Evaluation:

Q-1) Which algorithm is fastest?
A-1) Running the algo on 5 distinct start and end point, we get

Total Iteration count:

Start Point	End Point	DFS	BFS	GBFS
1,1	4,4	**4**	16	**4**
1,20	20,1	212	270	**64**
13,6	7,6	158	247	**41**
1,1	20,20	44	264	**32**
1,20	20,20	**20**	180	**20**
3, 15	20, 8	231	225	**58**

Time Complexity
DFS => O(branching factor ^ depth of graph)
BFS => O(branching factor ^ depth of node)
GBFS => O(branching factor ^ depth of node)
GBFS is fastest in the above scenario. Since Greedy Best-first search is a greedy method. Greedy methods maximize short-term advantage without worrying about long-term consequences. If a straight path is available towards target, GBFS runs fastest else GBFS might not give fastest solution.

Q-2) Which is most memory efficient?
A-2) Running the algo on 5 distinct start and end point, we get

Max Frontier Size

Start Point	End Point	DFS	BFS	GBFS
1,1	4,4	**7**	9	**7**
1,20	20,1	99	**22**	51
13,6	7,6	91	**22**	35
1,1	20,20	43	**21**	36
1,20	20,20	20	**22**	20
3, 15	20, 8	91	**20**	37

Space Complexity:
Surprisingly BFS consumes lowest memory and in some cases DFS and GBFS are good. But these cases are restricted to this problem. Generally, BFS consumes more memory as compared to DFS since it expands node on each level whereas DFS stores the length of root to leaf node path at maximum.

Q-3) Which visits the fewest vertices?
A-3) Running the algo on 5 distinct start and end point, we get

Vertices Visited

Start Point	End Point	DFS	BFS	GBFS
1,1	4,4	10	23	**10**
1,20	20,1	262	271	**114**
13,6	7,6	232	263	**75**
1,1	20,20	86	271	**67**

1,20	20,20	39	190	**39**
3, 15	20, 8	269	229	**94**

GBFS visits fewest vertices as it shoots towards the target in a straight line. There could be cases where it could be obstructed. If node is present at a close distance from source, BFS might cover few vertices as compared to DFS but if node is far from source, both DFS and BFS have to cover large number of vertices. There could be a chance where DFS might reach target with few vertices visited as it shoots towards the depth of graph.

Q-4) Which generates shortest path length?(Is any of them optimal?)
A-4)Running the algo on 5 distinct start and end point, we get

Path Length

Start Point	End Point	DFS	BFS	GBFS
1,1	4,4	3	**3**	3
1,20	20,1	46	**33**	47
13,6	7,6	68	**28**	28
1,1	20,20	40	**28**	31
1,20	20,20	19	**19**	19
3, 15	20, 8	32	**22**	24

BFS will give you the shortest path length as it traverses level by level provided all path cost are same. **BFS is not optimal** in general but in the above case, since all path length are same, it is optimal in the above scenario. **GBFS is also not optimal**. GBFS also sometimes gives equal to BFS unless it is obstructed by some barrier. In that case, it diverges from the path and has to cover extra path. **DFS is not optimal.**

Q-5)Are the performance differences what you expected based on the theoretical complexity analysis?
A-5) Some parameters were different than expected.

- **Time Complexity** : BFS turned out to perform worse as compared to DFS as we saw in Table 1.
- **Space Complexity** : DFS is supposed to take less memory as compared to BFS but we saw the opposite in Table 2.

Q-6)Does BFS always find the shortest path? Does GBFS always go "straight" to the goal, or are there cases where it gets side-tracked?
A-6) **BFS always find the shortest path if all edge cost are same and positive**. GBFS does always go straight to the goal but it does get side-tracked when there is an obstruction in between. Take an example from (1,20) to (20,1) where GBFS shoots straight to the goal and where it got obstructed by “T”.