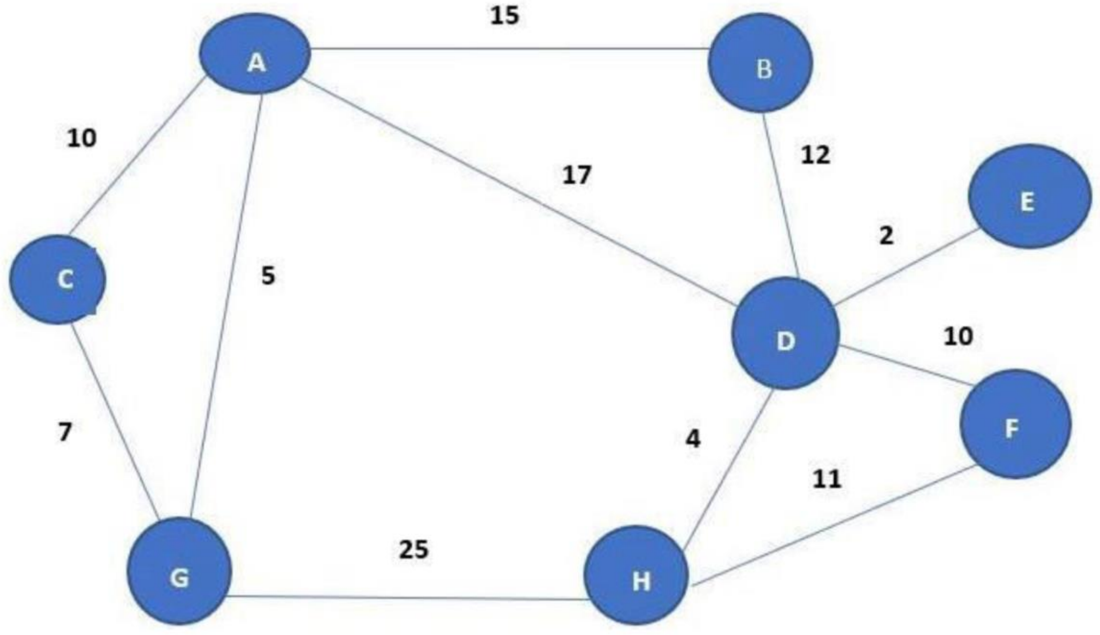
Artificial and Computational Intelligence

Developed By – Sushil Kumar

Problem statement

Given below are the shared and distributed servers placed at various locations in your city. Now imagine it to be in a crisis and you have been asked to compromise some communication lines and keep a minimum number of connections to keep the network up and going. Find the sub set of connections for your agent to go un-interrupted. Use the following algorithms to find the minimum network connections possible and help the network agent.



Explain the PEAS (Performance measure, Environment, Actuator, Sensor) for agent

Performance Measure:

The performance measure in this problem can be defined as the minimum number of network connections required to keep the network up and running while minimizing the total transmission cost. The goal is to find a subset of connections that ensures uninterrupted communication while minimizing the overall cost.

Environment:

The environment in this problem is a network infrastructure consisting of shared and distributed servers placed at various locations in the city. The nodes represent the servers, and the edges represent the communication lines between them. The edge costs depict the approximate transmission cost between each pair of nodes.

Actuators:

The actuators in this problem are the network connections between the nodes. The objective is to select a subset of connections to keep the network functioning with minimum interruption.

