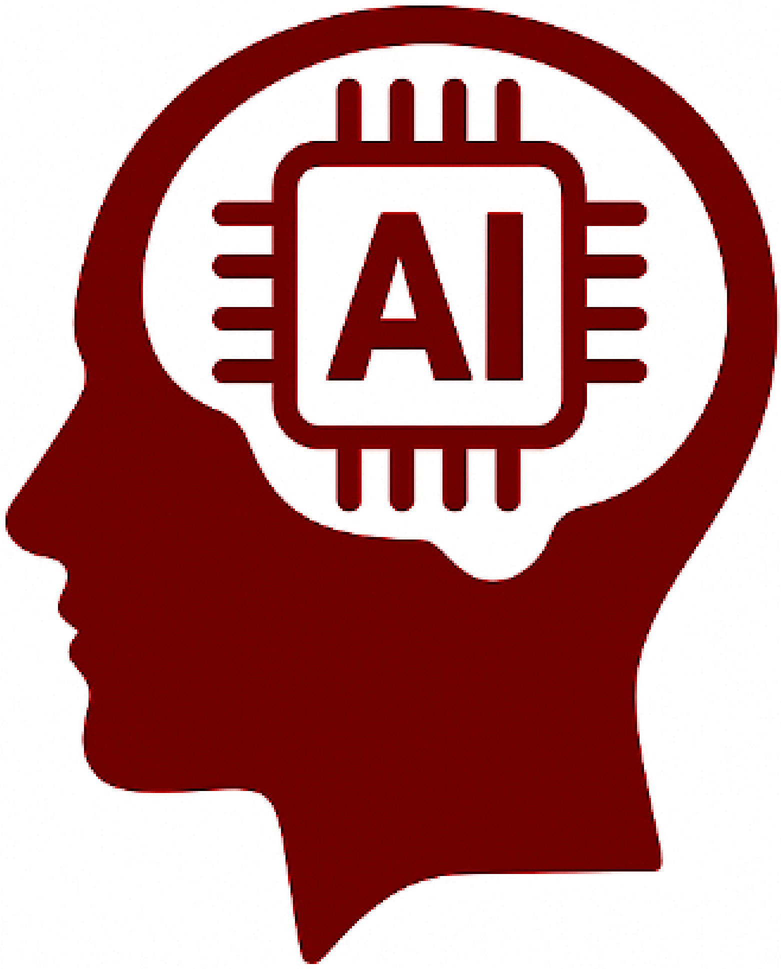
Practice 1 - Artificial Intelligence

Problem solving and searching: Uninformed search.



Author: Fernando Pastor Peralta (897113)

Delivery date: 06/10/2025

Group: Tuesday B, 4:00 PM - 7:00 PM

Computer Engineering - University of Zaragoza, Academic Year 2025-2026

Index

Introduction:

The objective of this lab is to become familiar with Java code for solving search problems. We will gain an in-depth understanding of how the algorithms used work and how to develop the environment required to solve these problems.

We will use different blind search techniques for the 8-puzzle problem and the Missionaries and Cannibals problem (which we will have to define). Afterwards, we will collect metrics from both problems, using the results of each algorithm, in order to draw conclusions about which algorithms are most suitable for each case.

The obtained metrics can be consulted in the appendices.

Methodology:

8-puzzle problem:

To begin with, an analysis and understanding of the code defining the problem was carried out: EightPuzzleBoard (state), EightPuzzleFunctionFactory (actions), and EightPuzzleGoalTest (goal state). Then, the demo (EightPuzzleDemo) was executed, which shows the execution of the 8-puzzle using different search algorithms, both uninformed (DLS and IDDFS) and informed (Greedy Best-First with MisplacedTileHeuristic and ManhattanHeuristic, A* with the same heuristics, and Simulated Annealing).

From this demo, several conclusions can be drawn:

1. DLS, when using an appropriate depth limit, finds the optimal solution with very few generated/expanded nodes. If the limit were too small, a cutoff would occur; if it were too large, a non-optimal solution with higher cost could be found.

2. Iterative DLS must generate and expand more nodes, since it increases the DLS depth limit each time (limit = 0, 1, …) until it finds the first solution, which will be the optimal one.

3. Greedy Best-First Search with both heuristics finds a solution in three steps, which is correct and very efficient, but optimality of the solution is not always guaranteed.

4. With A*, the optimal path is found using both heuristics, but with a large difference between them: using *MisplacedTiles*, 25 nodes are expanded, while using *Manhattan*, only 11 are expanded. This shows the superiority of the Manhattan heuristic over MisplacedTiles in the 8-puzzle problem (fewer expansions and less memory usage).

5. Simulated Annealing may fail; in this case, it fails after expanding 100 nodes. It is useful for very large search spaces when guarantees are not required, but in this case its use is not desirable.

Next, our own demo was designed, grouping all available uninformed search algorithms (BFS in graph and tree versions, DFS in graph and tree versions, DLS with depths 9 and 3, IDDFS, and UCS in graph and tree versions). These were tested using different initial states with solutions at depths of 3, 9, and 30 steps respectively. Data were collected on solution depth, expanded nodes, number of elements in the frontier queue when the solution is found, maximum number of elements in the frontier queue, time, generated nodes, and rate (nodes generated per second). Algorithms that do not finish in a reasonable time or that require excessive memory are marked with (1) or (2); since they are aborted, no execution data are shown.

