**Blind Searches**

|  |  |  |  |
| --- | --- | --- | --- |
| **Problem**  **Search Method** | **Path**  **Start to Goal** | **Path Length** | **Number of Nodes Expanded** |
| HumansRobotsFerry  DFS | H on left:3  R on left:3  H on right:0  R on right:0  ferry is on the left.  H on left:2  R on left:2  H on right:1  R on right:1  ferry is on the right.  H on left:3  R on left:2  H on right:0  R on right:1  ferry is on the left.  H on left:0  R on left:2  H on right:3  R on right:1  ferry is on the right.  H on left:2  R on left:2  H on right:1  R on right:1  ferry is on the left.  H on left:1  R on left:1  H on right:2  R on right:2  ferry is on the right.  H on left:3  R on left:1  H on right:0  R on right:2  ferry is on the left.  H on left:0  R on left:1  H on right:3  R on right:2  ferry is on the right.  H on left:1  R on left:1  H on right:2  R on right:2  ferry is on the left.  H on left:0  R on left:0  H on right:3  R on right:3  ferry is on the right. | 9 | 10 |
| HumansRobotsFerry  BFS | H on left:3  R on left:3  H on right:0  R on right:0  ferry is on the left.  H on left:2  R on left:2  H on right:1  R on right:1  ferry is on the right.  H on left:3  R on left:2  H on right:0  R on right:1  ferry is on the left.  H on left:0  R on left:2  H on right:3  R on right:1  ferry is on the right.  H on left:2  R on left:2  H on right:1  R on right:1  ferry is on the left.  H on left:0  R on left:1  H on right:3  R on right:2  ferry is on the right.  H on left:1  R on left:1  H on right:2  R on right:2  ferry is on the left.  H on left:0  R on left:0  H on right:3  R on right:3  ferry is on the right. | 7 | 10 |
| Farmer\_Fox  DFS | Farmer is on the left  Fox is on the left  Chicken is on the left  Grain is on the left  Boat is on the left.  Farmer is on the right  Fox is on the left  Chicken is on the right  Grain is on the left  Boat is on the right.  Farmer is on the left  Fox is on the left  Chicken is on the right  Grain is on the left  Boat is on the left.  Farmer is on the right  Fox is on the right  Chicken is on the right  Grain is on the left  Boat is on the right.  Farmer is on the left  Fox is on the right  Chicken is on the left  Grain is on the left  Boat is on the left.  Farmer is on the right  Fox is on the right  Chicken is on the left  Grain is on the right  Boat is on the right.  Farmer is on the left  Fox is on the right  Chicken is on the left  Grain is on the right  Boat is on the left.  Farmer is on the right  Fox is on the right  Chicken is on the right  Grain is on the right  Boat is on the right. | 7 | 7 |
| Farmer\_Fox  BFS | Farmer is on the left  Fox is on the left  Chicken is on the left  Grain is on the left  Boat is on the left.  Farmer is on the right  Fox is on the left  Chicken is on the right  Grain is on the left  Boat is on the right.  Farmer is on the left  Fox is on the left  Chicken is on the right  Grain is on the left  Boat is on the left.  Farmer is on the right  Fox is on the right  Chicken is on the right  Grain is on the left  Boat is on the right.  Farmer is on the left  Fox is on the right  Chicken is on the left  Grain is on the left  Boat is on the left.  Farmer is on the right  Fox is on the right  Chicken is on the left  Grain is on the right  Boat is on the right.  Farmer is on the left  Fox is on the right  Chicken is on the left  Grain is on the right  Boat is on the left.  Farmer is on the right  Fox is on the right  Chicken is on the right  Grain is on the right  Boat is on the right. | 7 | 9 |
| TowersOfHanoi (4)  DFS | [[4, 3, 2, 1] ,[] ,[]]  [[4, 3, 2] ,[1] ,[]]  [[4, 3] ,[1] ,[2]]  [[4, 3, 1] ,[] ,[2]]  [[4, 3] ,[] ,[2, 1]]  [[4] ,[3] ,[2, 1]]  [[4, 1] ,[3] ,[2]]  [[4] ,[3, 1] ,[2]]  [[4, 2] ,[3, 1] ,[]]  [[4, 2, 1] ,[3] ,[]]  [[4, 2] ,[3] ,[1]]  [[4] ,[3, 2] ,[1]]  [[4, 1] ,[3, 2] ,[]]  [[4] ,[3, 2, 1] ,[]]  [[] ,[3, 2, 1] ,[4]]  [[1] ,[3, 2] ,[4]]  [[] ,[3, 2] ,[4, 1]]  [[2] ,[3] ,[4, 1]]  [[2, 1] ,[3] ,[4]]  [[2] ,[3, 1] ,[4]]  [[] ,[3, 1] ,[4, 2]]  [[1] ,[3] ,[4, 2]]  [[] ,[3] ,[4, 2, 1]]  [[3] ,[] ,[4, 2, 1]]  [[3, 1] ,[] ,[4, 2]]  [[3] ,[1] ,[4, 2]]  [[3, 2] ,[1] ,[4]]  [[3, 2, 1] ,[] ,[4]]  [[3, 2] ,[] ,[4, 1]]  [[3] ,[2] ,[4, 1]]  [[3, 1] ,[2] ,[4]]  [[3] ,[2, 1] ,[4]]  [[] ,[2, 1] ,[4, 3]]  [[1] ,[2] ,[4, 3]]  [[] ,[2] ,[4, 3, 1]]  [[2] ,[] ,[4, 3, 1]]  [[2, 1] ,[] ,[4, 3]]  [[2] ,[1] ,[4, 3]]  [[] ,[1] ,[4, 3, 2]]  [[1] ,[] ,[4, 3, 2]]  [[] ,[] ,[4, 3, 2, 1]] | 40 | 40 |
| TowersOfHanoi (4)  BFS | [[4, 3, 2, 1] ,[] ,[]]  [[4, 3, 2] ,[1] ,[]]  [[4, 3] ,[1] ,[2]]  [[4, 3] ,[] ,[2, 1]]  [[4] ,[3] ,[2, 1]]  [[4, 1] ,[3] ,[2]]  [[4, 1] ,[3, 2] ,[]]  [[4] ,[3, 2, 1] ,[]]  [[] ,[3, 2, 1] ,[4]]  [[] ,[3, 2] ,[4, 1]]  [[2] ,[3] ,[4, 1]]  [[2, 1] ,[3] ,[4]]  [[2, 1] ,[] ,[4, 3]]  [[2] ,[1] ,[4, 3]]  [[] ,[1] ,[4, 3, 2]]  [[] ,[] ,[4, 3, 2, 1]] | 15 | 70 |

