CS 330 Artificial Intelligence and Game Development
Spring 2022

Mikel D. Petty, Ph.D., pettym@uah.edu, 256-824-6140

Instructions that apply to all programming assignments in this course are on this page.
Instructions specific to the individual programming assignments are on the following pages.

General Instructions

1. Write your program(s) in C++, Java, or Python. Any other language requires instructor pre-approval. R is not allowed. (If you really want to program in R, see me offline.)
2. Use the same algorithms and calculations as shown in the course textbook and lecture slides.
3. Write your code with reasonable attention to software engineering quality. Include explanatory comments as needed. See my R code for examples of commenting that I consider to be sufficient.
4. Combine all deliverables into a single zip file before submission.
5. Include the name(s) of the students who contributed to the assignment in both the main program's comments and in the submission zip file's filename.
6. Submit the zip file containing the deliverables via Canvas file upload.
7. Due dates for the programming assignments are given on the Canvas course Assignments page. Submissions are due at 11:59pm (not 12:00 midnight) on the due date.
8. If you submit late, do not submit via Canvas; send the zip file to me as an email attachment. Send from your official UAH email address, not Canvas; Canvas messages can drop attachments.

Teaming

1. You are permitted and encouraged to work in groups of up to two (2) students on the programming assignments. Both students in a group will receive the same grade.
2. Teams need only submit once. Either team member may submit. Include both team members' names in the submission file's filename and the main program's comments.
3. Teams are not required to be permanent. You may change partners, or change from teaming to not teaming, or vice versa, for different assignments.
4. Teaming is not required. Individuals may submit with no grade penalty or bonus.

General Recommendations

1. Don't wait until the day before the due date to start the assignments. Allocate enough time to do a good job, and allow for unexpected difficulties.
2. In previous courses, some students have tried to do a line-by-line port of my example R code into C++, Java, or Python. You are welcome to try that, but it may not be the best approach. It may be better to use my code to understand the algorithms, and then implement the algorithms natively in your chosen programming language.
3. The algorithms in the programming assignments are well known and it is easy to find code for them online. If anyone is tempted to use code found online, please resist the temptation and instead write your own. Not only are the consequences of plagiarism, if discovered, distinctly undesirable for everyone involved, but more importantly, you will learn a lot more from the experience of writing and debugging your own implementations.

CS 330 Artificial Intelligence and Game Development
Spring 2022

Programming Assignment 3: A* Pathfinding

Objective

Implement the A* pathfinding algorithm and to use it to find paths in a pathfinding graph that is input to the program.

Requirements

You may implement either the pseudocode version of A* (Millington section 4.3.3), which creates and stores node records in separate Open and Closed lists, or the node array version of A* (Millington section 4.3.7), which stores node information in an array of node records. Implement a suitable data structure to represent the graph that includes the nodes' locations (*x* and *z* coordinates) and the connections' directionality. Use Euclidean 2D distance as the A* heuristic.

Two data files containing the Adventure Bay pathfinding graph have been uploaded to Canvas. Your program must load the pathfinding graph data structure from the information in the Adventure Bay files, and then find and report the paths listed later in this document.

Your implementation should:

1. Input the Adventure Bay pathfinding graph files and load the data to a suitable graph data structure of your choice. Do not hardcode the nodes and connections in your program.
2. Output the structure of the loaded pathfinding graph by listed all of the nodes, connections, and connection weights.
3. Output the shortest (lowest cost) path for each of five pairs of nodes; the specific start and goal nodes for the five test cases will be listed later in this document. For each of the test cases, report the start and goal nodes, the sequence of nodes from start to goal (inclusive), and the total cost of the path.
4. Write all output for items 2 and 3 above to a single text file.

No plotting is required for this assignment. The output of your program is a text file. You are not required to implement in your program and include in your output the iteration-by-iteration display of node status that my program generates.

Deliverables

1. Source code file(s) for the program
2. Output text file

Grading

The assignment is worth a maximum of fifteen (15) points. The grading rubric is as follows:

Points	Criterion
0	Assignment not submitted or deliverables missing
1	Assignment deliverables all submitted
0-2	Software engineering quality of the program reasonably good
0-2	Adventure Bay data files read, loaded to data structure, and output
0-3	A* algorithm present and correct
0-2	Pathfinding list data structure or node array used
0-5	Each requested path present, has correct nodes, and has correct cost
0-15	Total

The grading items are independent of each other (except, of course, that you cannot get any of the other points if you don't turn the assignment deliverables in). Note that if your program does not correctly find any of the requested paths, the highest score possible is 10.

Input and Test Cases

The Adventure Bay pathfinding graph is in two text (.txt) files in a non-R CSV format. One file contains the nodes and the other containing the connections of the graph. There are comments in the text files documenting their structure. Note that both files contain data fields that are used to plot the pathfinding graph; you will not need that data for your assignment, and your program should simply ignore that data.

Find these five paths for your submission:

#	Start node	Goal node
1	1	29
2	1	38
3	11	1
4	33	66
5	58	43

You may read these five test cases from an input file or simply hardcode them in your program.

Resources

All of the following have been posted to the course Canvas page in Files > Programming Assignments > Program 3: A* Pathfinding:

1. My R code for A*. Note that my code (1) uses the node array data structure, and (2) includes many features that you do not need for this assignment, including Dijkstra's algorithm, code to plot the pathfinding graph, and automated testing code. If you have questions about my code, please attend office hours or contact me.
2. A sample output file. Your output file should resemble its format fairly closely. Note that the sample output shows the paths used for the examples in Lecture 16, not the paths required for your assignment.
3. An image of the Adventure Bay pathfinding graph for your reference.

Due date

See Assignments page in the course Canvas website.

Assignment-specific recommendations
(See the general instructions for general recommendations.)

1. Begin testing your program with simple graphs, such as the Wikipedia 1 and Wikipedia 2b graphs presented in class, before tackling the Adventure Bay graph.
2. Test your program on the Adventure Bay graph using the Adventure Bay examples shown in Lecture 16.
3. Implementing the iteration-by-iteration display of node status produced by my program and shown in Lecture 16 is not required. However, doing so may help you to debug your program.
4. If you decide to team, one way to partition the project between two students is the following: One person writes code that reads the pathfinding graph data files, builds the pathfinding graph data structure, and reports the pathfinding graph to the text file. The other student implements the A* algorithm itself. However, teams are not obligated to use this approach and are free to partition the work any way they wish.

End of Programming Assignment 3