The table with these metrics is available in Appendix 1.

From this table, several conclusions can be drawn:

1. As the solution depth increases (3, 9, 30), the computational effort grows explosively for all algorithms. This is clearly visible in BFS-GRAPH: 5, 288, and 181,058 expanded nodes respectively.
2. Graph-based algorithms exhibit more memory growth than tree-based ones, but tree-based algorithms grow much more rapidly overall (many of them do not even finish).
3. In terms of memory cost, the most efficient algorithms are BFS-GRAPH and UCS-GRAPH.

For the development of the demo, a function called *eightPuzzleSearch()* was created. This function receives a search algorithm type (any of those registered in *tipoSearch()*) and an initial board state. It manages the execution of the algorithm and prints the results to the screen. Error handling is performed using exceptions (both memory-related and time-related). For time management, a separate thread executes the search; if this thread does not finish within a given time, a *TimeoutException* is thrown and the thread is stopped. From the *main* method, all algorithms are executed with the three different initial states.

Regarding time complexity, from a theoretical point of view we know that:

- **BFS**: time complexity = $O(b^d)$ and space complexity = $O(b^d)$,
  where *b* is the branching factor and *d* is the solution depth.
- **DFS**: time complexity = $O(b^m)$ and space complexity = $O(b \cdot m)$,
  where *m* is the maximum depth of the tree.
- **DLS**: time complexity = $O(b^l)$ and space complexity = $O(b \cdot l)$,
  where *l* is the depth limit.
- **IDDFS**: time complexity = $O(b^d)$ and space complexity = $O(b \cdot d)$.
- **UCS**: time complexity = $O(b^{(1 + \lfloor C^*/\varepsilon \rfloor)})$ and space complexity = $O(b^{(1 + \lfloor C^*/\varepsilon \rfloor)})$.

As observed, the experimental results are consistent with the theoretical analysis.

Missionaries and Cannibals Problem:

With the knowledge acquired so far, we developed the code defining the Missionaries and Cannibals problem, as well as a demo to observe the metrics and the evolution of actions in each case:

- **CanibalesBoard**: Defines the current state of the game, storing the number of cannibals on one side, from which the numbers on the other side can be inferred and queried.

- **CanibalesFunctionFactory**: Defines the different possible actions, along with methods to obtain the list of possible successor states, update the state of the problem after performing an action, and determine the cost of each action (which is 1 for all of them).

- **CanibalesGoalTest**: Used to check whether the current state is the goal state.

- **CanibalesPract1**: Our demo, in which uninformed search algorithms are executed starting from an initial state with 3 cannibals and 3 missionaries on the left bank. In design, it is very similar to the 8-puzzle demo; however, more detail is given to the on-screen presentation using several functions.

The table with metrics and the sequence of actions leading to the goal state is available in Appendix 2.

DFS-TREE does not finish within the established time limit, and DLS does not find a solution, resulting in a cutoff. All other algorithms find the optimal solution in 11 actions. UCS-GRAPH, BFS-GRAPH, and DFS-GRAPH are the most efficient both in terms of time and space.

Conclusion:

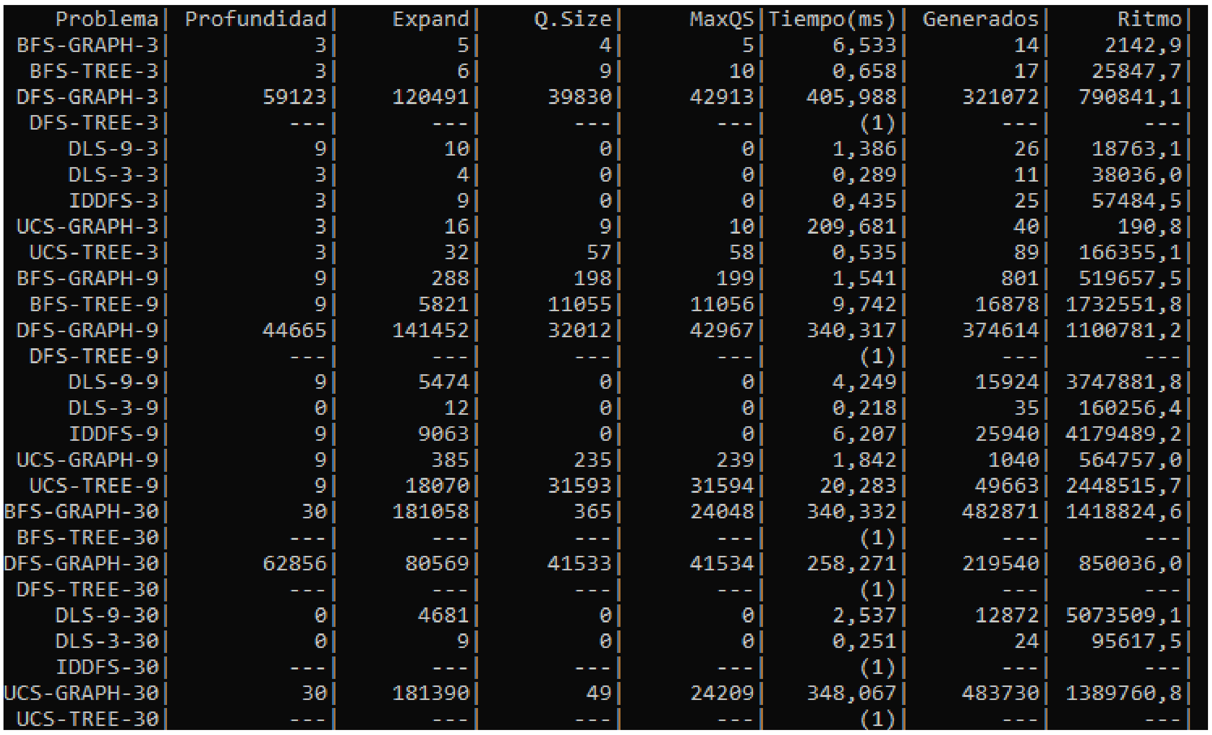
I have learned how to define search problems and have gained a detailed understanding of the specifications of each uninformed search algorithm (and some informed ones).