Without paths for brevity,

|  |  |  |
| --- | --- | --- |
| **Problem**  **Search Method** | **Path Length** | **Number of Nodes Expanded** |
| HumansRobotsFerry  DFS | 9 | 10 |
| HumansRobotsFerry  BFS | 7 | 10 |
| Farmer\_Fox  DFS | 7 | 7 |
| Farmer\_Fox  BFS | 7 | 9 |
| TowersOfHanoi (4)  DFS | 40 | 40 |
| TowersOfHanoi (4)  BFS | 15 | 70 |

Regarding the Towers of Hanoi problem,

1. Why is the maximum length of the OPEN list more for one algorithm than the other?

The OPEN list is larger for breadth-first search because it stores each node and continues to build on the open list with successors until it either finds a solution on the current level or moves on to the next level of a graph. A node is only ever deleted from the open list in step 3. This makes it easy to backtrace a solution at the expense of increased memory use. Depth-first search, on the other hand, progresses deeper into a state space whenever possible, and only adds successors for single branches at a time, before no further descendants of a state can be found.

1. Why is the solution PATH length different for one algorithm from that of another?

In this case, there is more than one way to solve the puzzle. The path length is shorter for breadth-first search because it found a more efficient solution by considering level-by-level rather than expounding upon an early node. Depth-first search found a longer path because it progressed many levels into the puzzle without ever returning to a higher level to find a more efficient solution.