Implement the A* Algorithm

90 Points Possible

2/17/2026

Unlimited Attempts Allowed

Details

Implementation of A* Algorithm

Upload the code and results to ELMS

Each student does their own assignment.

Two important notes:

1) Any Programming Language is allowed, but I'll provide starter code in C++ (since there is more overhead in C++ than, say Matlab or Python) you can find it in the Code/GraphSearch folder here in ELMS.

2) Regarding Queue Data Structures:

- You are free to use any priority queue (heap) code that you can find (online, or otherwise)
- Please **document** where you obtained this priority queue code in a **readme** file
- (I will provide C++ code, but don't feel obligated to use it)

Data files for the homework assignment are located in Files/Homeworks/H1

For each problem (see problems.txt) you are given a node file, an edge file, a start node ID, and a goal node ID. More information about node and edge file formats follows:

- Node files: a CSV File of nodes. The first line tells the number of nodes in the file. After that, each line in the file is:
  - x,y
    - where x and y are the x and y position of a node in a 2D c-space
    - the x and y on the first line are for node 1, the second line are for node 2, etc.
- Edge files: A CSV File ofedges. The first line tells the number of edges in the file. After that, each line in the file is:
  - i,j,d
    - i is the index of the node at the start of the edge
    - j is the index of the node at the end of the edge
    - d is the length of the edge

What you turn in: a zip file (please use zip or gz) containing:

- Your code (again, any language is ok)
- Documentation about where the priority queue code that you are using came from.
- For each problem:
    - A text csv file containing the path that is computed between the start and goal (1 file per each problem) with one node's data per line in the format:
        - ID, x_location, y_location
    - **A picture containing the graph, the search tree, and the path that was found**

Starting Code

Code for this assignment (including some matlab functions to help with visualization, and my C++ starter code) can be found in **Files/Code/GraphSearch**

plot_graph.m is a matlab file that will read in and then plot a graph given the node and edges files, you can also specify IDs to display as the start and the goal. This is useful even if you choose not to use my C++ code as a starting point.

My C++ code:

- README.txt (look at this first)
- heap (queue) files:
    - heap.h
    - heap.cpp
    - heap_test.cpp
- graph search starter code
    - graph.h
    - simple_graph_search.cpp
- Search visualization (loads files output by simple_graph_search.cpp and then displays the search tree and resulting path in matlab)
    - display_search.m
- NOTE: node IDs start at 1 in the homework files, but the C++ code that I provide assumes that IDs start at 0. Thus, the file i/o will automatically decrease the ID of each node by 1 when reading in a file and increase the ID by 1 when writing out a file. **HOWEVER, you'll need to be sure to use ID-1 instead of ID when setting the start and goal nodes inside of simple_graph_search.cpp**
- NOTE: the simple graph search code uses a very naive random coin flip to determine which neighbor a particular nodes uses as its parent (this is obviously not A*). You will need to edit the node data structure in graph.h to store/track the values that A* requires. You will also need to modify the function "expand" in simple_graph_search.cpp to run the A* algorithm instead of the random thing it is currently doing.

View Rubric

A* grading

Criteria	Ratings		Pts
provided any code	**10 pts** **Full Marks**	**0 pts** **No Marks**	/ 10 pts
code looks correct by a quick inspection	**10 pts** **Full Marks**	**0 pts** **No Marks**	/ 10 pts
files are in the correct format	**10 pts** **Full Marks**	**0 pts** **No Marks**	/ 10 pts
path problem 1	**10 pts** **Full Marks**	**0 pts** **No Marks**	/ 10 pts
path problem 2	**10 pts** **Full Marks**	**0 pts** **No Marks**	/ 10 pts
path problem 3	**10 pts** **Full Marks**	**0 pts** **No Marks**	/ 10 pts
path problem 4	**10 pts** **Full Marks**	**0 pts** **No Marks**	/ 10 pts
path problem 5	**10 pts** **Full Marks**	**0 pts** **No Marks**	/ 10 pts
documented where queue code came from	**10 pts** **Full Marks**	**0 pts** **No Marks**	/ 10 pts
			Total Points: 0