Depending on the problem, some algorithms are more optimal than others, and it is worthwhile to understand each algorithm in detail, since they can be applied specifically to the search environments of a given problem, making searches faster and/or more memory-efficient.

We have also seen that, in some cases, it is useful to develop programs to compare which method is most effective for different initial states of the problem.

Appendix 1:

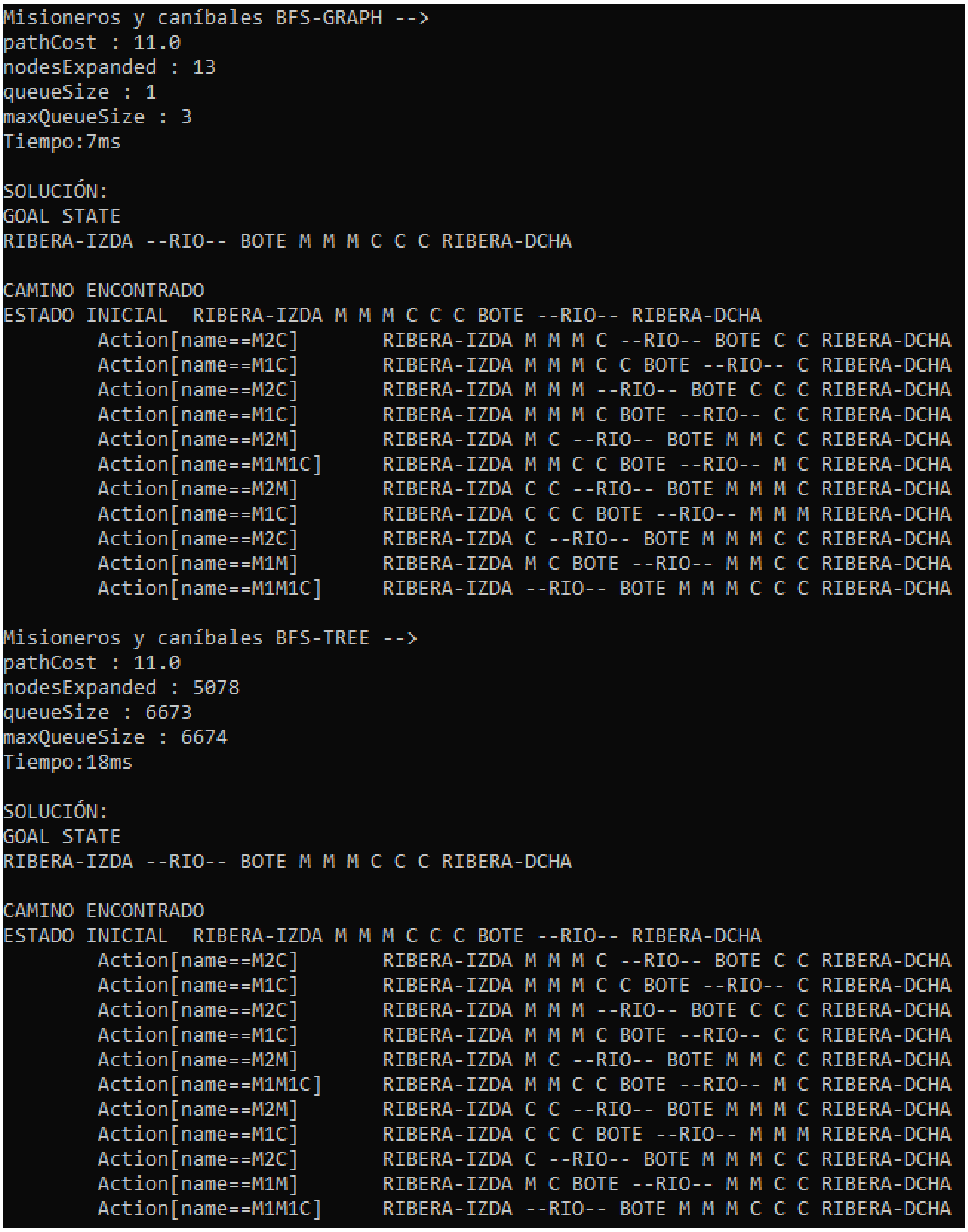
<u>Table with metrics of different search algorithms for 8-puzzle:</u>



