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*.
- ✓ Performance Measure: Performance measure is the unit to define the success of an agent. Performance varies with agents based on their different precepts.
- ✓ 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
  1.Episodic & Sequential
  2.Static & Dynamic
  3.Discrete & Continuous
  4.Deterministic & Stochastic
- ✓ Actuator: An actuator is a part of the agent that delivers the output of action to the environment.
- ✓ Sensor: Sensors are the receptive parts of an agent that takes in the input for the agent.

Agent	Performance	Environment	Actuator	Sensor
Automated Taxi Driver	The comfortable trip, Safety, Maximum Distance,Optimum speed	Roads, Traffic, Vehicles,	Steering wheel, Accelerator, Brake, Mirror	Camera, GPS, Odometer
Microsoft	Availability	Windows	microphone,Keyb	smart

Cortana	to alert users answer user questions correctly set reminders show users the weather of the day in consistent fashion, Correctly recognize voices , Save time	operating system, Microsoft phones,Micro soft edge	oard	speakers that sense voice comman ds

PEAS Description for Automated Taxi driver:

Performance :

Safety: Automated system should be able to drive the car safely without dashing anywhere and without creating any damage of on the environment. Also should be controllable .
Optimum speed: Automated system should be able to maintain the optimal speed depending upon the surroundings.It should be able to maximize distance by decreasing costs such as fuel and time.
Comfortable journey: Automated system should be able to give a comfortable journey to the end user. The passengers shouldn't be worry about accidents and horrible driving speeds. The Automated car shouldn't create crushes with poles or road edges.

Environment:

Roads: Automated car driver should be able to drive on any kind of a road ranging from city roads to highway. The Automated car should follow left and right rules when it comes a new environment.
Traffic conditions: You will find different sort of traffic conditions for different type of roads.

Pedestrians: The AI in the Taxi should watch pedestrians while crossing roads and when they cross roads accidentally without using Zebras.
Bridges: The Automated car should decrease its speed while crossing bridges, because accident may happen
Cars: The Automated car should watch the car around it. It should watch which side they are coming , including the distance how far they are from it.

Actuator:

Speedometer: instrument that indicates the speed of a vehicle, usually combined with a device known as an odometer that records the distance traveled
Steering wheel: used to direct car in desired directions.
Accelerator, gear: To increase or decrease speed of the car.

Accelerator: pedal controls the amount of gas being fed into the engine and thereby controls the speed of the vehicle

Mirrors : allow you to observe what is happening around your car. They are your most important visual driving aid, and are vital for safe driving. Their purpose is to let you know what is happening behind, which is just as important as knowing what is happening in front

Sensor:

Cameras: The devices are conveniently placed to observe the whole of the road ahead, therefore any accident which happen to see or be involved in is going to be recorded

Odometer: a device that is used for measuring the distance traveled by a vehicle. The odometer is usually situated in the vehicle's dashboard.

GPS: his tracking device for vehicles provides information about its exact location so that it can report details on where a vehicle is.

PEAS Description for Subject Microsoft Cortana :

Performance:

alert users to upcoming meetings:should send the user notification about meetings and big events

answer user questions: Should answer the questions asked by the user fast and correctly

search the user's PC and the web: Able to search the users personal computer location and the web contents.

set reminders:should set Alarms correctly and ring it on the right time without any mistake

show users the weather: should show the user of the day. Because it helps the user to put on a close the matches with the weather.

open applications: open application with few clicks

Environment:

Windows 10 and 11: Cortana works in windows 10 and 11.It can be enabled in windows operating system in the settings: by clicking Talk **to** Cortana and Under Hey Cortana, by switching the toggle to **On**.

Microsoft edge: Ms edge is an internet explorer that uses cortana for voice searches.

Actuator:

Microphone: Cortana uses microphone to record voices and to answer back.

Keyboard: user can use keyboard to talk with and to ask questions.

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.

