B551 Assignment 1: Search and Python II

Fall 2024

Due: September 20, 11:59 PM Eastern (Bloomington) time
Late submissions accepted until Thursday September 22 with 10% penalty per day.

This assignment will give you practice posing AI problems as search and an opportunity to dust off your python coding skills. To help ensure that everyone is fully comfortable with python for future assignments, *In this assignment you will work individually.* 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 Q&A Community or in office hours.

If your Python skills are rusty, don't worry—this is your chance to refresh! But you might have to spend some significant amount of out of class time to do it. Students in past years have recommended the Python CodeAcademy website, `https://www.codecademy.com/catalog/language/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 notes on any particularly good or bad resources on the Q&A Community. The AIs are also happy to help on coding issues during office hours. If you are struggling, please come and get help! We can't help if you don't ask.

1 Guidelines for this and future assignments

1.1 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. 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.

3. 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 Luddy Linux servers. However, using additional modules is typically not required; we'll usually tell you if we feel a module would be helpful.

4. Make sure to use the program file name we specify. For example, for part 1, please call your program submission place_turrets.py, as we specify below. Submitting your program with a different name, or submitting multiple versions of your code, will cause our autograder to fail and you to lose points.

5. Please avoid using global (public) variables in your programs as they may cause your program to behave unexpectedly when run by our grading program. A global variable is one that is defined outside of any function. If you need to access the same variable in multiple functions, use return statements and function arguments instead of global variables.

6. You should test your code on silo.luddy.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.

7. To help you test, we have included an automated testing program in your github repo. We also use this testing program for autograding. Ensuring that your code works well with the testing program will help make sure we grade your code correctly. The program requires you to make a one-time installation. To install, type:

   ```
   pip install pytest-timeout
   ```

   Once installed you do not need to repeat this step in the future. To test your code, type:

   ```
   python3 -m pytest -v
   ```

   This will automatically run your programs on two sample test cases, and indicate whether your programs passed or failed. These test cases are just samples; we will run this on more cases while grading, so make sure to test your code thoroughly.

Submissions. 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 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. Being well versed with git commands of add, commit, push and pull will be useful for all upcoming assignments in this class.

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 short 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. You may not use generative AI systems to generate any part of your code.

Part 1: Search for Design

As the legend goes, the benevolent guardian of the castle, needs you to install multiple conscious guardian turrets to protect their castle. The problem is that these turrets do not like one another, which means that they have to be positioned such that no two turrets can see one another. Write a program called place_turrets.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 turrets that you have. You can assume $k \geq 1$. Assume two turrets can see each other if they are on either the same row, column, or diagonal of the map, and there are no walls (defined by 'X') between them. A turret can only be positioned on empty squares (marked with .). It's okay if turrets see you (you are defined by '@'), and you obscure the view between turrets, as if you were a wall (they won't attack you). Your program should output a new version of the map, but with the turrets' 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 False. Here's an example on the same sample output on the same map as in Assignment-0:

```
[<>djcran@silo ~] python3 place_turrets.py map1.txt 5
....XXX
.XXXp..
.p..X..
.X.X...
.X.X.Xp
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 turrets to see one another, and it can be quite slow. Fix the code so that it works correctly, and then try to make it run as quickly as possible. Again do not rewrite the entire code from scratch. 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. Check the starter code comments for specifications on the solve() function, because our autograder (and the pytest command above in Coding Requirements) calls it directly.

As described earlier, we have provided some initial code that is already available on GitHub. Here's what to do to run the program.

1. We've created a GitHub repository, a1-release, for this assignment. You can see the repository by logging into IU Github, at http://github.iu.edu/ or from Canvas. In the upper left hand corner of the screen, you should see a pull-down menu. Select cs-b551-fall2024. Then in the box below, click on view organization and you will see the repository. To get started, from the SICE Linux machines, clone the github repository:

   ```
   git clone git@github.iu.edu:cs-b551-fall2024/a1-release.git
   ```

If that doesn't work, instead try:

```
git clone https://github.iu.edu/cs-b551-fall2024/a1-release.git
```

(If neither command works, you probably need to set up IU GitHub ssh keys. See GitHub tutorial on Canvas for help.)

2. Now you should see a program called place_turrets.py. We have also provided two sample map files, map1.txt and map2.txt. You can run our program like this:

```
python3 place_turrets.py map1.txt 5
```

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. Do not rewrite the entire code from scratch, which will cause our autograder to fail and you to lose points.

4. Complete the program so that it finds and displays the correct solution. Check the starter code comments for specifications on the solve() function, because our autograder (and the pytest command above in Coding Requirements) calls it directly.

Turning in your work

Create a private repository using the name your_iu_id-a1 on GitHub under cs-b551-fall2024, where your_iu_id is the text before the @ sign in your IU email address. It should be the same as your login ID on Canvas (and your ID on GitHub:IU). Make sure that you stick to this naming scheme so that our autograder can locate your submission correctly. And make sure it is private so others will not see your submission. 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.