TÉCNICO LISBOA


# IASD 2022/23 Project Assignment #3: 'Roll the Ball' Slide Puzzle

Luis Custódio and Rodrigo Ventura
*Instituto Superior Técnico*, University of Lisbon

(Version 1.1, October 25, 2022)

## Introduction

Free the Ball, or Roll the Ball, is a classic tile puzzle where the goal is to move the sliding tiles to unblock a path for the ball to roll to the exit.

The goal of the project is to develop a program using Python (version 3) to search for a solution, if any, for a 'Roll the Ball' puzzle. The project will be divided in three parts, and three deliverables. This document defines the third assignment and deliverable.

Notice that you should read again Assignment #1 to remember the rules of the puzzle and the input file format.

# 1 Third Assignment and Deliverable

The third assignment is to develop a Python program capable of reading a puzzle (see Assignment #1), represent the state of the problem, define the `result`, `actions` and `goal_test` methods (see Assignment #2) and solving it using the A* search algorithm with a heuristic function that has to be developed.

The Python program to be delivered should be called `solution.py` and include (at least) a class with name `RTBProblem` containing (at least) the following methods (see Annex A for the class template):

`result(state,action)` returns the state that results from executing the given action in the given state.

`actions(state)` returns the list of actions that can be executed in the given state.

`goal_test(state)` returns `True` if the given state is a goal.

`load(fh)` loads a puzzle (initial configuration) from a file object $fh$.

`h(node)` estimates the cost from the current node to a goal node (the heuristic function). Notice that the argument for the `h` method is a node not a state.

In order to solve a puzzle with a heuristic one needs to:

- read the initial configuration of the puzzle from a file object like you did for Assignment #1,
- implement a representation for the state of the problem and the methods `result`, `actions` and `goal_test`. Preferably, you should reuse what you did for Assignment #2, but you may change whatever you want,
- implement a heuristic function for the RTB problem. Use docstrings (triple quotation marks) to (see Annex A):
  1. describe what your heuristic do, and
  2. justify if it is consistent and/or admissible, and its consequence to optimality.

The optimal solution for a RTB problem is a solution with the minimum number of movements, and each movement costs the same (e.g., 1).

## 2 Evaluation

The deliverable for this assignment is shown through DEEC Moodle, with the submission of a single python file, called `solution.py`, implementing the modules mentioned above. Instructions for this platform are available on the course webpage. Finally, the grade is computed in the following way:

- 50% from the public tests;
- 50% from the private tests; and
- -15% from the code structure, quality and readability.

On the Moodle plataform, the evaluator will make the following calls to determine the execution time and the number of nodes generated[1] by the search algorithm with and without the heuristic:

```
problem1 = CountCalls(solution.RTBProblem())
t10 = time.process_time()
with open(input) as fh:
        problem1.load(fh)
result1 = uniform_cost_search(problem1)
t11 = time.process_time()
problem2 = CountCalls(solution.RTBProblem())
t20 = time.process_time()
with open(input) as fh:
        problem2.load(fh)
result2 = astar_search(problem2)
t21 = time.process_time()
```

[1] See section 'Reporting Summary Statistics on Search Algorithms' in the file search4e.ipynb of the AIMA repository to understand what the class CountCalls is for.

Depending on the type of the heuristic developed, the solution returned may be optimal or suboptimal.

Let $G$ be the max grade of each test, $C$ be the path cost of the solution returned with the heuristic, $O$ be the optimal cost, $SH$ be the number of nodes generated without the heuristic, $CH$ be the number of nodes generated with the heuristic and $m$ be a margin for the suboptimality of each test.

Then the grade of each test is calculated as follows:

```
if C == O: #optimal found
        if 3/4*SH < CH < SH:
                grade = 0.9 * G
        elif CH <= 3/4*SH:
                grade = G
        else:
                grade = 0
elif O < C <= O + m: #suboptimal found
        if 1/2*SH < CH < SH:
                grade = 0.8 * G
        elif CH <= 1/2*SH:
                grade = 0.9 * G
        else:
                grade = 0
else:
        grade = 0
```

Deadline: **04-November-2022**. Projects submitted after the deadline will not be considered for evaluation.

**Closing Remarks on Ethics:**

- Any kind of sharing code outside your group is considered plagiarism;
- Developing your code in any open software development tool is considered sharing code;
- You can use GitHub. Make sure you have private projects and remove them afterward;
- If you get caught in any plagiarism, either by copying the code/ideas or sharing them with others, you will not be graded; and
- The scripts and other supporting materials produced by the instructors cannot be made public!

# A Class Template

```
import search
""" For this assignment, you just need to import search.py,
which in turn imports utils.py; both files are available
in the repository mentioned on page 2 of the Assignment 2"""

class RTBProblem(search.Problem):

        def __init__(self):
                """Method that instantiate your class.
                        You can change the content of this.
                        self.initial is where the initial state of
                        the puzzle should be saved."""
                self.initial = None
                self.algorithm = None

        def result(self, state, action):
                """Return the state that results from executing
                the given action in the given state."""
                pass

        def actions(self, state):
                """Return the actions that can be executed in
                the given state."""
                pass

        def goal_test(self, state):
```

```
        """Return True if the state is a goal."""
        pass

    def load(self, fh):
        """loads a RTB puzzle from the file object fh.
             You may initialize self.initial here."""
        pass

    def h(self, node):
        """This heuristic works like this:
        ...
        It is consistent/not consistent because...
        It is admissible/not admissible because..."""
        pass
```