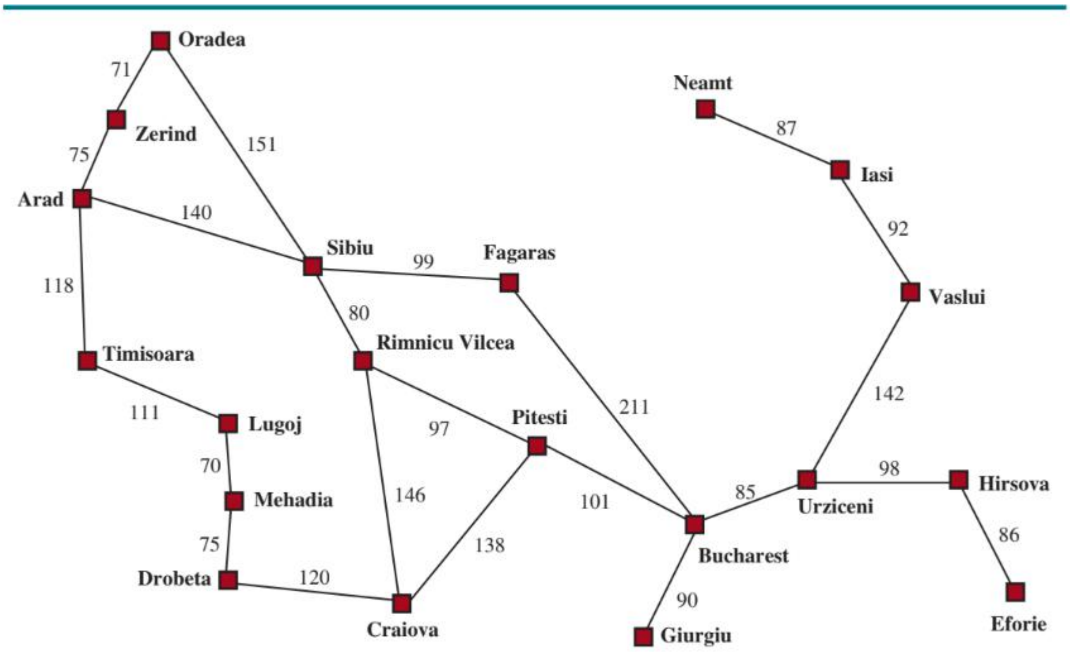


Figure 1. A simplified road map of part of Romania, with road distances in miles.

1 . **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

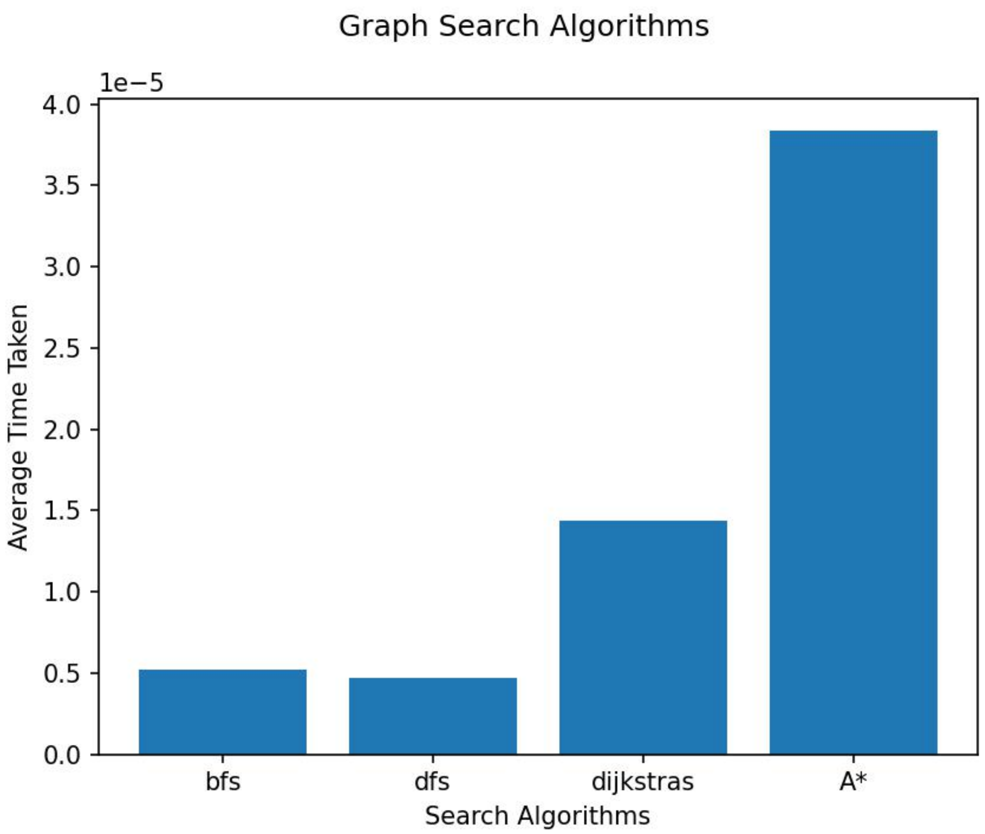
Algorithm	Time(seconds)
Breadth first search	7.285789553478887e-06
Deepth first search	7.474999945675097e-06
Dijkstras shortest path search	2.0752105291122508e-05
Astart search(A*)	6.919473686426308e-05

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



Graph 1. Average Time

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.

