Heuristic Search Methods for Cohesion Free

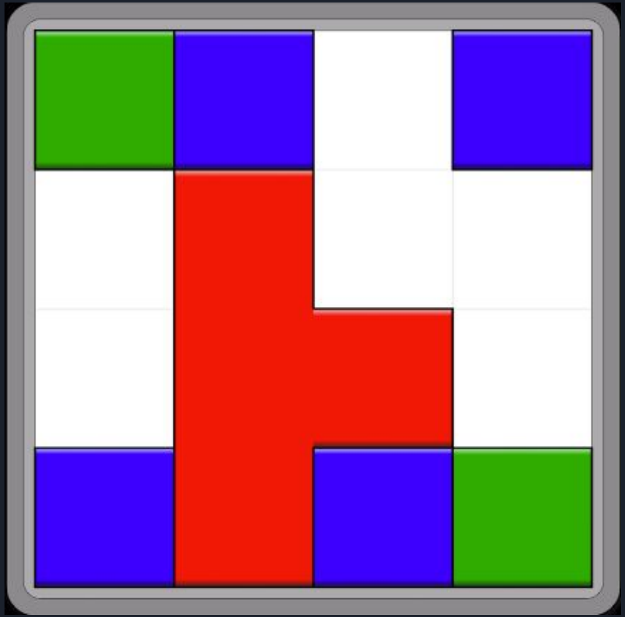
T05 - G05

Artificial Intelligence

Diogo Costa, 202007770
Fábio Sá, 202007658
João Araújo, 202004293

Game Definition

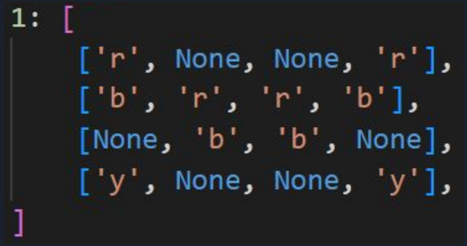
- The game Cohesion Free is a single-player puzzle game played on a pre-generated board.
- The goal is to combine all squares of the same color by moving them until they touch and stick together.
- The game consists of a grid of squares of different colors, and adjacent squares of the same color will bond together permanently.
- The game ends when all tiles of each color have combined or when the player can no longer make any moves, in which case he has lost.



State Representation

The game state can be represented as a matrix of chars and null values, *B*, which represents a pre-generated board with symmetrical, yet variable, height and length. The number accompanied represents the level.
Each tile on the board links to a unique color (of 4 available), including null, in case of an empty space, and its position in the matrix indicates its coordinates.

Ex:



```
1: [
    ['r', None, None, 'r'],
    ['b', 'r', 'r', 'b'],
    [None, 'b', 'b', None],
    ['y', None, None, 'y'],
]
```

Initial State

The initial state of the game is variable, as there are different starting combinations of the board.

Objective Test

The game ends when the player can either no longer make any moves, in which case he lost, or when there is only 1 cluster of each color. In case of a win, the score of the game is determined by how many moves the player executed to win: the game's score is higher the smaller the number of moves

Operators

(*p*,*d*)

The operators can be defined as sliding a color cluster or separated tile either *right*, *down*, *left,* or *up*.

The operators can be represented as a tuple (*p*,*d*) where *p* represents a cluster/piece and *d* is the direction of the slide (0 for *right*, 1 for *down*, 2 for *left,* and 3 for *up*).

The preconditions for the operator are that the slide does not go beyond the edge of the board and that there are no tiles in the way. The effect is to slide the cluster/tile by 1 unit in the specified direction.

The score is inversely proportional to the amounts of slides used, so the cost is to lessen the score with each move.

Search Algorithms

Uninformed Search

- **Depth First Search (DFS)** even if it can get stuck for a long time on dead-end branches, all tested boards have a solution.
- **Breadth-first Search (BFS)** takes a lot of memory space to visit all nodes but finds the closest solution (with lowest cost, number of moves).

