# Performance Analysis

The below analysis is based on the average values of the outputs for the following inputs:

1. 62 (4, 4) 0 (1, 1)
2. 255 (20, 1) 19 (1, 20)
3. 102 (7, 6) 163 (13, 6)
4. 274 (20, 20) 19 (1, 20)
5. 274 (20, 20) 0 (1, 1)

|  |  |  |  |
| --- | --- | --- | --- |
|  | **BFS** | **DFS** | **GBFS** |
| **Number of iterations** | 199.8 | 115.4 | 33 |
| **Maximum frontier size** | 19.6 | 68.2 | 32.4 |
| **Number of vertices visited** | 206.8 | 174.2 | 64.4 |
| **Path length** | 22.2 | 64.8 | 23.2 |

# Questions & Answers

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

GBFS

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

BFS

1. Which visits the fewest vertices?

GBFS

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

BFS, and it is optimal

1. Are there performance differences what you expected based on the theoretical complexity analysis?

Yes. BFS is expected to have an exponential Frontier size but it has the least here. One reasoning could be that since there can be a maximum of only O(20\*sqrt(2)) points equi-distant from any point, the effective branching factor is reduced, hence leading to a lower frontier size.

1. Does BFS always find the shortest path?

Yes

1. Does GBFS always go "straight" to the goal, or are there cases where it gets side-tracked?

Not always, for instance when finding a path between the points below ‘T’ (Input #3), the number of iterations is double the path-length returned.