1. Identify two very different existing AI systems and characterize them based on the PEAS problem formulation. Give a detailed explanation of the applications based on these four fundamental concepts.

We know that there are different types of agents in AI. PEAS System is used to categorize similar agents together. The PEAS system delivers the performance measure with respect to the environment, actuators, and sensors of the respective agent. Most of the highest performing agents are Rational Agents.

Rational Agent: The rational agent considers all possibilities and chooses to perform the highly efficient action. For example, it chooses the shortest path with low cost for high efficiency.

PEAS stands for a *Performance measure, Environment, Actuator, Sensor*.

1. **Performance Measure:** Performance measure is the unit to define the success of an agent. Performance varies with agents based on their different precepts.
2. **Environment**: Environment is the surrounding of an agent at every instant. It keeps changing with time if the agent is set in motion. There are 5 major types of environments:
   - Fully Observable & Partially Observable
   - Episodic & Sequential
   - Static & Dynamic
   - Discrete & Continuous
   - Deterministic & Stochastic
3. **Actuator**: An actuator is a part of the agent that delivers the output of action to the environment.
4. **Sensor**: Sensors are the receptive parts of an agent that takes in the input for the agent.

Agent	Performance	Environment	Actuator	Sensor
Automated Car Drive	The comfortable trip, Safety, Maximum Distance	Roads, Traffic, Vehicles,	Steering wheel, Accelerator, Brake, Mirror	Camera, GPS, Odometer
Subject Tutoring	Maximize scores, Improvement is students	Classroom, Desk, Chair, Board, Staff, Students	Smart displays, Corrections	Eyes, Ears, Notebooks

2. To read file in the graph library and to create nodes and edges, I have used the following code snippet.

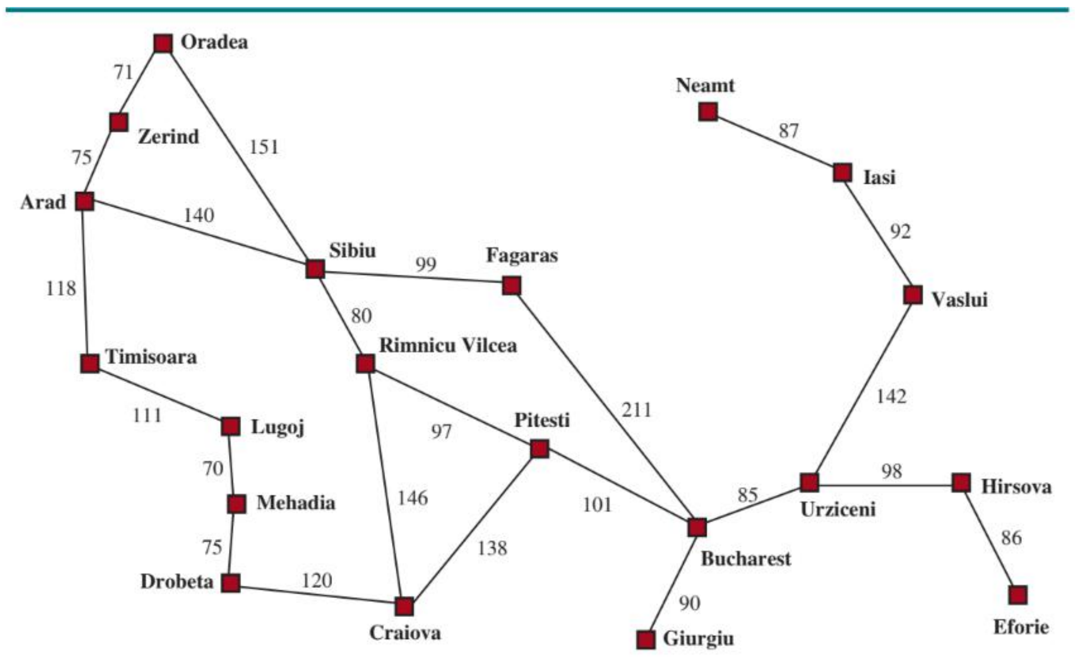
Create graph nodes and edges between the node

```
with open(file, "r") as ef:
    for edges in ef:
        edges_content = re.split('[:\n]', edges)
        graph.add_edge(Node(edges_content[0]), Node(edges_content[1]),
edges_content[2])
```

Heuristice Nodes with latitude and longtitude

```
with open(heuristic, "r") as hf:
    for nodes in hf:
        node_content = re.split('[:\n]', nodes)
        h_nodes[Node(node_content[0]).name] = (float(node_content[1]),
float(node_content[2]))
```

3.



