guide for C# programmers," "Python for Java programmers," etc. Feel free to share any particularly good or bad resources on Slack and/or Piazza. The instructors are also happy to help during office hours. **If you are struggling, please come and get help!** We can't help if you don't ask.

Guidelines for this and future assignments

Coding requirements. For fairness and efficiency, we use a semi-automatic program to grade your submissions. This means you must write your code carefully so that our program can run your code and understand its output properly. In particular:

1. **You must code this assignment in Python 3, not Python 2.** Please be forewarned that Python 2 is still quite popular, despite being obsolete, so many examples and tutorials you see online may be written in Python 2. There are many minor changes between the two versions (see `https://docs.python.org/3/whatsnew/3.0.html`), but two come up particularly often. First, in Python 3, *print* is a function, so its arguments must be surrounded by parentheses. Second, in Python 3, dividing one integer by another integer performs floating point division, instead of integer division.

2. **You should test your code on one of the SICE Linux systems** such as burrow.sice.indiana.edu. We will test your code on this system, so this is the only way to guarantee that your program will work correctly when we run it. You may (and probably should!) do your actual day-to-day development work on your laptop, but it is important that you periodically check that your code operates correctly on our servers, as minor differences in Python versions, available modules, etc. may cause your code to work differently — and may seriously affect your grade.

3. **Your code must obey the input and output specifications given below.** Because our grading program is automatic (and is not powered by advanced AI!), it will not understand your code's input and output unless you carefully follow the specifications given below. For example, your code should not require keyboard input (i.e., someone actually typing keys once your program is running) unless we specifically indicate it below. We usually will say that the last few lines of output from your program must be in a certain format; your code can output extra lines before that (e.g., for debugging purposes) because our grading program will look at just the last lines of output.

4. **You may import standard Python modules for routines not related to AI,** such as basic sorting algorithms and data structures like queues, as long as they are already installed on the SICE Linux servers. However, using additional modules is typically not required; we'll usually tell you if we feel a module would be helpful.

5. **Make sure to use the program file name we specify.** For example, for part 1, please call your program submission `route_pichu.py`, as we specify below. Submitting your program with a different name, or submitting multiple versions of your code, is just a recipe for confusing the graders.

Submissions. To help you make sure your code satisfies our requirements, we will give you the chance to revise your code based on some initial feedback from our grading program. If you submit your code by the **Due date** listed above, we will attempt to run your code through a simplified version of our grading program and send you feedback within 24 hours. You may then submit a revised version before the **Resubmission** listed above, with no grade penalty. If you do not submit anything by the **Due date** date, you may still submit by the **Resubmission** deadline, but it will count as a late submission and the 10% grade deduction will apply.

You'll submit your code via GitHub. We strongly recommend using GitHub as you work on the assignment, pushing your code to the cloud frequently. Among other advantages, this: (1) makes it easier to collaborate on a shared project in later assignments, when you are working in groups, (2) prevents losing your code from a hard disk crash, accidental deletion, etc., (3) makes it possible for the AIs to look at your code if you need help, (4) makes it possible to retrieve previous versions of your code, which can be crucial for successful debugging, and (5) helps document your contribution to team projects (since Git records who wrote which code). If you have not used GitHub before, instructions for getting started with git are available on Canvas and there are many online tutorials.

Coding style and documentation. We will not explicitly grade based on coding style, but it's important that you write your code in a way that we can easily understand it. Please use descriptive variable and function names, and use comments when needed to help us understand code that is not obvious.

Report. For each assignment, we'll require a written report that summarizes your programming solutions and that answers specific questions that we pose. Please put the report in the Readme.md file in your Github repository. For each programming problem, your report should include a brief overview of how your solution works, including any problems you faced, any assumptions, simplifications, or design decisions you made, any parts of your solution that you feel are particularly innovative, etc. These comments are your opportunity for us to better understand your code and the amount of work that you did in writing it; they are especially important if your code does not work as well as you would like, since it is a chance to document how much energy and thought you put into your solution. For example, if you tried several different approaches before finding one that worked, feel free to describe this process so that we can appreciate the work that you did that might not otherwise be reflected in your final solution.

Academic integrity. We take academic integrity very seriously. To maintain fairness to all students in the class and integrity of our grading system, we will prosecute any academic integrity violations that we discover. *Before beginning this assignment, make sure you are familiar with the Academic Integrity policy of the course, as stated in the Syllabus, and ask us about any doubts or questions you may have.* To briefly summarize, you may discuss the assignment with other people at a high level, e.g. discussing general strategies to solve the problem, talking about Python syntax and features, etc. You may also consult printed and/or online references, including books, tutorials, etc., but you must cite these materials (e.g. in source code comments). We expect that you'll write your own code and not copy anything from anyone else, including online resources. *However, if you do copy something (e.g., a small bit of code that you think is particularly clever), you have to make it explicitly clear which parts were copied and which parts were your own. You can do this by putting a very detailed comment in your code, marking the line above which the copying began, and the line below which the copying ended, and a reference to the source.* Any code that is not marked in this way must be your own, which you personally designed and wrote. You may not share written answers or code with any other students, nor may you possess code written by another student, either in whole or in part, regardless of format.

