CSCI 3202 - Introduction to Artificial Intelligence
Instructor: Hoenigman
Assignment 3
Due Friday, September 16 by 4pm

Problems:

A* Search
There is a .txt file on Moodle that contains a list of nodes and edges that can be used to build a graph. The first few lines of the file look like:

```
[S,A,2]
[A,B,1]
[B,C,4]
[B,P,6]
```

You can read this as, there is a node called S, which connects to node A with an edge weight of 2. The node A connects to node B with an edge weight of 1. The node B connects to nodes C and P with weights of 4 and 6 respectively.

The .txt file also includes heuristic values for a search from S to F. For example,

```
A=8
B=10
```

means that the estimate for the cost from A to F is 8, and the estimate for the cost from B to F is 10.

Write a program to do the following

1. Build a graph from the text file provided. The graph is undirected, any edge in the text file also needs to be created in the opposite direction as well. For example, your graph needs an edge from S to A with weight of 2 and from A to S with a weight of 2.
2. Apply A* search and Dijkstra's algorithm to find the shortest path through the graph. The path includes the cost and the nodes traversed.
3. Compare A* and Dijkstra's for how long it took each algorithm to find the solution. The comparison needs to include the number of nodes evaluated.

Document your results

- In addition to the code you submit, you need to submit a write-up that describes your results.
- Included in the write-up:
    - Purpose of the assignment.
    - Data used: Include a picture of the graph generated from the .txt file.
    - Procedure: The algorithms you implemented and ran

- Results: The shortest path that each algorithm generated and the nodes that each algorithm evaluated as the algorithm was executing. Comment on whether either algorithm was more efficient, i.e. evaluated fewer nodes. If they were the same, provide an example where a modification to the data would have provided a different result.

What to submit

Zip your source code and write-up and submit it to moodle. In assignment 1, we wrote a simple graph class. You are welcome to use that code as your starting point for this assignment. You will need to modify the nodes to include additional properties to support A* and Dijkstra's algorithm.

Other details

The TAs will assume your code is in Python 2 unless it is really clearly stated that you used Python 3. They will grade your code by running it command line in Ubuntu 16.04. The name of the file used to build the graph needs to be a command-line argument to your program.

Grading rubric

The rubric divides the assignment into 30% of the grade for the write-up and 70% of the grade for the code. Points shown in the rubric should sum to 100 - 30 points for the write-up and 70 points for the code.

Grading rubric for assignment code

If code doesn't run, use this rubric.

Code Quality	Points awarded
No code present	0
Code is submitted, but doesn't compile. If the code doesn't compile, the max points available is 40. You do not need to both checking the other features of the code shown in this rubric.	28

If the code runs, use this rubric.

Code Quality	Points awarded
Code compiles and runs.	49
Code handles filename as a command-line argument	3.5
Code produces a graph of nodes and edges.	3.5
Code correctly implements Dijkstra's algorithm	3.5
Code correctly implements A* algorithm	3.5
Code produces correct path and weight. If there isn't a print statement that displays this information, you may need to add one to check.	7

Grading rubric for assignment write-up

Report quality	Points awarded
Report includes sections for purpose, data used, procedure, and results.	12
Purpose section clearly describes comparing A* and Dijkstra's search results.	3
Data used section clearly describes using a text file provided with nodes and edges in a graph.	3
Procedure section includes implementing A* and Dijkstra's algorithm and comparing their outputs, including cost and path.	3
Results section includes the path and	3

cost that each algorithm produced, they should be the same.	
Results section comments on the efficiency of each algorithm, either by evaluating nodes added to the open list or absolute time to run the algorithm.	3
Results section includes an example of where the two algorithms would evaluate different nodes.	3