2 . **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

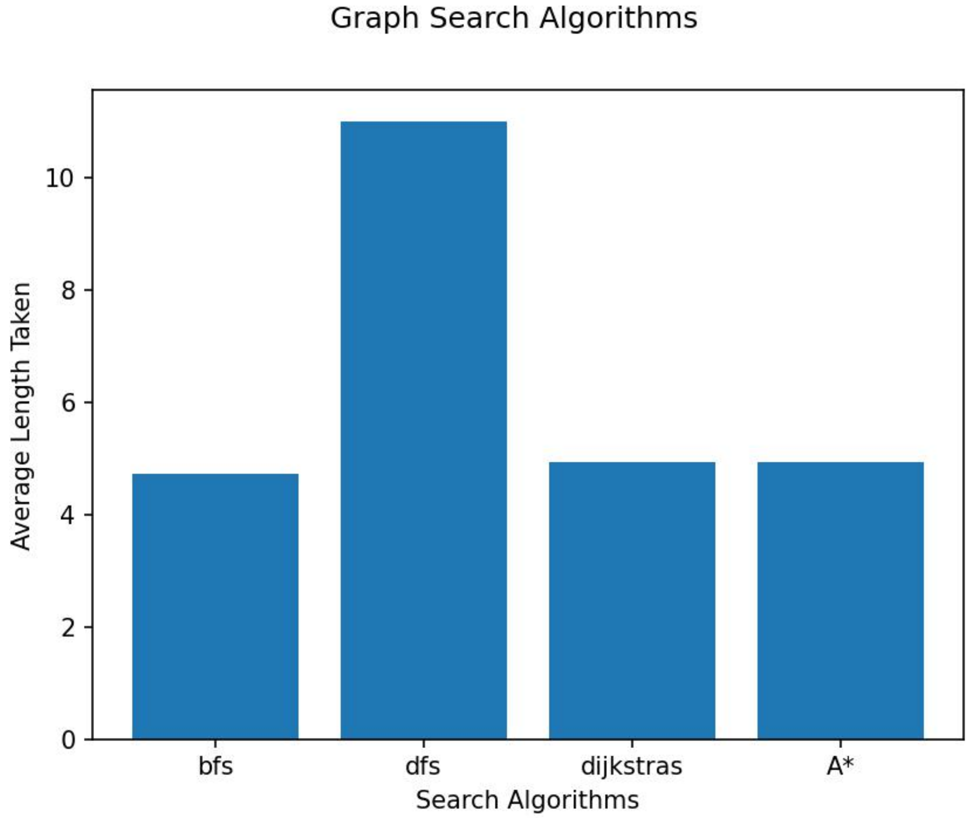
Algorithm	Length(no. Of nodes)
Breadth first search	4.721052631578948
Deepth first search	11.0
Dijkstras shortest path search	4.936842105263158
Astart search(A*)	4.942105263157894

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



Graph 2 . Average length

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.

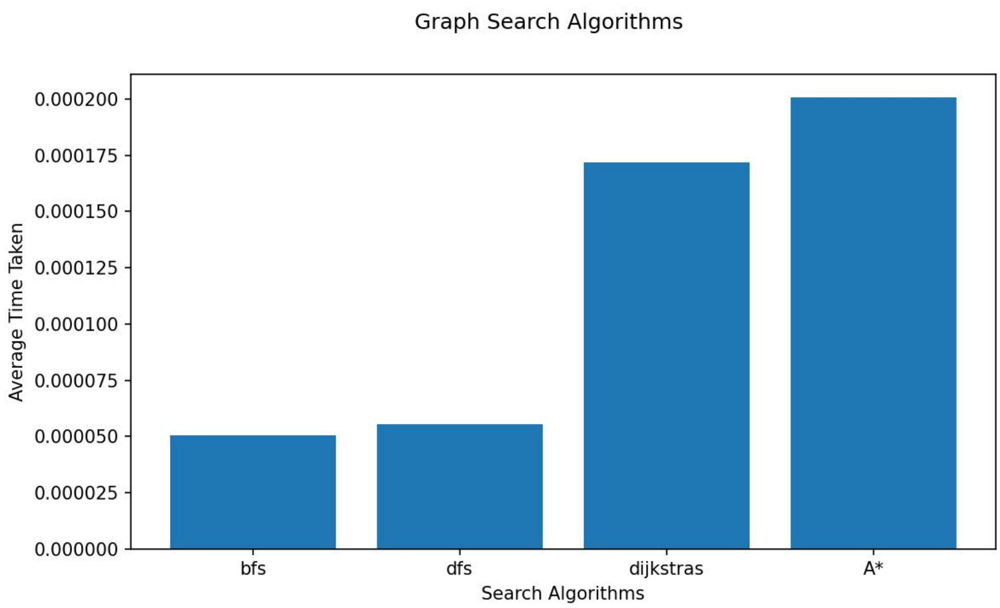
What would happen If we increase the number of Node?