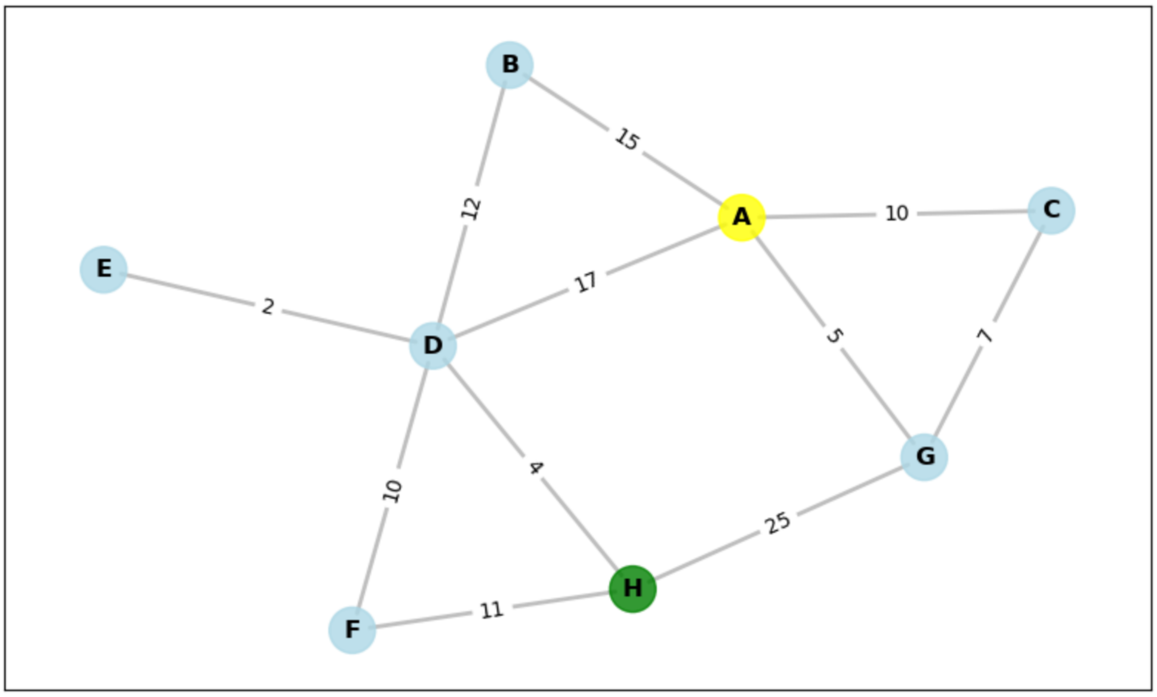
Sensors:

The sensors in this problem provide information about the network structure, including the nodes and the edges (communication lines) between them. The edge costs provide information about the transmission cost between each pair of nodes.
Code and output are shown in the Jupyter notebook file submitted alongside this document.

1. Use Breadth First Search and Recursive Best First Search and implement the algorithms in PYTHON. The program should be able to take-in start and goal nodes dynamically from the user at run time. Compare to interpret the results in terms of the algorithm working, performance & shortest path if obtained relevant to the given problem.

```
Enter the start node: A
Enter the goal node: H
```

Graph Representation



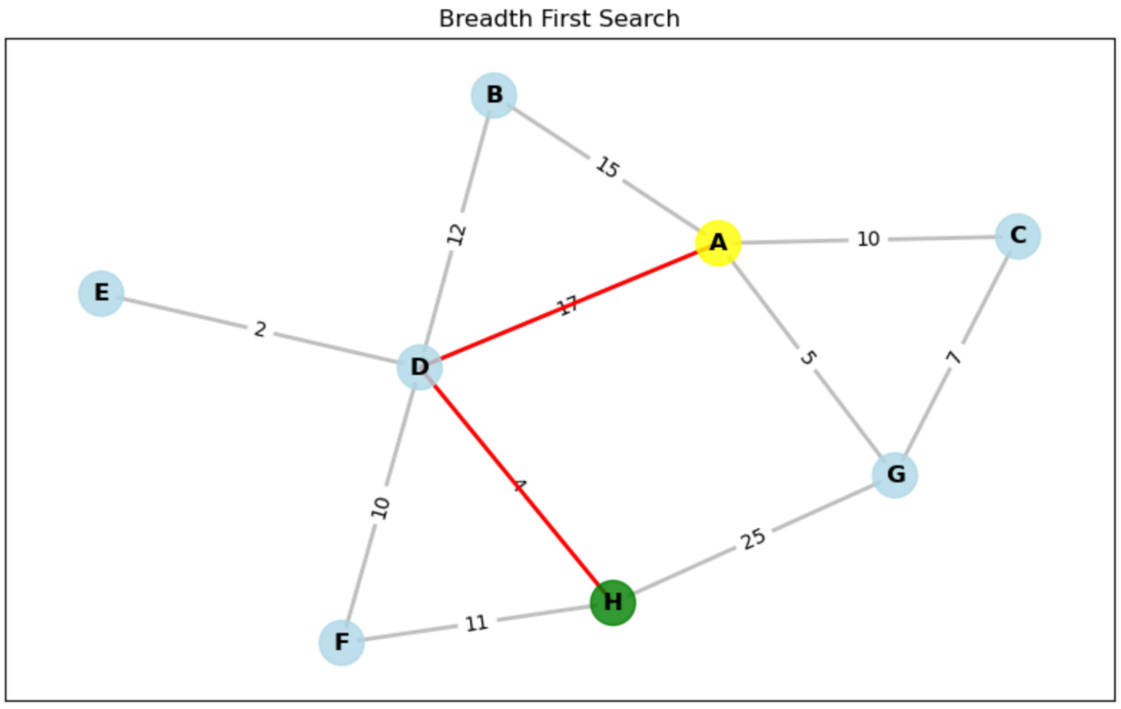
Breadth First Search (BFS)