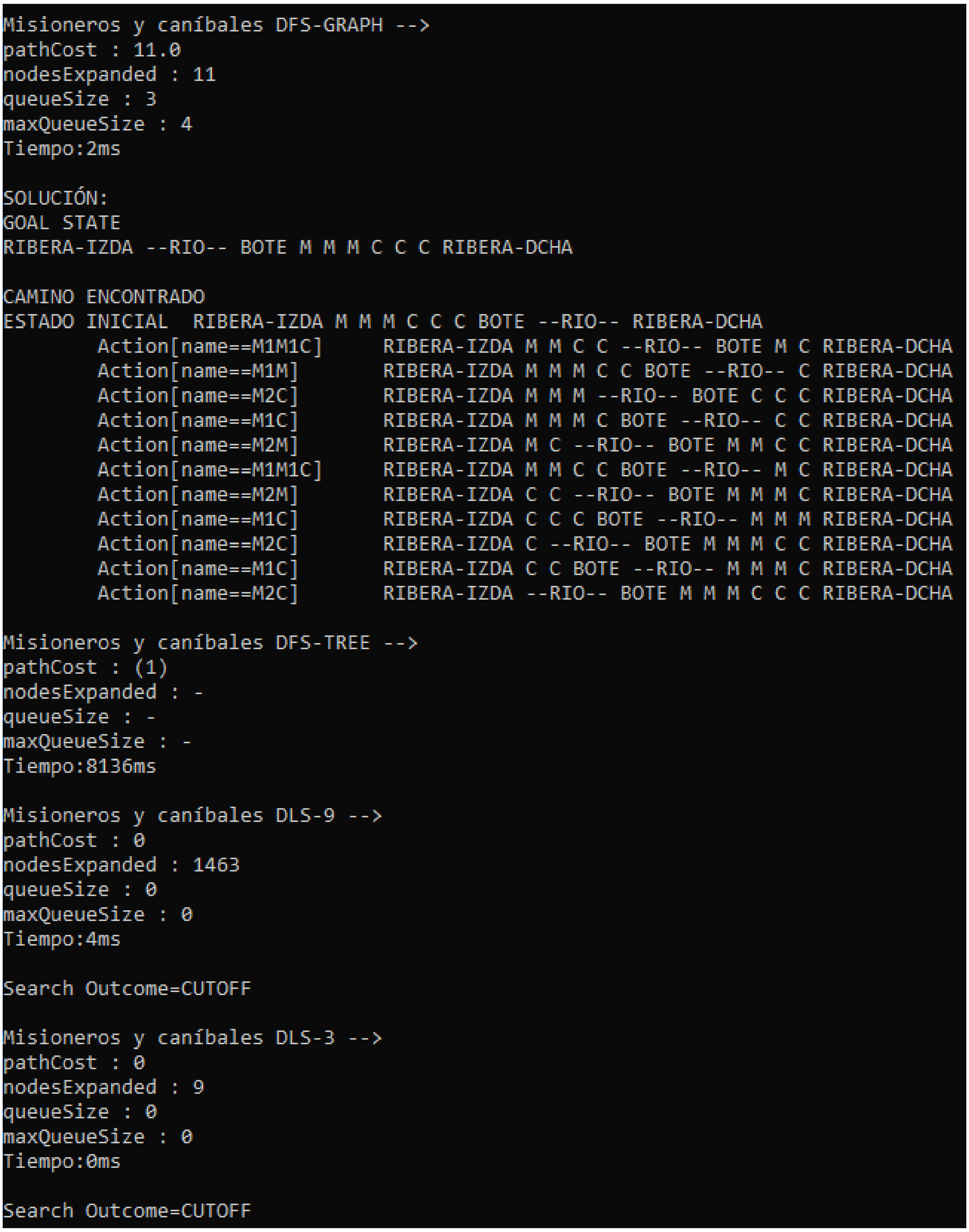
```
     Problema| Profundidad|     Expand|     Q.Size|      MaxQS|Tiempo(ms)|  Generados|      Ritmo|
  BFS-GRAPH-3|           3|          5|          4|          5|     6,533|         14|     2142,9|
   BFS-TREE-3|           3|          6|          9|         10|     0,658|         17|    25847,7|
  DFS-GRAPH-3|       59123|     120491|      39830|      42913|   405,988|     321072|   790841,1|
   DFS-TREE-3|         ---|        ---|        ---|        ---|       (1)|        ---|        ---|
      DLS-9-3|           9|         10|          0|          0|     1,386|         26|    18763,1|
      DLS-3-3|           3|          4|          0|          0|     0,289|         11|    38036,0|
      IDDFS-3|           3|          9|          0|          0|     0,435|         25|    57484,5|
  UCS-GRAPH-3|           3|         16|          9|         10|   209,681|         40|      190,8|
   UCS-TREE-3|           3|         32|         57|         58|     0,535|         89|   166355,1|
  BFS-GRAPH-9|           9|        288|        198|        199|     1,541|        801|   519657,5|
   BFS-TREE-9|           9|       5821|      11055|      11056|     9,742|      16878|  1732551,8|
  DFS-GRAPH-9|       44665|     141452|      32012|      42967|   340,317|     374614|  1100781,2|
   DFS-TREE-9|         ---|        ---|        ---|        ---|       (1)|        ---|        ---|
      DLS-9-9|           9|       5474|          0|          0|     4,249|      15924|  3747881,8|
      DLS-3-9|           0|         12|          0|          0|     0,218|         35|   160256,4|
      IDDFS-9|           9|       9063|          0|          0|     6,207|      25940|  4179489,2|
  UCS-GRAPH-9|           9|        385|        235|        239|     1,842|       1040|   564757,0|
   UCS-TREE-9|           9|      18070|      31593|      31594|    20,283|      49663|  2448515,7|
BFS-GRAPH-30|           30|     181058|        365|      24048|   340,332|     482871|  1418824,6|
  BFS-TREE-30|         ---|        ---|        ---|        ---|       (1)|        ---|        ---|
DFS-GRAPH-30|        62856|      80569|      41533|      41534|   258,271|     219540|   850036,0|
  DFS-TREE-30|         ---|        ---|        ---|        ---|       (1)|        ---|        ---|
     DLS-9-30|           0|       4681|          0|          0|     2,537|      12872|  5073509,1|
     DLS-3-30|           0|          9|          0|          0|     0,251|         24|    95617,5|
     IDDFS-30|         ---|        ---|        ---|        ---|       (1)|        ---|        ---|
UCS-GRAPH-30|           30|     181390|         49|      24209|   348,067|     483730|  1389760,8|
  UCS-TREE-30|         ---|        ---|        ---|        ---|       (1)|        ---|        ---|
```