Case 1: when the number of nodes increased to 40

Algorithm	Average Time(seconds)
Breadth first search	4.528868421881557 5e-05

Deepth first search	4.692000017568812e-05
Dijkstras shortest path search	0.00017254315789614344
Astart search(A*)	0.00025284657893184646

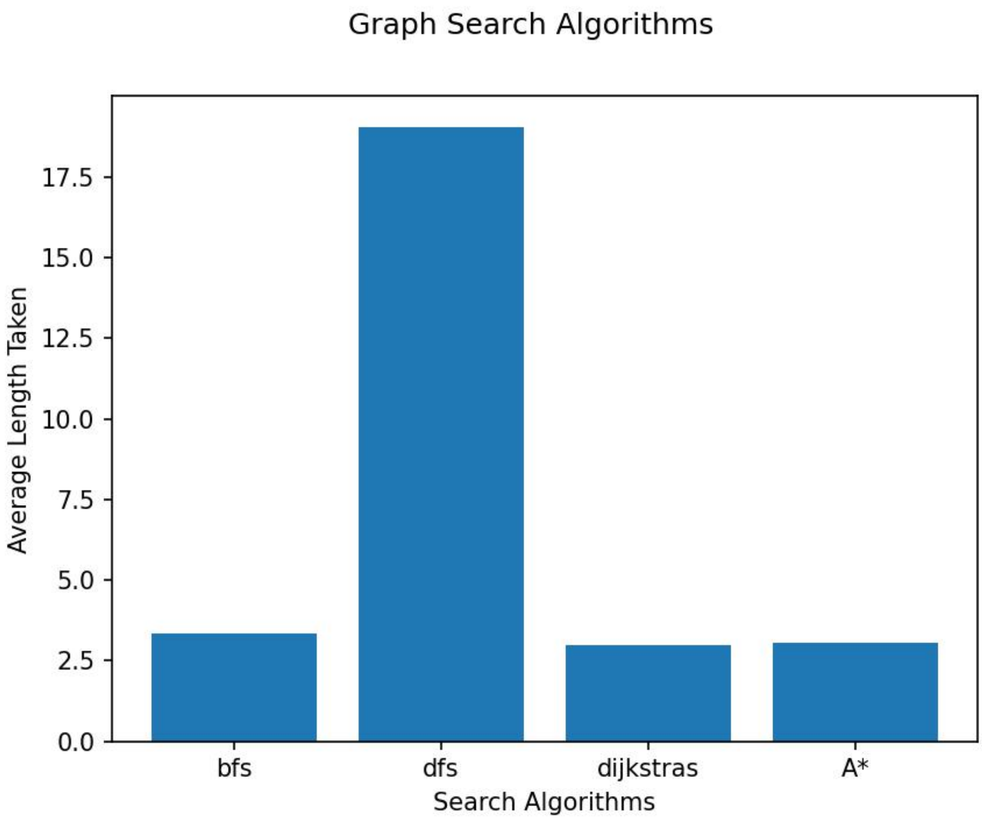
Table 6. Average Time



Graph 3. Average Time

Algorithm	Average Length(No. nodes)
Breadth first search	3.3421052631578947
Deepth first search	19.042105263157893
Dijkstras shortest path search	2.9842105263157896
Astart search(A*)	3.057894736842105

Table 7. Average Length

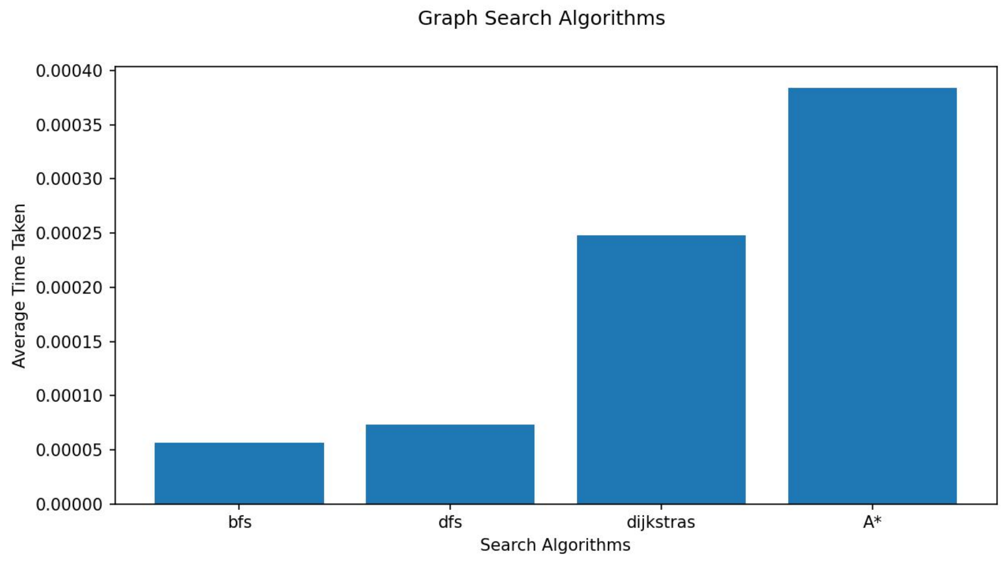


Graph 4. Average Length

Case 2: when the number of nodes increased to 60

Algorithm	Average Time(seconds)
Breadth first search	5.675657898552201e-05
Deepth first search	7.358131569114784e-05
Dijkstras shortest path search	0.0002478797371287855
Astart search(A*)	0.00038418078967017785