In this example: start node: A, Goal Node : H

Iterations	Open List	Closed List	Goal Test
1		(A)	Fail
2	(C, D, G)	(A), (A,B)	Fail
3	(C,D,G) (D,G)	(A) (A, B) (A, C)	Fail
4	(C, D, G) (D, G) (G)	(A) (A, B) (A, C) (A, D)	Fail
5	(C, D, G) (D, G) (G) (E, F, H)	(A) (A, B) (A, C) (A, D) (A, G)	Fail
6	(C, D, G) (D, G) (G) (E, F, H) (F, H)	(A) (A, B) (A, C) (A, D) (A, G) (A, D, E)	Fail
7	(C, D, G) (D, G) (G) (E, F, H) (F, H), (H)	(A) (A, B) (A, C) (A, D) (A, G) (A, D, E) (A, D, F)	Fail
8		(A) (A, B) (A, C) (A, D) (A, G) (A, D, E) (A, D, F) (A, D, H)	Pass

Below is the screenshot of the path obtained via BFS algorithm

```
Path Found: ['A', 'D', 'H']
Path Cost: 21
```



Recursive Best First Search (RBFS)

Recursive Best First Search (RBFS) is an informed search algorithm used to find the optimal path from a start node to a goal node. It is a variation of the Best First Search algorithm, which uses a heuristic function to guide the search towards the most promising nodes.

In RBFS, the algorithm recursively explores nodes with the lowest estimated cost, combining the advantages of depth-first search and best-first search. At each step, RBFS chooses the node with the lowest f-value, where f(n) = g(n) + h(n), where g(n) is the cost of the path from the start node to node n, and h(n) is the estimated cost from node n to the goal node. The search continues until the goal node is reached or the f-value exceeds a predefined threshold.

We have used priority queue data structure for RBFS. RBFS may not guarantee the shortest path to the goal node, but it can be more memory-efficient compared to breadth-first search and other uninformed search algorithms, as it only keeps track of the nodes along the current path. However, it is essential to have an admissible heuristic function to ensure that RBFS finds an optimal solution if one exists.

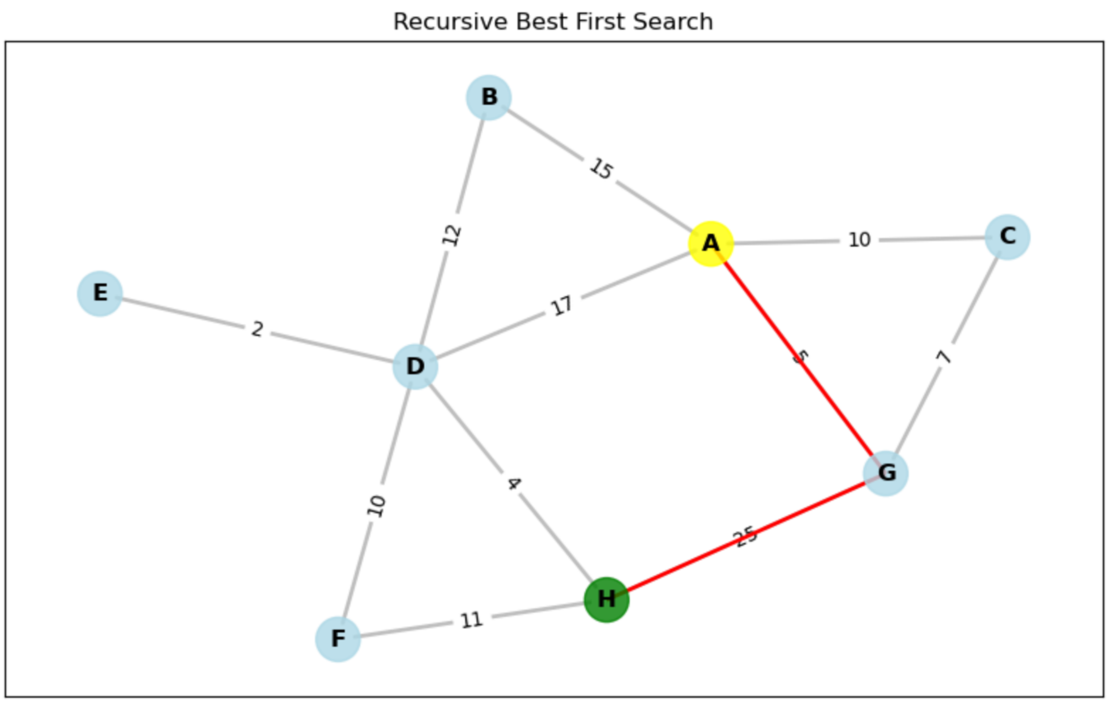
As per instructions in the problem statement, we have calculated the heuristic value of each node for a given node, based on the all possible paths from the start node.

