Performance Metrics

Points

(1,1) to (20,20)	Total iterations	Max Frontier size	Vertices visited	Path Length
BFS	264	**21**	271/275	**28**
DFS	44	43	86/275	40
GBFS	**32**	36	**67/275**	31

(13,6) to (7,6)	Total iterations	Max Frontier size	Vertices visited	Path Length
BFS	247	**22**	263/275	**28**
DFS	158	91	232/275	68
GBFS	**40**	35	**74/275**	**28**

(1,1) to (4,4)	Total iterations	Max Frontier size	Vertices visited	Path Length
BFS	16	9	23/275	**3**
DFS	**4**	**7**	**10/275**	**3**
GBFS	**4**	7	**10/275**	**3**

(8,12) to (17,8)	Total iterations	Max Frontier size	Vertices visited	Path Length
BFS	248	**23**	254/275	**26**
DFS	**47**	40	**86/275**	37
GBFS	67	43	109/275	**26**

(20,10) to (3,8)	Total iterations	Max Frontier size	Vertices visited	Path Length
BFS	247	**21**	258/275	**25**
DFS	57	51	107/275	40
GBFS	**42**	33	**74/275**	**25**

(10,20) to (20,1)	Total iterations	Max Frontier size	Vertices visited	Path Length
BFS	254	**21**	262/275	**25**
DFS	216	96	262/275	39
GBFS	**30**	33	**62/275**	**25**

• Which algorithm is fastest (finds goal in fewest iterations)?

Greedy Best First Search (GBFS) was able to find the goal in least iterations as compared to BFS and DFS in most of the test cases. However, in the test case (8,12) to (17,8) Depth First Search proved faster. This is because in GBFS the shortest path, determined by our Heuristic function (Straight Line Distance), was blocked by an obstacle.

• Which is most memory efficient (smallest max frontier size)?

In 5 out of 6 tests performed Breadth First Search (BFS) proved more memory efficient, as it had the lowest 'Max Frontier Size'. However, in the test case (1,1) to (4,4) - GBFS and DFS had lower Max Frontier size than BFS.

• Which visits the fewest vertices?

Greedy Best First Search (GBFS) visits the lowest number of vertices. This is because GBFS visits the most promising vertex in each turn, as determined by the heuristic function. However, in the case of an obstacle in ((8,12) to (17,8)), GBFS surpassed DFS in the Vertices visited count.

• Which generates the shortest path length? (Is any of them optimal?)

BFS has given the shortest path length in all of the test cases. In fact, GBFS has also found the shortest path in 5/6 comparisons. However, in no test has GBFS found a shorter path length than BFS. Breadth First Search will give the optimal result in case of constant positive step costs. GBFS on the other hand is not always optimal but is more efficient.

• Are the performance differences what you expected based on the theoretical complexity analysis?

In theory DFS should be more memory efficient, as it analyzes one particular path and then backtracks while emptying its frontier. But, in 5/6 of our test cases BFS used a lower frontier size.

Moreover, BFS should search faster than DFS, but in our tests DFS required lesser iterations than BFS.

• Does BFS always find the shortest path? Does GBFS always go "straight" to the goal, or are there cases where it gets side-tracked?

In all of our test cases BFS was able to find the shortest path. This case is true for constant positive step costs. On the other hand GBFS was also able to find the shortest path in most of the test cases.

GBFS always tries to go “straight” to the goal. This causes problems when the straight path, as determined by heuristic function, is blocked by an obstacle. Thus, causing it to take a detour around it.