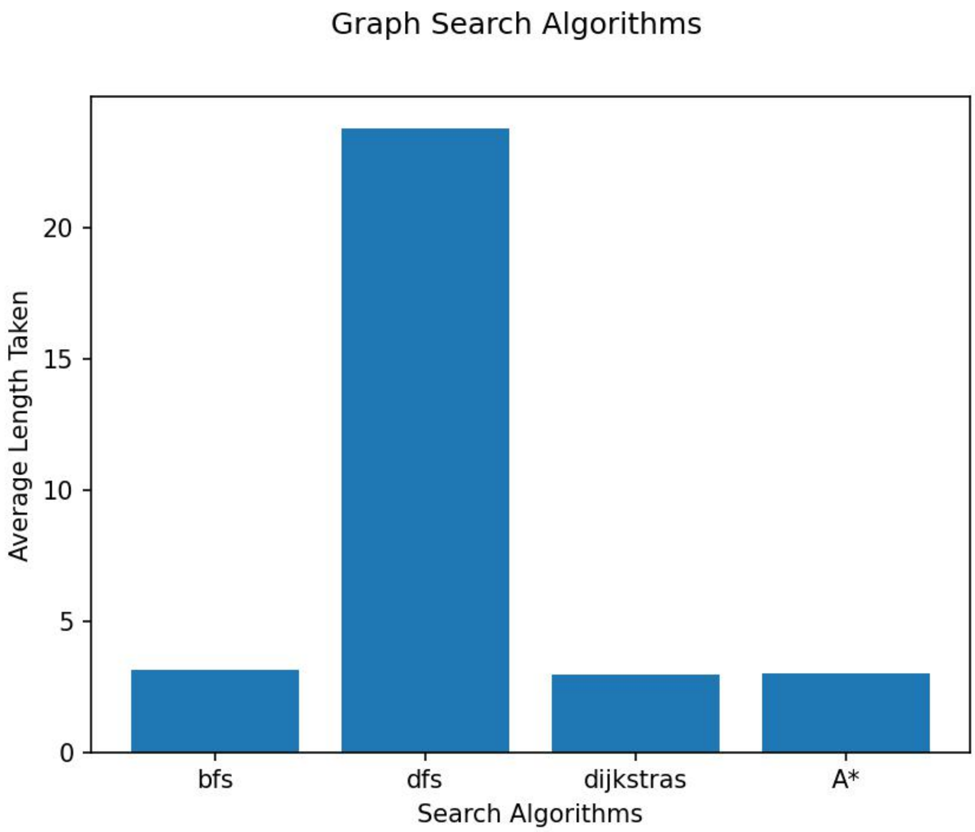
Table 8. Average Time



Graph 5. Average Time

Algorithm	Average Length(No. nodes)
Breadth first search	3.1473684210526316
Deepth first search	23.79736842105263
Dijkstras shortest path search	2.9473684210526314
Astart search(A*)	3.0

Table 9. Average Length

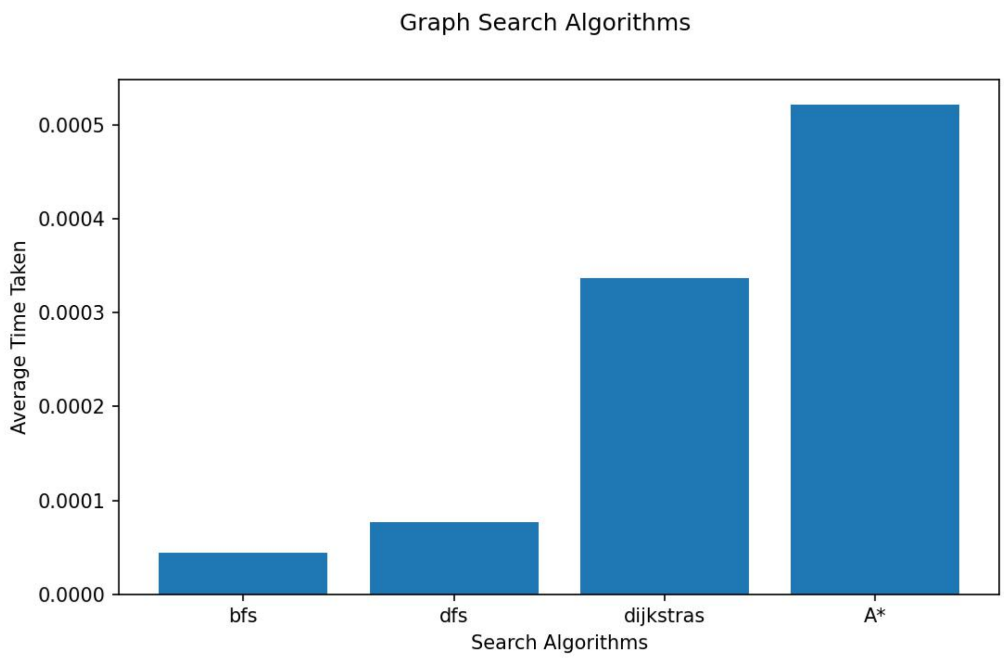


Graph 6. Average Length

Case 3: when the number of nodes increased to 80

Algorithm	Average Time(seconds)
Breadth first search	2.3895000080259420e-05
Deepth first search	7.689605278612458e-05
Dijkstras shortest path search	0.00033713763169591974
Astart search(A*)	0.00052176157914800057