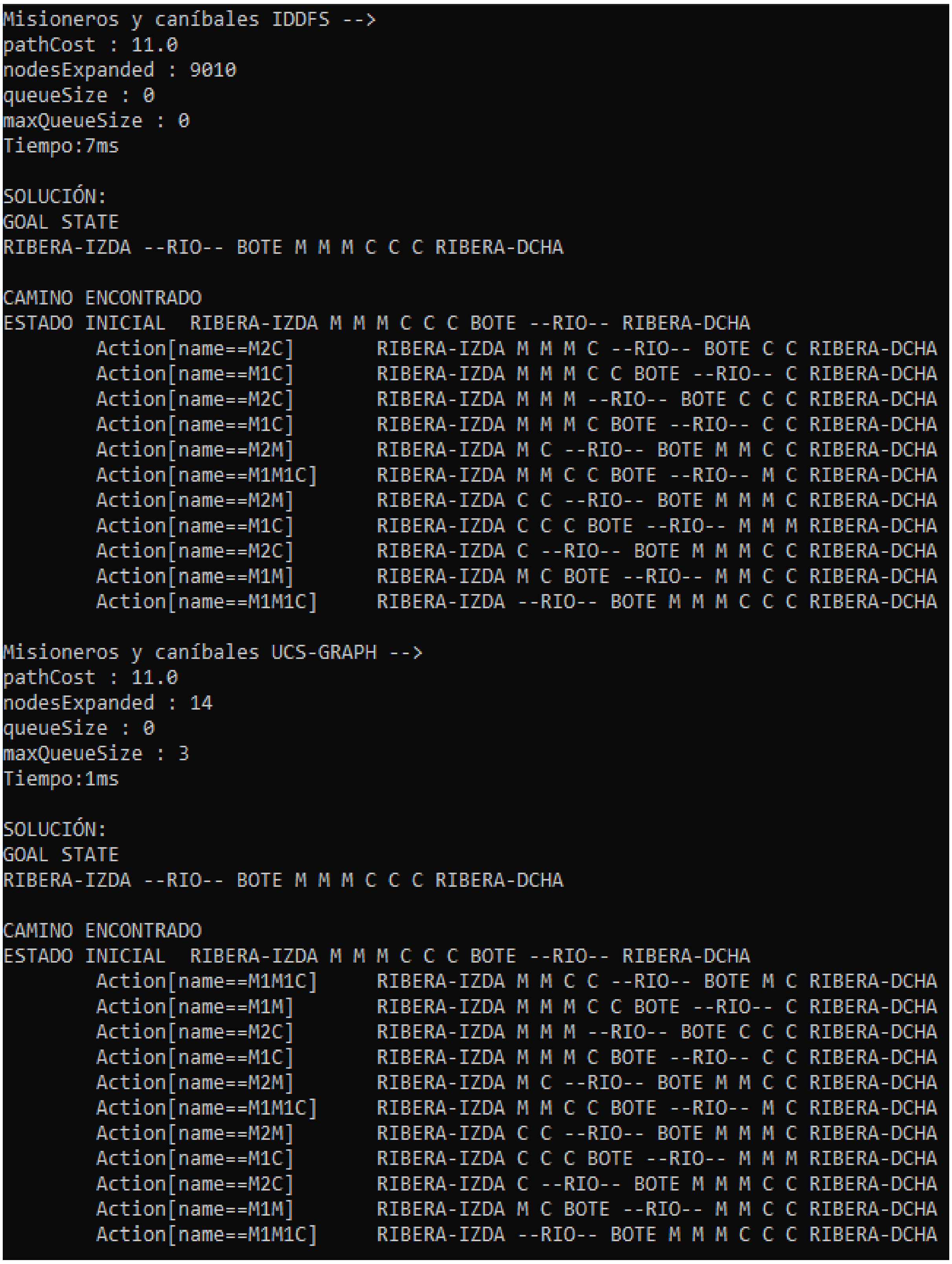
Appendix 2:

Actions of uninformed search algorithms regarding missionaries and cannibals:

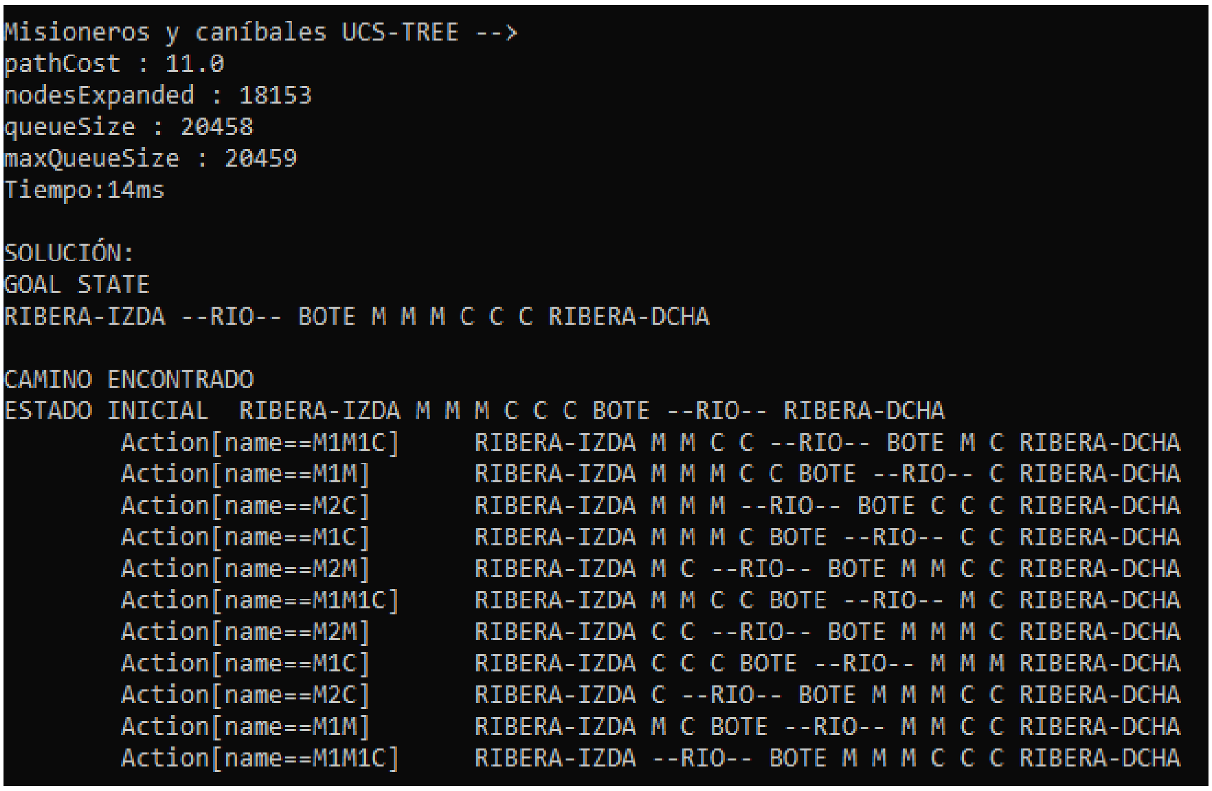


```
Misioneros y caníbales BFS-GRAPH -->
pathCost : 11.0
nodesExpanded : 13
queueSize : 1
maxQueueSize : 3
Tiempo:7ms

SOLUCIÓN:
GOAL STATE
RIBERA-IZDA --RIO-- BOTE M M M C C C RIBERA-DCHA

CAMINO ENCONTRADO
ESTADO INICIAL  RIBERA-IZDA M M M C C C BOTE --RIO-- RIBERA-DCHA
        Action[name==M2C]       RIBERA-IZDA M M M C --RIO-- BOTE C C RIBERA-DCHA
        Action[name==M1C]       RIBERA-IZDA M M M C C BOTE --RIO-- C RIBERA-DCHA
        Action[name==M2C]       RIBERA-IZDA M M M --RIO-- BOTE C C C RIBERA-DCHA
        Action[name==M1C]       RIBERA-IZDA M M M C BOTE --RIO-- C C RIBERA-DCHA
        Action[name==M2M]       RIBERA-IZDA M C --RIO-- BOTE M M C C RIBERA-DCHA
        Action[name==M1M1C]     RIBERA-IZDA M M C C BOTE --RIO-- M C RIBERA-DCHA
        Action[name==M2M]       RIBERA-IZDA C C --RIO-- BOTE M M M C RIBERA-DCHA
        Action[name==M1C]       RIBERA-IZDA C C C BOTE --RIO-- M M M RIBERA-DCHA
        Action[name==M2C]       RIBERA-IZDA C --RIO-- BOTE M M M C C RIBERA-DCHA
        Action[name==M1M]       RIBERA-IZDA M C BOTE --RIO-- M M C C RIBERA-DCHA
        Action[name==M1M1C]     RIBERA-IZDA --RIO-- BOTE M M M C C C RIBERA-DCHA

Misioneros y caníbales BFS-TREE -->
pathCost : 11.0
nodesExpanded : 5078
queueSize : 6673
maxQueueSize : 6674
Tiempo:18ms

SOLUCIÓN:
GOAL STATE
RIBERA-IZDA --RIO-- BOTE M M M C C C RIBERA-DCHA

CAMINO ENCONTRADO
ESTADO INICIAL  RIBERA-IZDA M M M C C C BOTE --RIO-- RIBERA-DCHA
        Action[name==M2C]       RIBERA-IZDA M M M C --RIO-- BOTE C C RIBERA-DCHA
        Action[name==M1C]       RIBERA-IZDA M M M C C BOTE --RIO-- C RIBERA-DCHA
        Action[name==M2C]       RIBERA-IZDA M M M --RIO-- BOTE C C C RIBERA-DCHA
        Action[name==M1C]       RIBERA-IZDA M M M C BOTE --RIO-- C C RIBERA-DCHA
        Action[name==M2M]       RIBERA-IZDA M C --RIO-- BOTE M M C C RIBERA-DCHA
        Action[name==M1M1C]     RIBERA-IZDA M M C C BOTE --RIO-- M C RIBERA-DCHA
        Action[name==M2M]       RIBERA-IZDA C C --RIO-- BOTE M M M C RIBERA-DCHA
        Action[name==M1C]       RIBERA-IZDA C C C BOTE --RIO-- M M M RIBERA-DCHA
        Action[name==M2C]       RIBERA-IZDA C --RIO-- BOTE M M M C C RIBERA-DCHA
        Action[name==M1M]       RIBERA-IZDA M C BOTE --RIO-- M M C C RIBERA-DCHA
        Action[name==M1M1C]     RIBERA-IZDA --RIO-- BOTE M M M C C C RIBERA-DCHA
```


```
Misioneros y caníbales DFS-GRAPH -->