A simplified road map of part of Romania, with road distances in miles.

Average Time to travel from one City to every other city using graph searching algorithms

Algorithm	Time(seconds)
Breadth first search	6.879736927083669e-06
Deepth first search	7.045263167751009e-06
Dijkstras shortest path search	1.9818421031614936e-05
Astart search(A*)	5.759999997903115e-05

Test 1

Algorithm	Time(seconds)
Breadth first search	6.906315832709783e-06
Deepth first search	7.166315840310601e-06
Dijkstras shortest path search	1.9813157895872505e-05
Astart search(A*)	5.3945789510946365e-05

Test 2

Algorithm	Time(seconds)
Breadth first search	8.282631547205567e-06
Deepth first search	8.913684211476815e-06
Dijkstras shortest path search	2.7335789448108317e-05
Astart search(A*)	7.0659473711107e-05

Test 3

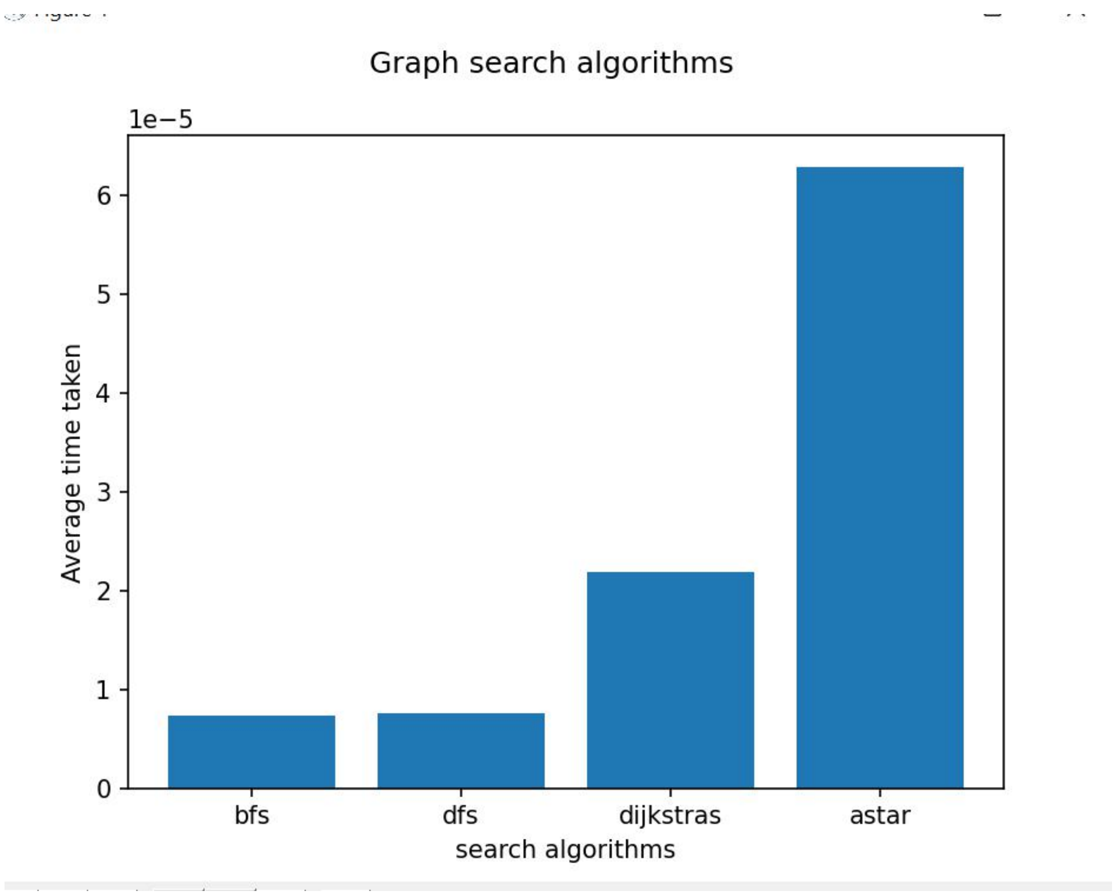
Algorithm	Time(seconds)
Breadth first search	7.285789553478887e-06
Deepth first search	7.474999945675097e-06
Dijkstras shortest path search	2.0752105291122508e-05
Astart search(A*)	6.919473686426308e-05