Informed Search

- **Greedy Search**, expand the node that appears to be closest to the solution. In our case, it will always choose the movement that allows to join pieces or clusters of the same color.
- **A* Algorithm**, to guide the solution based on heuristic functions that give an informed degree to the research. In the game it is important to know the distance between pieces of the same color and the number of moves performed.

Heuristic functions

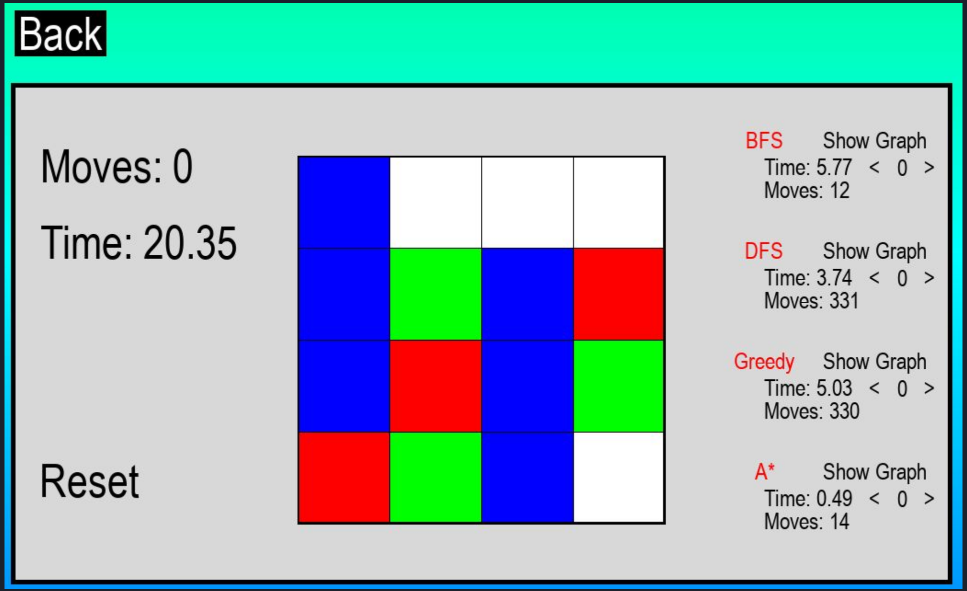
Heuristic 1
The value is given by the number of clusters present minus the number of colors on the board, which represents the number of clusters that remain separated.

Heuristic 2
The sum of all orthogonal distances between tiles of the same color. It's a realistic estimate of the number of steps to take to win the game.

Heuristic 3
The number of matches between pieces left to win the game is multiplied by the width of the board. It estimates the number of moves since, on average, to join two pieces it is necessary to cover half of the board in X plus half of the board in Y. As it is always square, X/2 + Y/2 = side of the board.

Approach

The game allows you to choose the difficulty of the game board. All levels have a solution.
Each algorithm can run individually and be visualized step by step.
The informed algorithms show the best result, that is, with the best heuristic.
The search tree can be accessed once the algorithm ends.



