# B551 Assignment 2: Applied Search and Search Heuristics

Fall 2024

**Due: Wednesday, November 6 (changed from Tuesday, November 5), 11:59 PM Eastern (Bloomington) time**. The period for completing this assignment has been extended to allow for time preparing for the second midterm. We recommend starting it promptly and then pausing for midterm preparation. Late submissions accepted until Thursday, November 7 with 10% penalty per day after November 5.

This assignment will give you practice with posing AI problems as search, developing search heuristics, and applying search in messy real-world contexts. 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 Inscribe or in office hours.

## 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 mystical_castle.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.*** Please put a report describing your assignment in the Readme.md file in your Github repository. For each problem, please include: (1) a description of how you formulated the search problem, including precisely defining the state space, the successor function, the edge weights, the goal state, and (if applicable) the heuristic function(s) you designed, including an argument for why they are admissible; (2) a brief description of how your search algorithm works; (3) and discussion of any problems you faced, any assumptions, simplifications, and/or design decisions you made. These comments are especially important if your code does not work as well as you would like, since it is a chance to document the energy and thought you put into your solution. For example, if you tried several different heuristic functions before finding one that worked, feel free to describe this in the report so that we appreciate the work that you did.

***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 0: Getting Started!

To get started, clone the github repository:

```
git clone git@github.iu.edu:cs-b551-fall2024/a2-release.git
```

If that doesn't work, try -

```
git clone https://github.iu.edu/cs-b551-fall2024/a2-release.git
```

# Part 1: Road Trip!

If you stop and think about it, finding the shortest driving route between two distant places — say, one on the east coast and one on the west coast of the U.S. — is extremely complicated. There are over 4 million miles of roads in the U.S. alone, and trying all possible paths between two places would be nearly impossible. So how can mapping software like Google Maps find routes nearly instantly? The answer is A* search!

We've prepared a dataset of major highway segments of the United States (and parts of southern Canada and northern Mexico), including highway names, distances, and speed limits; you can visualize this as a graph with nodes as towns and highway segments as edges. We've also prepared a dataset of cities and towns with corresponding latitude-longitude positions. These files should be in the GitHub repo you cloned in step 0. Your job is to implement algorithms that find good driving directions between pairs of cities given by the user. Your program should be run on the command line like this:

```
python3 ./route.py [start-city] [end-city] [cost-function]
```

where:

- start-city and end-city are the cities we need a route between.
- cost-function is one of:
  - **segments** tries to find a route with the fewest number of road segments (i.e. edges of the graph).
  - **distance** tries to find a route with the shortest total distance.
  - **time** finds the fastest route, for a car that always travels five miles per hour above the speed limit.
  - **delivery** tries to find the safest route for cycling. (Assume that in each mile of roadway, there is a probability of accident $p(s) = 0.000001s$ where $s$ is the speed limit. The expected number of accidents on a given road segment is then $p(s) \times l$, where $l$ is the length of the road. For example, if the speed limit is 50mpg and a road is 10 miles long, the expected number of accidents on that roadway is 0.0005. The goal is to find a route with the fewest total number of expected accidents).

The output of your program should be a nicely-formatted, human-readable list of directions, including travel times, distances, intermediate cities, and highway names, similar to what Google Maps might produce. In addition, the last line of output should be the following line summarizing the route you found:

```
[total-segments] [total-miles] [total-hours] [total-expected-accidents] [start-city] [city-1] [city-2] ... [end-city]
```

Please be careful to follow these interface requirements so that we can test your code properly. For instance, the last line of output might be:

```
3 51 1.0795 0.002498 Bloomington,_Indiana Martinsville,_Indiana Jct_I-465_&_IN-37_S,_Indiana Indianapolis,_Indiana
```

Like any real-world dataset, our road network has mistakes and inconsistencies; in the example above, for example, the third city visited is a highway intersection instead of the name of a town. Some of these "towns" will not have latitude-longitude coordinates in the cities dataset; you should design your code to still work well in the face of these problems.

***Extra credit*** Implement an additional cost-function: statetour should find the shortest route from the start city to the end city, but that passes through at least one city in each of the 48 contiguous U.S. states.