Test 4

The above benchmarks are based on finding the path between each nodes in the road map graph.There are 20 nodes in the road map.By fixing a certain node in the graph as a solution to find, I have tested the 4 graph algorithms average time to find a valid path to the solution.I have tested each node as solution starting from any node except itself.Since there are 380 nodes as a solution except the self nodes,I have divided the total time to find each node as a solution to 380.

Algorithm	Time(seconds)to get solution
Breadth first search	7.338618465119476e-06
Deepth first search	7.650065791303381e-06
Dijkstras shortest path search	2.1929868416679567e-05
Astart search(A*)	6.28500000163369e-05

Average Time in 4 tests



Observations

- The time measure of the each algorithm in this scenario depend on other running algorithms and the cpu clock speed of the the computer used.
- Each **algorithm** has a scenario to work optimally and completely.
- for example if we don’ t want to consider the weight or cost of the nodes or if we think their weight is the same, we have better results using depth first search and breadth first search graph algorithms.
- if we want the shortest distance **dijkstras** or A* algorithm works fine but if we want to find the number of layers to find the solution, breadth first search works well.
- In the above bar graph, the **A* search algorithm** takes the longest average time to find the solution compared the rest of the algorithms. The reason behind is the algorithm calculates the distance between the neighbors of the current node to the destination node to choose the shortest path possible. That is the heuristic function that informs the algorithm to choose the optimal path to the destination node.

- **The dijkstras shortest path search algorithm** used to find the shortest path between source to the destination as the A* search algorithm.But the two algorithm are different.Dijkstras shortest path algorithm is totally blind about the destination node.It is uninformed search algorithm.where as A* star search is an informed search algorithm which is dependent on the the heuristic function that allow the algorithm to maximize the utility and minimize the costs to get the solution node.
- **Depth first search algorithm** searches the solution node as deep as possible and backtracks if it does not get the destination until the stack is empty.if the solution is found it returns the paths it has traveled if not, path not found.The algorithm does not consider any edge weights or node values. It assumes the weight between edge node is one.It is not also a chosen one to search the shortest distance between the source and destination.The average time of depth first search for each node in the road map graph is higher than breadth first search.Because the algorithm goes deeper to find the solution.So it takes longer time than breadth first search in average even though it give better time sometimes.
- **Breadth search algorithm** takes better time in average.It aims to find the shortest path blindly.The algorithm find the shortest path storing its parent when travel each nodes. The when it founds the destination it iteration through the parents dictionary in revers order to find the path it has traveled.since Breadth first search goes layer by layer it is faster to find the path between nodes.