Moves: 0
Time: 20.35

BFS
Time: 5.77
Moves: 12

DFS
Time: 3.74
Moves: 331

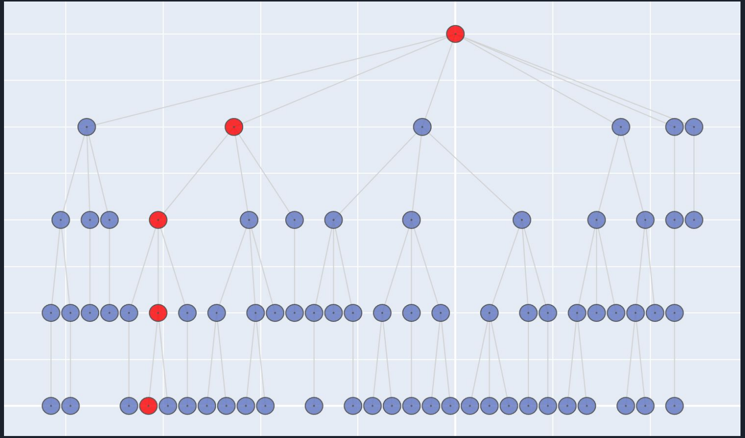
Greedy
Time: 5.03
Moves: 330

A*
Time: 0.49
Moves: 14

Uninformed Search

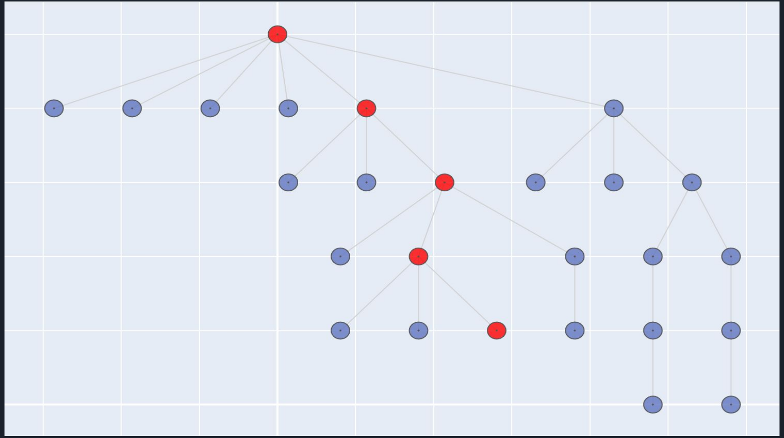
BFS

The breadth-first search algorithm always found the best solution (path with the least number of moves). The expansion of the child nodes was total, so there were considerable time and space costs.



DFS

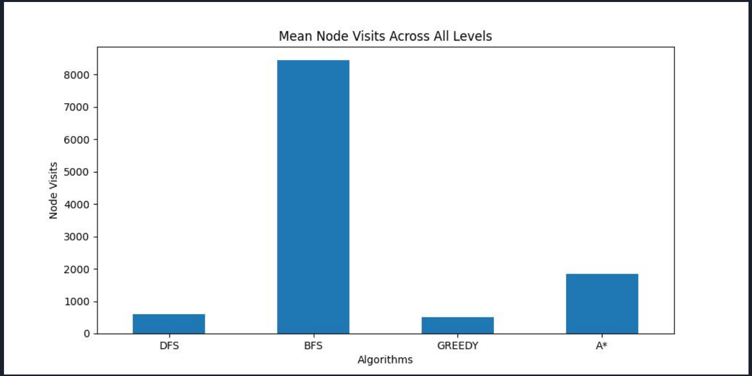
The depth-first search algorithm found a solution in a very short time. Always expanding the last node found allowed finding a solution quickly, but also with too many accumulated moves.

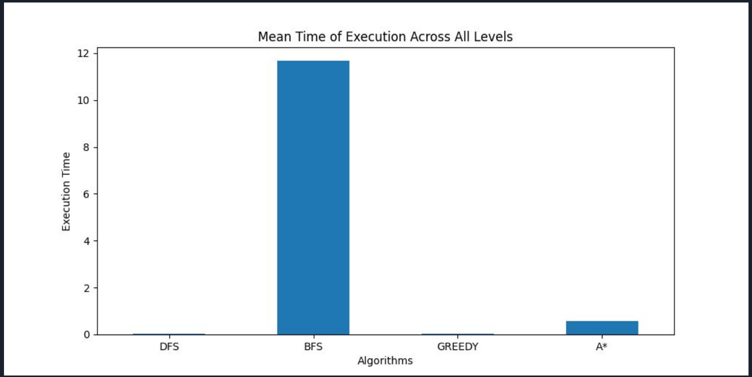


Uninformed Search

Regardless of the difficulty and board size, it was found that:

- DFS execution time is much lower than BFS;
- BFS solutions are optimal (finds the solution with the minimum number of moves);
- The memory spent and the number of expanded nodes in BFS search is much higher than in DFS;