pathCost : 11.0
nodesExpanded : 11
queueSize : 3
maxQueueSize : 4
Tiempo:2ms

SOLUCIÓN:
GOAL STATE
RIBERA-IZDA --RIO-- BOTE M M M C C C RIBERA-DCHA

CAMINO ENCONTRADO
ESTADO INICIAL  RIBERA-IZDA M M M C C C BOTE --RIO-- RIBERA-DCHA
        Action[name==M1M1C]     RIBERA-IZDA M M C C --RIO-- BOTE M C RIBERA-DCHA
        Action[name==M1M]       RIBERA-IZDA M M M C C BOTE --RIO-- C RIBERA-DCHA
        Action[name==M2C]       RIBERA-IZDA M M M --RIO-- BOTE C C C RIBERA-DCHA
        Action[name==M1C]       RIBERA-IZDA M M M C BOTE --RIO-- C C RIBERA-DCHA
        Action[name==M2M]       RIBERA-IZDA M C --RIO-- BOTE M M C C RIBERA-DCHA
        Action[name==M1M1C]     RIBERA-IZDA M M C C BOTE --RIO-- M C RIBERA-DCHA
        Action[name==M2M]       RIBERA-IZDA C C --RIO-- BOTE M M M C RIBERA-DCHA
        Action[name==M1C]       RIBERA-IZDA C C C BOTE --RIO-- M M M RIBERA-DCHA
        Action[name==M2C]       RIBERA-IZDA C --RIO-- BOTE M M M C C RIBERA-DCHA
        Action[name==M1C]       RIBERA-IZDA C C BOTE --RIO-- M M M C RIBERA-DCHA
        Action[name==M2C]       RIBERA-IZDA --RIO-- BOTE M M M C C C RIBERA-DCHA

Misioneros y caníbales DFS-TREE -->
pathCost : (1)
nodesExpanded : -
queueSize : -
maxQueueSize : -
Tiempo:8136ms

Misioneros y caníbales DLS-9 -->
pathCost : 0
nodesExpanded : 1463
queueSize : 0
maxQueueSize : 0
Tiempo:4ms

Search Outcome=CUTOFF

Misioneros y caníbales DLS-3 -->
pathCost : 0
nodesExpanded : 9
queueSize : 0
maxQueueSize : 0
Tiempo:0ms

Search Outcome=CUTOFF
```



```
Misioneros y caníbales IDDFS -->
pathCost : 11.0
nodesExpanded : 9010
queueSize : 0
maxQueueSize : 0
Tiempo:7ms

SOLUCIÓN:
GOAL STATE
RIBERA-IZDA --RIO-- BOTE M M M C C C RIBERA-DCHA

CAMINO ENCONTRADO
ESTADO INICIAL  RIBERA-IZDA M M M C C C BOTE --RIO-- RIBERA-DCHA
        Action[name==M2C]       RIBERA-IZDA M M M C --RIO-- BOTE C C RIBERA-DCHA
        Action[name==M1C]       RIBERA-IZDA M M M C C BOTE --RIO-- C RIBERA-DCHA
        Action[name==M2C]       RIBERA-IZDA M M M --RIO-- BOTE C C C RIBERA-DCHA
        Action[name==M1C]       RIBERA-IZDA M M M C BOTE --RIO-- C C RIBERA-DCHA
        Action[name==M2M]       RIBERA-IZDA M C --RIO-- BOTE M M C C RIBERA-DCHA
        Action[name==M1M1C]     RIBERA-IZDA M M C C BOTE --RIO-- M C RIBERA-DCHA
        Action[name==M2M]       RIBERA-IZDA C C --RIO-- BOTE M M M C RIBERA-DCHA
        Action[name==M1C]       RIBERA-IZDA C C C BOTE --RIO-- M M M RIBERA-DCHA
        Action[name==M2C]       RIBERA-IZDA C --RIO-- BOTE M M M C C RIBERA-DCHA