Part 1: Navigation

As we learned in Module 1, certain autonomous agents like to fly around the house and interrupt video recordings at the most inopportune moments. Suppose that a house consists of a grid of $N \times M$ cells, represented like this:

```
....XXX
.XXX...
....X..
.X.X...
.X.X.X.
pX...X@
```

As you can see, the map conssists of N lines (in this case, 6) and M columns (in this case, 7). Each cell of the house is marked with one of four symbols: `p` represents the agent's current location, `X` represents a wall through which the agent cannot pass, `.` represents open space over which the agent can fly, and `@` represents your location (presumably with video recording in progress).

Your goal is to write a program that finds the shortest path between the agent and you. The agent can move one square at a time in any of the four principal compass directions, and the program should find the shortest distance between the two points and then output a string of letters (compass directions) indicating that solution. The final line of output from your program should have the following format:

`[length-of-path] [path]`

where [length-of-path] is the length (cost) of the path you have found, and [path] is a string of compass directions (N, S, E, and W for North, South, East, and West) indicating the path. Your program should take a single command line argument, which is the name of the file containing the map file. For example:

```
[<>djcran@tank ~] python3 route_pichu.sh map.txt
Shhhh... quiet while I navigate!
Here's the solution I found:
16 NNNEESSSEENNEESS
```

You can assume that there is always exactly one `p` and one `@` in the map file. If there is no solution, your program should display path length `Inf` (indicating infinite path length) and not display a path.

To help get you started, we have provided some initial code that is already available in your GitHub repo. Here's what to do to complete the program.

1. We've already created a GitHub repository for you for this assignment. You can see the name of your repository by logging into IU Github, at `http://github.iu.edu/`. In the upper left hand corner of the screen, you should see a pull-down menu. Select `cs-b551-fa2020`. Then in the box below, you should see a repository called *youruserid*-a0, (If you do not see `cs-b551-fa2020` or a repository with your userid, it probably means that did not log into GitHub to create your account during the A Few Action Items activity during week 1 of class, so we were not able to add you to a team. Post a private message on Piazza so that we can create your repo manually.)

   To get started, from the SICE Linux machines, clone the github repository:

   `git clone git@github.iu.edu:cs-b551-fa2020/`*`your-repo-name`*

   If that doesn't work, instead try:

   `git clone https://github.iu.edu/cs-b551-fa2020/`*your-repo-name* where *your-repo-name* is the one you found on the GitHub website above. (If neither command works, you probably need to set up IU GitHub ssh keys. See Canvas for help.)

2. Now you should see a program called `route_pichu.py`. We have also provided a sample map file, `map.txt`. You can run our program like this:

   `python3 route_pichu.py map.txt`

   Unfortunately, the program does not work very well; it will probably enter an infinite loop and you'll have to press CONTROL-C to kill it. Nevertheless, the code is a good starting point, so familiarize yourself with it. Figure out the precise search abstraction that the program is using and include it in your report. In other words, what is the set of valid states, the successor function, the cost function, the goal state definition, and the initial state?

3. Why does the program often fail to find a solution? Implement a fix to make the code work better, and explain what you did in the report.

4. The existing code is missing a number of the specifications given above. (For example, it doesn't display the path as a string of compass directions.) Complete the implementation, and explain what you did in your report. Make sure to test your program on maps other than the one we've given, including ones of different sizes and shapes.

Part 2: Hide-and-seek

Suppose that instead of a single agent as in Part 1, you now own k agents. The problem is that these agents do not like one another, which means that they have to be positioned such that no two agents can see one another. Write a program called `arrange_pichus.py` that takes the filename of a map in the same format as Part 1 as well as a single parameter specifying the number k of agents that you have. You can assume $k \geq 1$. Assume two agents can see each other if they are on either the same row or column of the map, and there are no buildings between them. An agent can only be positioned on empty squares (marked with `.`). It's okay if agents see you, and you obscure the view between agents, as if you were a wall. Your program should output to the screen (in the last lines of its output) a new version of the map, but with the agents' locations marked with `p`. Note that exactly one `p` will already be fixed in the input map file. If there is no solution, your program should just display `None`. Here's an example on the same sample map file as in Part 1:

```
[<>djcran@tank ~] python3 arrange_pichus.py map.txt 9
...pXXX
.XXXp..
.p..Xp.
.XpX...
.X.XpXp
pX.p.X@
```

We've again given you some code to get started with, but it's not fully working; the configurations it finds often allow friends to see one another, and it can be quite slow. Fix the code so that it works, and then try to make it run as quickly as possible. In your report, explain the search abstraction you've used – what is the state space, initial state, goal state, successor function, and cost function?

Make sure to test your program on maps other than the one we've given, including ones of different sizes and shapes.

For optional **extra credit**, consider the same problem, but where two birds – er, agents – can see each other if they are on the same row, column, or diagonal and there is no wall (or you) between them. Instead of finding a configuration for a given k, the program should in this case instead find a configuration such that the number of agents is as large as possible. This extra credit functionality should be activated when the value of k is specified as 0, and the output should be the same as above: a configuration with as many `p`'s as possible.

What to turn in

Turn in the two programs on GitHub (remember to `add`, `commit`, `push`) — we'll grade whatever version you've put there as of 11:59PM on the due date. Also remember to put your report in the Readme.md file. To make sure that the latest version of your work has been accepted by GitHub, you can log into the github.iu.edu website and browse the code online.