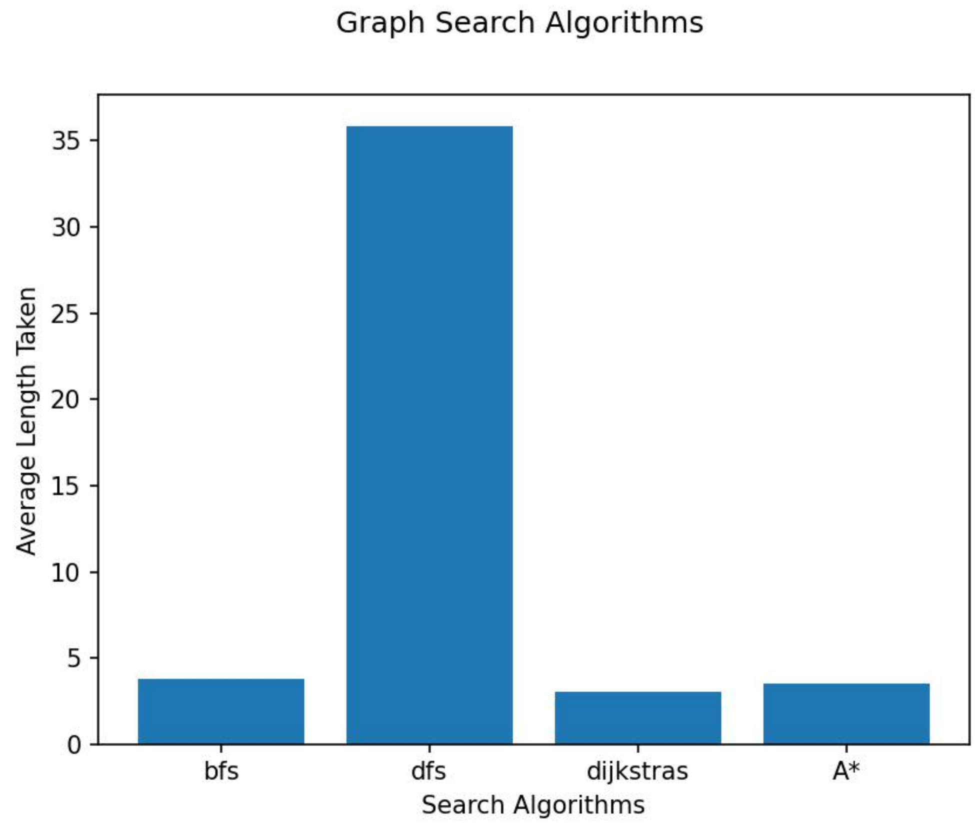
Table 10. Average Time



Graph 7. Average Time

Algorithm	Average Length(No. nodes)
Breadth first search	3.768421052631579
Deepth first search	35.86052631578947
Dijkstras shortest path search	3.0157894736842104
Astart search(A*)	3.4894736842105263