Average Length to travel from one City to every other city using graph searching algorithms

Algorithm	Length(no. Of nodes)
Breadth first search	4.721052631578948
Deepth first search	11.0
Dijkstras shortest path search	4.936842105263158
Astart search(A*)	4.942105263157894

Test 1

Algorithm	Length(no. Of nodes)
Breadth first search	4.721052631578948
Deepth first search	11.0
Dijkstras shortest path search	4.936842105263158
Astart search(A*)	4.942105263157894

Test 2

Algorithm	Length(no. Of nodes)
Breadth first search	4.721052631578948
Deepth first search	11.0
Dijkstras shortest path search	4.936842105263158
Astart search(A*)	4.942105263157894

Test 3

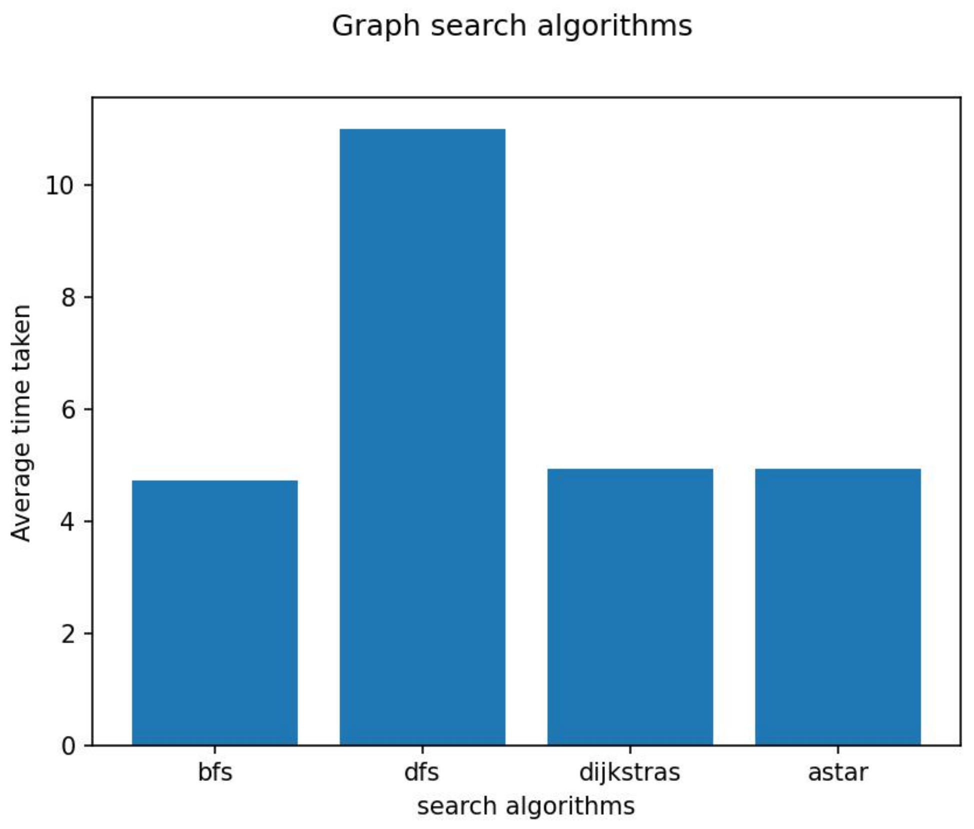
Algorithm	Length(no. Of nodes)
Breadth first search	4.721052631578948
Deepth first search	11.0
Dijkstras shortest path search	4.936842105263158
Astart search(A*)	4.942105263157894

Test 4

The above tables are based on finding the path between each nodes in the road map graph.There are 20 nodes in the road map.By fixing a certain node in the graph as a solution to find, I have tested the 4 graph algorithms average numbers of node traversed to find a valid path to the solution.I have tested each node as solution starting from any node except itself.Since there are 380 nodes as a solution except the self nodes,I have divided the total Length to find each node as a solution to 380.

Algorithm	Length(no. Of nodes)
Breadth first search	4.721052631578948
Deepth first search	11.0
Dijkstras shortest path search	4.936842105263158
Astart search(A*)	4.942105263157894

Average Length in 4 tests



Observation:

In the 4 tests the average number of node or length traversed from every node to any other node in the graph was the same.

Since the Dijkstras shortest path algorithm and A* search algorithm tries find the shortest path, both have the approximately the same average length.
Depth first search algorithm takes the longest path.it goes deep as much as possible to find the solution.so it traverse the longest node in average
Bread first search algorithm takes the shortest number of nodes to find the solution node.