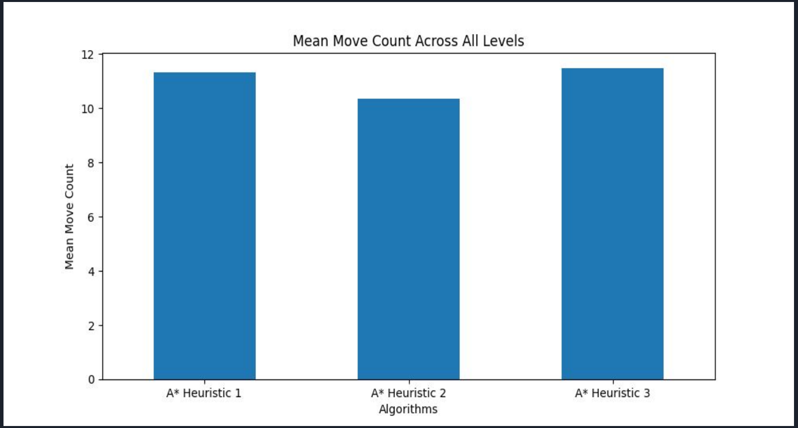


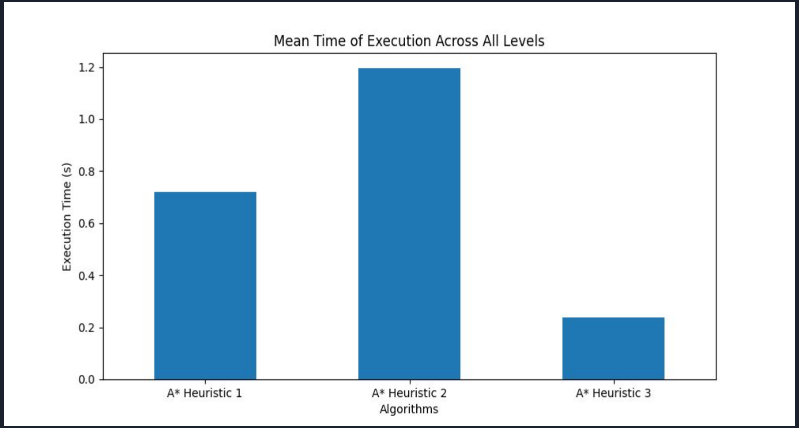


Informed Search - Heuristics

All three evaluated heuristics are admissible, as they never overestimate the real cost of reaching the final goal from the current state.

However, heuristic 2, which estimates the number of moves based on the orthogonal distance between two pieces, was the one with the best results. Heuristic 3 ensured faster results with less memory usage.