Table 11. Average Length

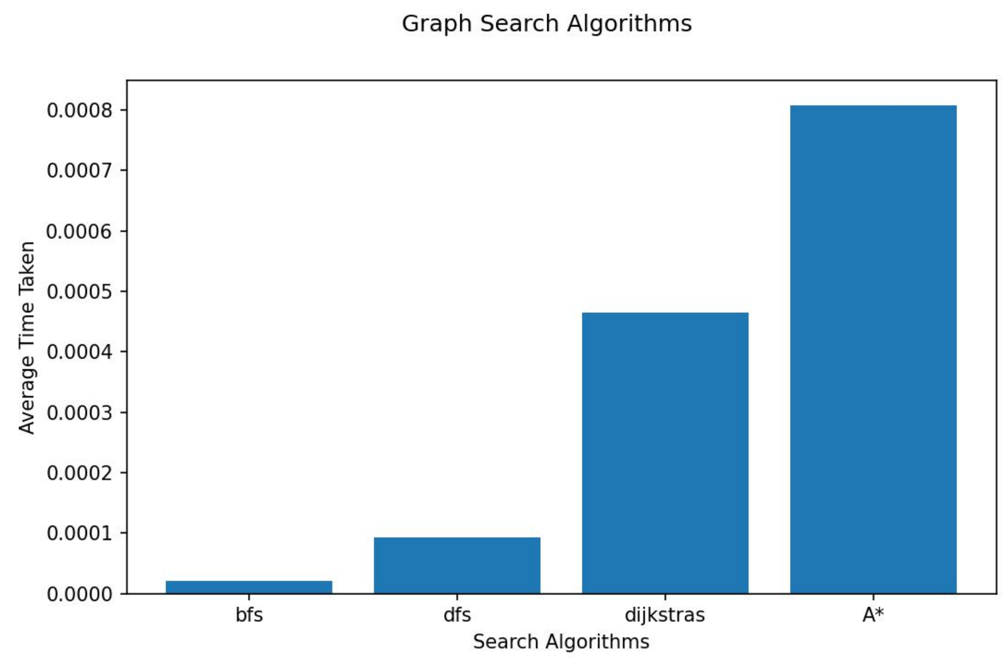


Graph 8. Average Length

Case 4: when the number of nodes increased to 100

Algorithm	Average Time(seconds)
Breadth first search	2.0793421055331188e-05
Deepth first search	9.34731578079089e-05
Dijkstras shortest path search	0.0004656115789609765
Astart search(A*)	0.0008082173682563926

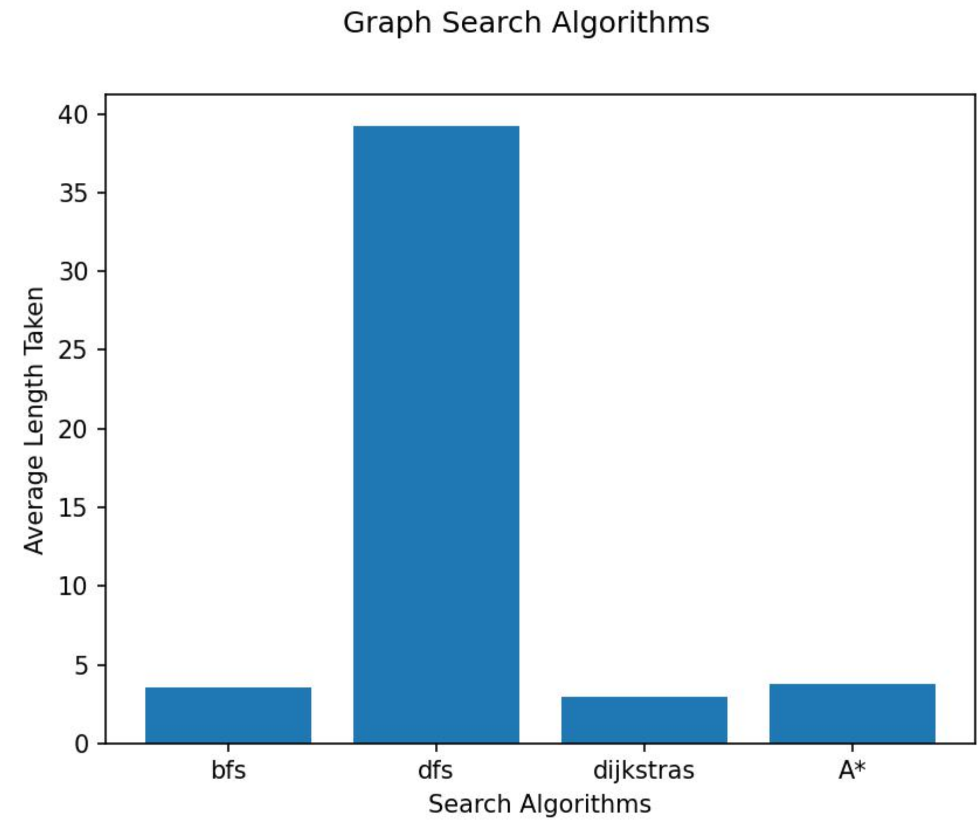
Table 12. Average Time



Graph 9. Average Time

Algorithm	Average Length(No. nodes)
Breadth first search	3.5157894736842104
Deepth first search	39.23947368421052
Dijkstras shortest path search	2.9947368421052634
Astart search(A*)	3.8026315789473686

