**IT159: Intro to Artificial Intelligence**

**Lab#3/Assignment#3**

**Informed Search in Pac-Man**

There are three exercises in this lab:

1. Best-first search
2. A\* Search

# Introduction

In this assignment, your Pac-Man agent will find paths through its maze world to reach a particular location. You will build general search algorithms and apply them to many Pac-Man scenarios.

**Files you'll edit:**

search.py Where all of your search algorithms will reside.

searchAgents.py Where all your search-based agents will reside. [ONLY for Ex: 3]

**Files you should look at but NOT edit:**

|  |  |
| --- | --- |
| util.py | Useful data structures for implementing search algorithms. |
| pacman.py | The main file that runs Pac-Man games. This file describes a Pac-Man GameState type, which you use in this lab. |
| game.py | The logic behind how the Pac-Man world works. This file describes several supporting types like AgentState, Agent, Direction, and Grid. |
|  |  |

# Finding a fixed food dot using Informed Search

**Exercise 1:** Implement the Best-First Search (BFS) algorithm in the bestFirstSearch function in search.py. Test your code the same way you did for other search algorithms.

python pacman.py -l tinyMaze -p SearchAgent -a fn=befs

python pacman.py -l mediumMaze -p SearchAgent -a fn=befs python pacman.py -l bigMaze -p SearchAgent -a fn=befs -z .5

Does BFS find a least cost solution? How many nodes are expanded?

**Exercise 2:** Implement the A\* Search algorithm in the aStarSearch function in search.py. Use the same algorithm as shown in your text (or class). aStarSearch function takes an optional heuristic function as an argument. The heuristic function itself takes two arguments (a state in the search problem, and the problem itself). search.py provides a nullHeuristic function that you can look at. Also, in the searchAgents.py a Manhattan heuristic as well as Euclidian heuristic function is defined. Test your code the same way you did for other search algorithms.

python pacman.py -l tinyMaze -p SearchAgent -a fn=astar

python pacman.py -l mediumMaze -p SearchAgent -a fn=astar python pacman.py -l bigMaze -p SearchAgent -a fn=astar -z .5

To specify a heuristic function from searchAgents.py, use the following:

python pacman.py -l bigMaze -z .5 -p SearchAgent -a

fn=astar,heuristic=manhattanHeuristic

The function manhattanHeuristic() is already written in searchAgents.py. Alternatively, you could write your own in search.py

# What to submit

1. Fill out the table below:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Best First Search** | | |  | **A\* Search** |  |
| **Maze** | **#nodes expanded** | **Solution length** | **Is it optimal?** | **#nodes expanded** | **Solution length** | **Is it optimal?** |
| **tiny** |  |  |  |  |  |  |
| **medium** |  |  |  |  |  |  |
| **big** |  |  |  |  |  |  |

1. What happens on openMaze for the various search strategies?
2. For each exercise where a heuristic is used, clearly show/mention the heuristic function.
3. Based on the above, a short discussion/reflection of how the searches compare to each other and to the uninformed searches from Assignment#2.
4. Source code includes search.py and searchAgents.py. This should include your code for the search node, Best-First Search, and A\*.
5. Please create a folder called "yourname**\_**studentID\_Lab3" that includes all the required files and generate a zip file called "yourname\_ studentID \_Lab3.zip".
6. Please submit your work (.zip) to Blackboard.