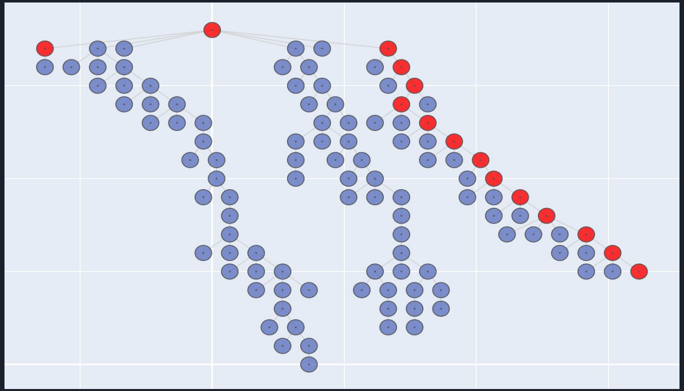




Informed Search

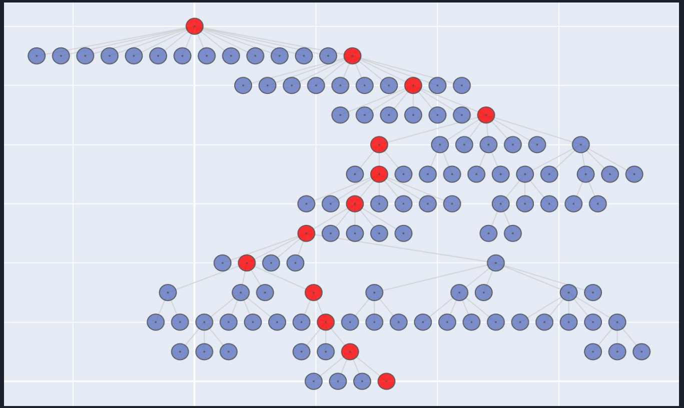
Greedy

Node expansion was guided only by the proximity to the final solution: a node that joined two or more pieces was always expanded over those that didn't. The algorithm did not always give the optimal solution, as there are situations where the combination of local optimal solutions does not guarantee a global optimal solution.



A *

The expansion of the nodes was guided by the proximity to the final solution and the cost of the path to each step, trying to minimize both. The A* algorithm always gave the optimal solution with much fewer resources spent compared to the BFS (time and space).



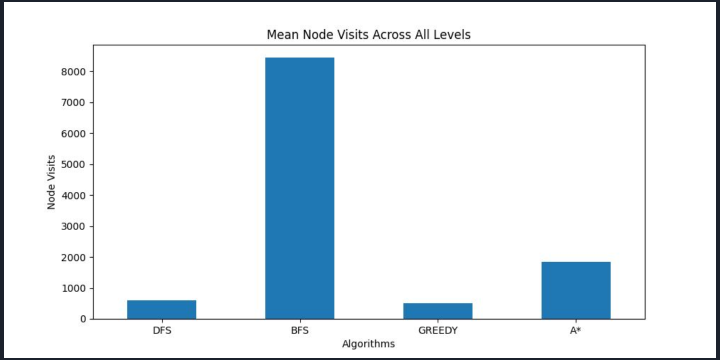
Informed Search

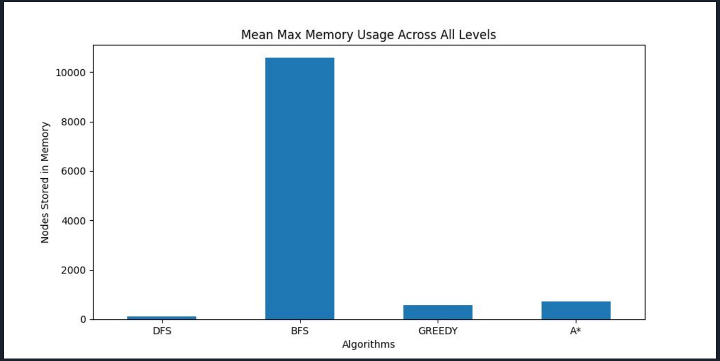
Regardless of the difficulty and board size, it was found that:

- The execution time of Greedy and A* are of the same order of magnitude;

- A* solutions are mostly optimal, finding one of the best solutions;

- Memory spent and number of expanded nodes in A* search is slightly more than Greedy;





Conclusions

With this project, we consolidated our knowledge of the search algorithms that stand behind single-player games.

We deepened the importance of good heuristics to guide informed search algorithms, which allows us to obtain one of the most optimal solutions in a timely manner, wasting far less space than otherwise required.

Additionally, we also learned the significance of considering the trade-off between time complexity and space complexity in developing efficient search algorithms for single-player games.

References

- NeatWits. Cohesion Free. 2015. Accessed 27 February 2023.
- Plotly. Graph Objects. 2023. Accessed 15 March 2023.
- Igraph. Graph Objects. 2023. Accessed 15 March 2023.
- Psutil. Process Memory Utilization. 2023. Accessed 24 March 2023.
- Prajjwal Pathak. Picture Sliding Puzzle. 2022. Accessed 3 March 2023. (Idea for the GUI)

Material Usage

Software:

- Visual Studio Code and PyCharm as the used IDE’s;
- Excel to export the data to be used on graphs and bar charts.

Languages:

- For the development of this application, we required only the usage of Python, version 3.10.10.