# Part 2: Choosing Teams

In a certain Computer Science course, students are assigned to groups according to preferences that they specify. Each student is sent an electronic survey and asked to give answers to three questions:

1. What is your IU email username?
2. Please choose one of the options below and follow the instructions
   - You would like to work alone. In this case, just enter your userid in the box and nothing else.
   - You would like to work in a group of 2 or 3 people and already have teammates in mind. In this case, enter all of your userids (including your own!) in the box below, in a format like userid1-userid2 for a team of 2, or userid1-userid2-userid3 for a team of 3.
   - You would like to work in a group of 2 or 3 people but do not have any particular teammates in mind. In this case, please enter your user ID followed by one "zzz" per missing teammate (e.g. leake-zzz where leake is your user ID to request a team of 2, or leake-zzz-zzz for a team of 3).
   - You would like to work in a group of 3 people and have some teammates in mind but not all. Enter all of your ids, with zzz's to mark missing teammates (e.g. if I only have one teammate (vsshinde) in mind so far, I'd enter leake-vsshinde-zzz).
3. If there are any people you DO NOT want to work with, please enter their userids here (separated by commas, e.g. userid1,userid2,userid3).

Unfortunately, the student preferences may not be compatible with each other: student A may request working with student B, but B may request not to work with A, for example. This makes some complaints unavoidable. The course staff decides to try to find groups that minimize the time spent on complaints and grading. The course staff estimates that:

- They need $k$ minutes to grade each assignment, so total grading time is $k$ times number of teams.
- Each student who requested a specific group size and was assigned to a different group size will complain to the staff, requiring 1 minute of staff time.
- Each student who is not assigned to someone they requested sends a complaint email, which will take $n$ minutes for the staff to read and respond. If a student requested to work with multiple people, then they will send a separate email for each desired person they were not assigned to.
- Each student who is assigned to someone they requested *not* to work with (in question 3 above) complains to the Instructor, who meets with the staff for $m$ minutes. If a student is assigned to a group with multiple people they did not want to work with, a separate meeting will be needed for each.

The total time spent by the course staff is equal to the sum of these four components. You can assume that each student fills out the survey exactly once, and fills it out according to the instructions. Your goal is to write a program to find an assignment of students to teams that minimizes the total amount of staff time, subject to the constraint that no team may have more than 3 students. Your program should take as input a text file that contains each student's response to these questions on a single line, separated by spaces. For example, a sample file might look like:

```
leake leake-vsshinde-yahrkappa dcshetty
dcshetty dcshetty _
sulagaop sulagaop-xxx-xxx _
fanjun fanjun-xxx yahrkappa
yahrkappa yahrkappa leake,fanjun
vsshinde vsshinde-dcshetty _
```

where the underscore character (_) indicates an empty value. Your program should be run like this:

```
python3 ./assign.py [input-file]
```

To make life easier for all, we will assume that $k = 30$ $m = 20$ and $n = 10$ for this assignment, where $k$, $m$, and $n$ are values for the parameters mentioned above, and the output of your program should be a list of group assignments, one group per line, with user names separated by dashes followed by the total time requirement for the instructors, e.g.:

```
[leake@silo part3]$ python3 ./assign.py input-file
leake-vsshinde-yahrkappa
dcshetty
sulagaop-fanjun
123
```

(In this example, the cost of 123 was calculated as k×3+1×3+m×1+n×1 = 30×3+1×3+20×1+10×1: there are 3 groups, each requiring 30 minutes of grading, three people (sulagaop, yahrkappa, and vsshinde) didn't get the requested number of teammates, one person (yahrkappa) had to work with someone they requested not to work with (leake), and one person (vsshinde) didn't get to work with a person they requested (sahmaini).)

*Note and Hint:* It may not always be possible to find the actual best solution in a reasonable amount of time, and "reasonable amount of time" may differ from one user to the next. Our grading program will eventually exit your program if it takes too long. You should make sure that your program will always have provided a solution by the time cut-off, but also that the program makes good use of the time it is provided. One trick is that since our grading program only looks at the last solution that you output, your program can output multiple solutions. This may be useful if your approach can quickly produce an estimate of the solution, and then as it performs more computation, finds better and betters solutions.

### 1.2 General Note

These problems intentionally give scope for pushing for high performance. Try to come up with the best solutions you can and discuss with the AIs if you have questions about whether your program's performance is sufficiently good.

## Turning in your work

Create a private repository using the name your_iu_id-a2 (If you have teammates your_iu_id-teammate1_iu_id-a2 or your_iu_id-teammate1_iu_id-teammate2_iu_id-a2) 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 your work via 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.