CSC 7336 : Advanced Software Engineering

J Paul Gibson, D311

paul.gibson@telecom-sudparis.eu

http://www-public.it-sudparis.eu/~gibson/Teaching/CSC7336/

Project

/~gibson/Teaching/CSC7336/CSC7336-Project-2015-16.pdf

The 15-puzzle: software engineering techniques and tools

This assignment is about demonstrating advanced software engineering skills.

You will be judged on your:

- Foundational skills: (50%)
  1. design, 10%
  2. implementation, 10%
  3. code, 10%
  4. tests, 10%
  5. documentation 10%

- Analytic skills: (20%)
  1. Complexity of solution 10%
  2. Simulation and experimentation 10%

- Use of Advanced techniques/tools: (30%)
  (See next slide for details)

The 15-puzzle: software engineering techniques and tools

Use of Advanced techniques/tools: (30%)

There are 6 advanced issues that may be addressed in the project:

1. Aspects : correct use
2. Reflection : correct use
3. AI : implementation of an advanced/standard algorithm
4. Web services : deployment of part of code as a service
5. Device Programming : Code front-end running on a smart device
6. Parallelisation : transforming sequential solution to run on parallel device

For each team, the number of issues addressed must be at least equal to the number of team members. So, a 1 person team must address at least 1 issue, and a 2 person team at least 2, etc…

The 15-puzzle: software engineering techniques and tools

The project can be done in teams (maximum 4 per group) or individually

The deadline for submission is Friday 12th February (there will be no extensions and any work submitted after noon on this day will be awarded a mark of 0).

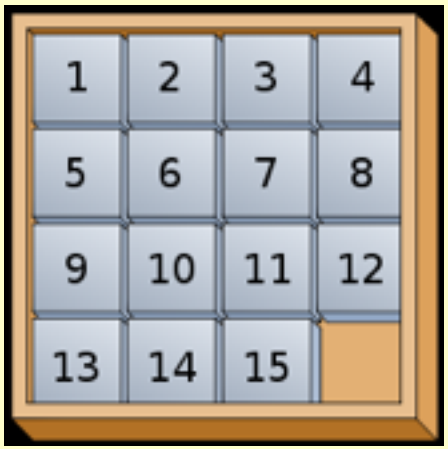
You may submit work before the deadline and then resubmit an improved version later (provided it is before the deadline). I will mark only the last submission received.

If working in a team, every member must summarise their contribution to the project in an appendix to the documentation.

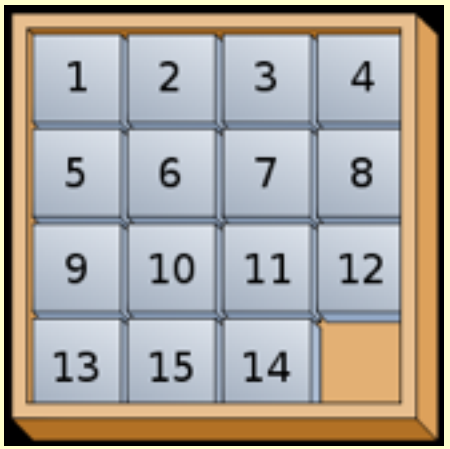
The 15-puzzle

The problem is to implement a 15-puzzle system analyser.

Final Position



Impossible Position



Solvable (valid) Position

The 15-puzzle

Your system is to be parameterised by a *fitness function:*

Input – board state
Output – value between 0 and 1

- 0 is to represent the board in a solved position
- 1 to represent the board in its 'most mixed up' position
- The ordering of the fitness function must rank board positions according to their 'closeness to being solved'

You must code a **generic** puzzle solver that can find a solution (sequence of moves) that returns the puzzle to the solved state from any given valid input state, **using any given fitness function**.

The 15-puzzle

Your generic solution must follow the following steps:

```
Repeat

    •For any given position look for the shortest sequence of
    moves that improves the value of the fitness function
    (towards the solution)

    •Carry out the moves

Until Solved
```

You must also:

Simulate the solution found (sequence of moves) for validation by a human

Test the solution automatically

Output the length of the sequence, and the length of the longest subsequence that was taken during a single step of the algorithm loop.

The 15-puzzle

Analysis and tests:

You must analyse 3 (significantly) different fitness functions with respect to the (sub)sequence solution length values:

- Which fitness functions find the shortest solution?
- Which fitness functions find a solution without having to look too deep during each loop step?
- What is the complexity of each of the 3 solutions (time/ memory)?

The 15-puzzle: submission

- **Full code, design and documentation**
- **Simple (automated) build instructions**
- **Simple test instructions**
- **Analysis report (complexity and performance)**
- **Team contribution breakdown (who did what)**

You should probably create and share a project repository with me (somewhere)