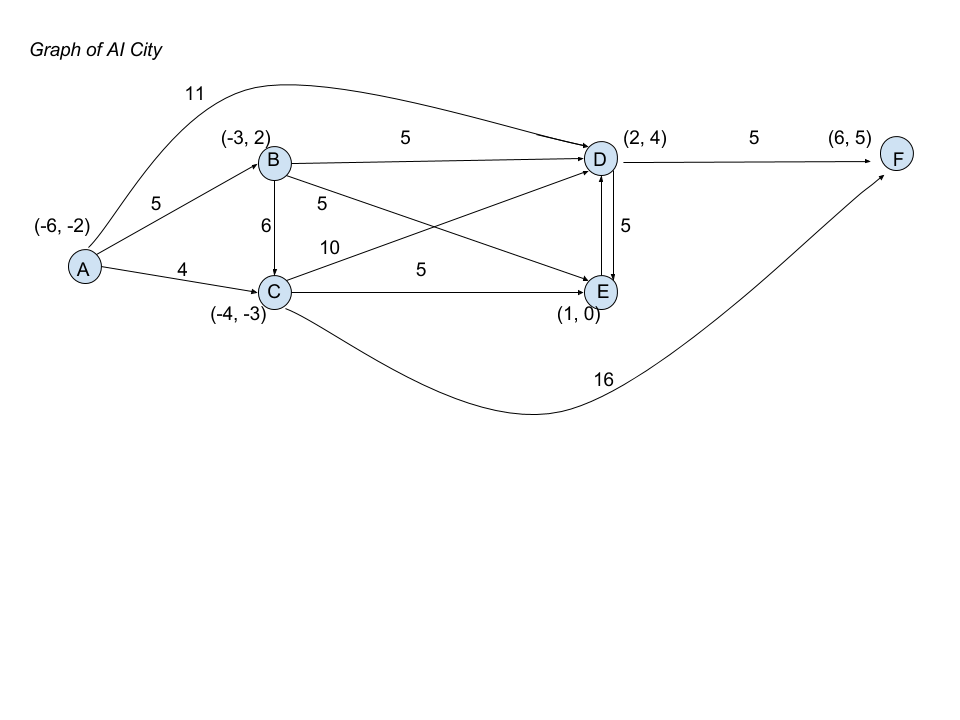
B351 Assignment 2: A\* Search

**Description:** For this assignment, you will be implementing A\* search to find the shortest path between two nodes on the graph, using Euclidean distance as your heuristic. For example, here is a graph of AI City, as is provided as a sample graph to test with in the a2.py file. Note that there is no specific start or goal state, you may test with any states you choose.



As you’ll see, the path cost between two nodes is not necessarily going to be exactly the same as their Euclidean distance. On this graph, the Euclidean distance is an admissible heuristic (the estimate is less than or equal to the true cost), on all other graphs you make for testing, you’ll want to be sure that this is the case.

**What you need to do:**

1. Download the a2.py file as a template for the assignment.
2. Fill out the euclidean\_dist function

This function is simple -- given two nodes, use their coordinate position to return the Euclidean distance

1. Fill out the select\_node\_to\_expand function

This function should take the current frontier and return the node with the lowest cost (this is the node that we want A\* to explore further)

1. Fill out the update\_frontier function

This function, when given a node to explore, the current frontier, the goal state, and the adjacency list representing the graph, should return the frontier with:

1. the explored state removed
2. All nodes directly reachable by the explored node added
3. Only the lowest-cost version of any duplicate nodes preserved
4. Fill out the aStar function,

This is how you will run the A\* algorithm, using the helper functions you wrote above. It takes the node list and adjacency list for the graph to be searched, as well as a start node and a goal node. When finished, this function should return a tuple with two elements: the final explored list, and the cost of the goal node.

1. Be sure that your function will perform correctly on any connected graph. Changing the start and goal nodes on our graph, or testing on one you create, should achieve this. Be sure you can handle the case where the goal is unreachable!

**Grading**

You will graded on having produced a correct implementation of A\*.

The points will be roughly divided as follows:

10 pts – euclidean\_distance

15 pts – select\_node\_to\_expand

35 pts – expand\_frontier

40 pts – working A\* search