Based on this logic, heuristic values calculated in the program are (goal node : H)

```
Heuristic Values

A: 35.0
B: 46.25
C: 43.6
D: 42.833333333333336
E: 44.833333333333336
F: 49.5
G: 43.2222222222222
```

```
Path found: A -> G -> H
Path cost: 30
```



BFS space complexity is higher as compared to RBFS. BFS (Breadth First Search) explores all nodes at a given depth level before moving to the next depth level, ensuring the shortest path but may have higher space complexity. RBFS (Recursive Best First Search) explores nodes based on heuristic values, reducing memory consumption but may not guarantee the shortest path.

Time & space complexity of BFS function in one run of the program

```
Time Complexity: 0.00098729 seconds
Space Complexity: 576 bytes
```

Similarly, time and space complexity of RBFS function in one run of the program is

```
Time Complexity: 0.00236273 seconds
Space Complexity: 456 bytes
```

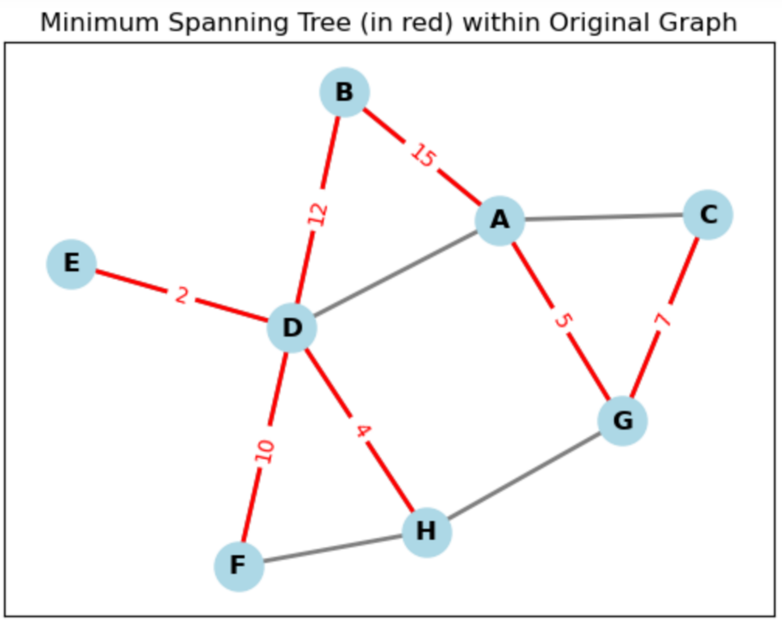
As per the python implementation, space complexity of BFS is 576 while for RBFS it's 456. So RBFS is memory wise more efficient.

Time and space complexity may vary depending on the specific path and heuristics used.

In summary, BFS is suitable for small graphs and situations where finding an optimal path quickly is not a priority. RBFS is more appropriate for larger graphs or situations where an optimal path is essential, especially if a good evaluation function and heuristic are available.

2. Print the minimum connections that keeps the whole network up and running.

Here we have used MST (Kruskal) algorithm to find the minimum connections needed to keep the network up and running. Also given the cost of the minimum tree.



```
Minimum Spanning Tree:
D -- E (Cost: 2)
D -- H (Cost: 4)
A -- G (Cost: 5)
C -- G (Cost: 7)
D -- F (Cost: 10)
B -- D (Cost: 12)
A -- B (Cost: 15)
Minimum Spanning Tree Cost: 55
```