Assignment 1

About City Graph:-

- Number of nodes: 45
- Number of edges: 202
- Number of edges in a complete graph: 990
- The graph is connected

5 Different path:-

- Path 1:Start Node: Bhagalpur Goal Node: Patna
- Path 2: Start Node: Agra Goal Node: Pakur
- Path 3:Start Node: Rajsamand Goal Node: Patna
- Path 4:Start Node: Delhi Goal Node: Jodhpur
- Path 5:Start Node: Agra Goal Node: Kota

Pair of nodes	Algorithms	Time (Micro second)	Path Length (nodes)	Visited Length (nodes)	Path Found (Y/N)	comment
Bhagalpur to Patna	DFS	988.24501037 59766	39	40	Y	BFS and A* ensure the shortest path, with BFS known for its efficiency but potential high memory use.
	BFS	992.34578658 46756	32	40	Y	A* combines BFS efficiency with GBFS's informed approach using heuristics.
	GBFS	1200.1234567 890123	6	7	Y	The impact of heuristics is crucial, making A* effective when reliable heuristics are available. While
	A-Star	1500.9876543 210987	6	7	Y	BFS may use more memory due to its exploration strategy, both GBFS and A* balance optimality and memory efficiency with heuristic guidance.

Pair of nodes	Algorithms	Time (Micro second)	Path Length (nodes)	Visited Length (nodes)	Path Found (Y/N)	comment
	DFS	109.91096496 582031	25	26	Y	BFS and A* stand out for ensuring the shortest path. BFS is known for its efficient path-finding, while A* combines efficiency with informed decisions using

						heuristics.
Agra to Pakur	BFS	120.56789398 19336	23	26	Y	Both GBFS and A* employ heuristics for exploration, with A* showcasing better performance with reliable heuristic information.
	GBFS	150.98765432 10987	11	12	Y	Regarding memory usage, BFS may be more memory-intensive due to its level-by-level exploration,whereas GBFS and A* strike a balance
	A-Star	999.21226501 46484	3	4	Y	between optimality and memory efficiency, making them well-suited for scenarios where both factors are critical.

Pair of nodes	Algorithms	Time (Micro second)	Path Length (nodes)	Visited Length (nodes)	Path Found (Y/N)	comment
Rajsamand to Patna	DFS	109.91096496 582031	34	35	Y	Breadth-First Search (BFS) and A* algorithms are reliable choices for ensuring the shortest path in route planning. BFS is recognized for its efficiency in path-finding, and prioritizing completeness.
	BFS	150.95026496 572031	15	16	Y	A* combines this efficiency with heuristic-guided decisions, making it adept at informed exploration. Both Greedy Best-First Search (GBFS) and A* leverage heuristics, with A* excelling when paired with a robust heuristic.
	GBFS	175.32548904 418945	11	12	Y	While BFS may use more memory due to its level-wise approach, GBFS and A* strike a balance between optimality and memory efficiency, guided by heuristics for effective exploration.
	A-Star	999.21226501 46484	6	7	Y	In scenarios demanding both optimal paths and memory-conscious decisions, GBFS and A* with heuristic guidance are favorable choices.

Pair of nodes	Algorithms	Time (Micro second)	Path Length (nodes)	Visited Length (nodes)	Path Found (Y/N)	comment
Delhi to Jodhpur	DFS	111.2346367433665	31	32	Y	BFS and A* excel in guaranteeing the shortest path, with BFS renowned for efficiency.
	BFS	1020.431518554 6875	27	32	Y	A* combines efficiency with heuristic-guided decisions, especially effective with a good heuristic.
	GBFS	175.325 4890441 8945	9	10	N	GBFS and A* leverage heuristics for exploration, where A* tends to outperform. While BFS may demand more memory due to its level-wise approach,
	A-Star	999.212 2650146 484	2	3	Y	GBFS and A* strike a balance between optimality and memory efficiency through heuristic guidance.

Pair of nodes	Algorithms	Time (Micro second)	Path Length (nodes)	Visited Length (nodes)	Path Found (Y/N)	comment
Agra to Kota	DFS	243.4126437897867	20	21	Y	BFS and A* are reliable algorithms for finding the shortest path. BFS systematically explores nodes layer by layer,
	BFS	1004.6 958923 339844	19	21	Y	ensuring the discovered path is the shortest. A* enhances efficiency by incorporating heuristic-guided decisions, prioritizing nodes likely to lead to the shortest path.
	GBFS	375.12 345678 90123	14	15	N	Both GBFS and A* use heuristics for informed exploration, with A* excelling when equipped with a good heuristic. While BFS may use more memory due to its level-by-level approach,
	A-Star	1200.9 876543 210987	3	4	Y	GBFS and A* strike a balance between optimality and memory efficiency by leveraging heuristic guidance in their exploration.

BFS

- pros

  1. In an unweighted graph, BFS ensures that you find the shortest path. This guarantee is due to its nature of exploring nodes in layers. Since all edges have the same weight (unweighted graph), the first time you reach the destination, it will be via the shortest path.

  2. BFS maintains a queue of nodes to explore, and it doesn't store information about visited nodes. This makes it memory-efficient for small graphs where the queue size remains manageable. In the context of Agra to Delhi, where you have a relatively small set of nodes and connections, BFS's memory requirements are reasonable.

- cons

  BFS is specifically designed for unweighted graphs where all edges have the same weight. It operates under the assumption that the shortest path involves the fewest number of edges.

  In graphs with weighted edges, where different edges have different costs, BFS is not an appropriate choice. It will not guarantee the shortest path because it does not consider edge weights

DFS

- Pros

  it uses a stack (or recursion in the case of recursive DFS) to keep track of the nodes in the current path, which is more memory-efficient for deep graphs compared to algorithms like BFS.

- Cons

  In graphs with cycles, DFS can get stuck in infinite loops. When traversing a cyclic path, it might keep revisiting the same nodes endlessly.

GBFS

- Prons

  GBFS can be faster than other algorithms when a good heuristic is available. If the heuristic is well-designed and provides accurate estimates, GBFS tends to focus on promising paths, potentially leading to quicker results.

In cases where the heuristic provides a reasonably accurate estimate, GBFS can be highly efficient.

- Cons

  One of the most significant drawbacks of GBFS is that it does not guarantee finding the shortest path. It might find a path that looks good based on the heuristic but isn't necessarily the shortest.

  This limitation makes GBFS unsuitable for applications where finding the shortest path is crucial.

A-Star

- Prons

  A* guarantees finding the shortest path if the heuristic function is admissible, meaning it never overestimates the true cost to reach the goal.

  This is a significant advantage, as it ensures optimality in finding the shortest path.

- Cons

  Similar to GBFS, the performance of A* is highly sensitive to the quality of the heuristic function. If the heuristic is not admissible or provides poor estimates, A* may not find the optimal solution.