Table 13. Average Length



Graph 10. Average Length

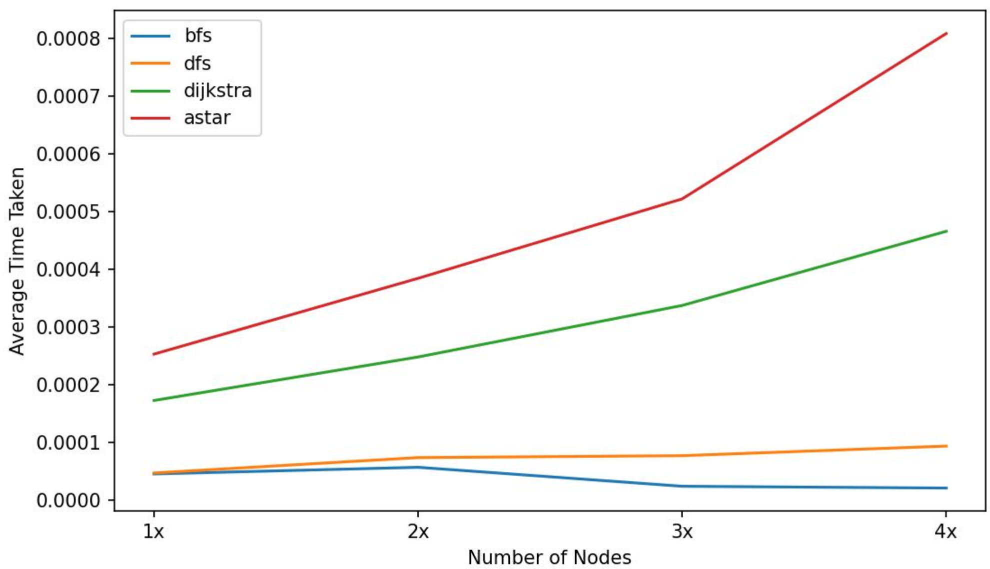


Figure 2. Average Time

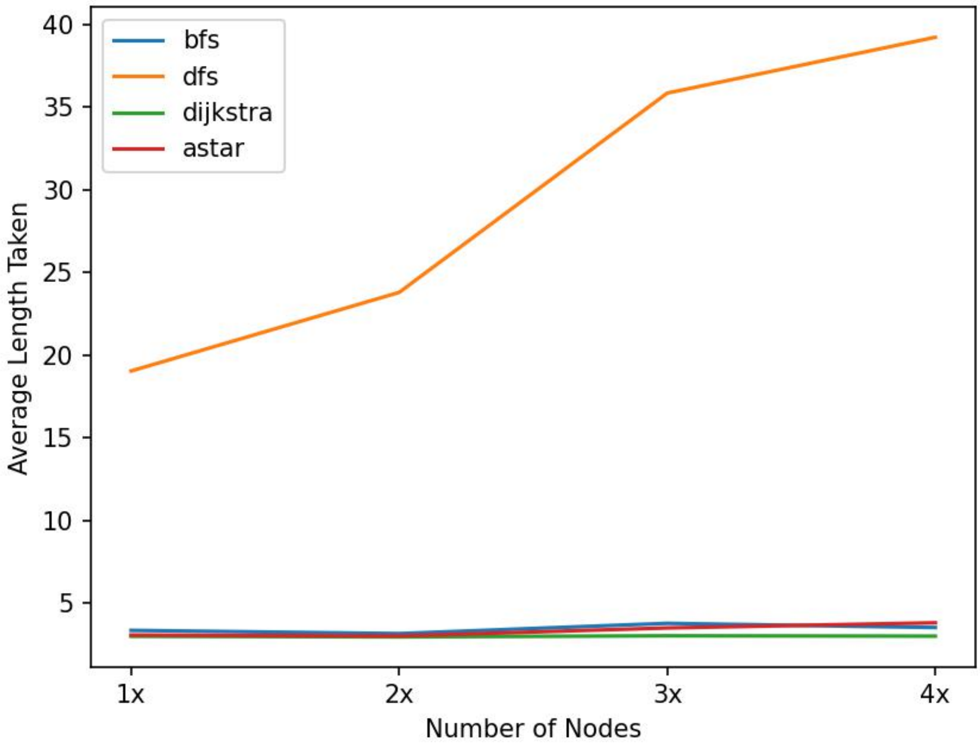


Figure 3. Average Length

Observation:

Average Time comparison:

Breadth first search : As the number of nodes increase the breadth search algorithm showed a decrease in time. The reason behind the this scenario is the nodes are made to be connected randomly. So breadth first search algorithm tries to find the shortest path blindly.At this time their may be disconnected nodes i.e nodes that are isolated. Then it only take the time it reaches dead end.

Depth first search algorithm showed slit increase in average time when the number of nodes increase. This is because this algorithm has to search as deep as possible blindly.

Dijkstras Shortest path algorithm: This algorithm showed a large increase in time as compared to bfs and dfs. The reason is, there will be much neighbors for a certain node to compare their weight and choose among them.

A* search: This algorithm takes the longest average time from all algorithms in this scenario. This algorithm tries to find the best utility and takes too much computation when the number of nodes increase to find the optimal solution.

Average Length Comparison:

Breadth first search: This algorithm showed slit difference in average length as the number of node increase. It tries to find the shortest path by tracking the parent in each destination node. It is uninformed and only see the parent node. So it takes the shortest average length.

Depth first search: As the number of nodes increase the average length of this algorithm increase. It search the node in solution as deep as possible. So when the number of nodes increase the depth also increase. That is why dfs takes large amount of average length.because there is a lot of back tracking.

Dikstras shortest path algorithm: This Algorithm also takes the shortest average length and showed only a slit increase. Because this algorithm only search for the shortest path

A* search: This algorithm also showed small change as the number of nodes increase as dijkstras and bfs. The reason is it only searches the shortest path and also there might be disconnected nodes since the nodes connected randomly.