        Action[name==M1M]       RIBERA-IZDA M C BOTE --RIO-- M M C C RIBERA-DCHA
        Action[name==M1M1C]     RIBERA-IZDA --RIO-- BOTE M M M C C C RIBERA-DCHA

Misioneros y caníbales UCS-GRAPH -->
pathCost : 11.0
nodesExpanded : 14
queueSize : 0
maxQueueSize : 3
Tiempo:1ms

SOLUCIÓN:
GOAL STATE
RIBERA-IZDA --RIO-- BOTE M M M C C C RIBERA-DCHA

CAMINO ENCONTRADO
ESTADO INICIAL  RIBERA-IZDA M M M C C C BOTE --RIO-- RIBERA-DCHA
        Action[name==M1M1C]     RIBERA-IZDA M M C C --RIO-- BOTE M C RIBERA-DCHA
        Action[name==M1M]       RIBERA-IZDA M M M C C BOTE --RIO-- C RIBERA-DCHA
        Action[name==M2C]       RIBERA-IZDA M M M --RIO-- BOTE C C C RIBERA-DCHA
        Action[name==M1C]       RIBERA-IZDA M M M C BOTE --RIO-- C C RIBERA-DCHA
        Action[name==M2M]       RIBERA-IZDA M C --RIO-- BOTE M M C C RIBERA-DCHA
        Action[name==M1M1C]     RIBERA-IZDA M M C C BOTE --RIO-- M C RIBERA-DCHA
        Action[name==M2M]       RIBERA-IZDA C C --RIO-- BOTE M M M C RIBERA-DCHA
        Action[name==M1C]       RIBERA-IZDA C C C BOTE --RIO-- M M M RIBERA-DCHA
        Action[name==M2C]       RIBERA-IZDA C --RIO-- BOTE M M M C C RIBERA-DCHA
        Action[name==M1M]       RIBERA-IZDA M C BOTE --RIO-- M M C C RIBERA-DCHA
        Action[name==M1M1C]     RIBERA-IZDA --RIO-- BOTE M M M C C C RIBERA-DCHA
```



```
Misioneros y caníbales UCS-TREE -->
pathCost : 11.0
nodesExpanded : 18153
queueSize : 20458
maxQueueSize : 20459
Tiempo:14ms

SOLUCIÓN:
GOAL STATE
RIBERA-IZDA --RIO-- BOTE M M M C C C RIBERA-DCHA

CAMINO ENCONTRADO
ESTADO INICIAL  RIBERA-IZDA M M M C C C BOTE --RIO-- RIBERA-DCHA
        Action[name==M1M1C]     RIBERA-IZDA M M C C --RIO-- BOTE M C RIBERA-DCHA
        Action[name==M1M]       RIBERA-IZDA M M M C C BOTE --RIO-- C RIBERA-DCHA
        Action[name==M2C]       RIBERA-IZDA M M M --RIO-- BOTE C C C RIBERA-DCHA
        Action[name==M1C]       RIBERA-IZDA M M M C BOTE --RIO-- C C RIBERA-DCHA
        Action[name==M2M]       RIBERA-IZDA M C --RIO-- BOTE M M C C RIBERA-DCHA
        Action[name==M1M1C]     RIBERA-IZDA M M C C BOTE --RIO-- M C RIBERA-DCHA
        Action[name==M2M]       RIBERA-IZDA C C --RIO-- BOTE M M M C RIBERA-DCHA
        Action[name==M1C]       RIBERA-IZDA C C C BOTE --RIO-- M M M RIBERA-DCHA
        Action[name==M2C]       RIBERA-IZDA C --RIO-- BOTE M M M C C RIBERA-DCHA
        Action[name==M1M]       RIBERA-IZDA M C BOTE --RIO-- M M C C RIBERA-DCHA
        Action[name==M1M1C]     RIBERA-IZDA --RIO-- BOTE M M M C C C RIBERA-DCHA
```