Eleven Tile Puzzle

Solving the eleven puzzle, that is planning a sequence of moves to get from one configuration to another, is a classic study in AI that dates back to the pioneering work of Donald Michie. For the sake of the assessment, the eleven tiles will be made up from two tiles that are labeled a, three tiles that are labeled b, one tile is labeled c and five tiles that are labelled d. The puzzle is arranged as a square but contains 4 cells that are immovable, indicated by +. The blank tile can only be moved horizontally (left and right) and vertically (up and down) but not diagonally in a three by four grid. The blank cannot move into a + cell. The blank tile also cannot wrap around. If the start and destination configuration are respectively:

$$\begin{bmatrix} + & + & d & d \\ + & b & \_ & d \\ + & d & a & b \\ c & b & a & d \end{bmatrix} \quad \begin{bmatrix} + & + & d & d \\ + & b & \_ & d \\ + & d & b & a \\ c & a & d & b \end{bmatrix}$$

then a series of moves between these two configurations is given below:

$$\begin{bmatrix} + & + & d & d \\ + & b & \_ & d \\ + & d & a & b \\ c & b & a & d \end{bmatrix} \begin{bmatrix} + & + & d & d \\ + & \_ & b & d \\ + & d & a & b \\ c & b & a & d \end{bmatrix} \begin{bmatrix} + & + & d & d \\ + & d & b & d \\ + & \_ & a & b \\ c & b & a & d \end{bmatrix} \begin{bmatrix} + & + & d & d \\ + & d & b & d \\ + & b & a & b \\ c & \_ & a & d \end{bmatrix} \begin{bmatrix} + & + & d & d \\ + & d & b & d \\ + & b & a & b \\ c & a & \_ & d \end{bmatrix} \begin{bmatrix} + & + & d & d \\ + & d & b & d \\ + & b & a & b \\ c & a & d & \_ \end{bmatrix}$$

$$\begin{bmatrix} + & + & d & d \\ + & d & b & d \\ + & b & a & \_ \\ c & a & d & b \end{bmatrix} \begin{bmatrix} + & + & d & d \\ + & d & b & d \\ + & b & \_ & a \\ c & a & d & b \end{bmatrix} \begin{bmatrix} + & + & d & d \\ + & d & b & d \\ + & \_ & b & a \\ c & a & d & b \end{bmatrix} \begin{bmatrix} + & + & d & d \\ + & \_ & b & d \\ + & d & b & a \\ c & a & d & b \end{bmatrix} \begin{bmatrix} + & + & d & d \\ + & b & \_ & d \\ + & d & b & a \\ c & a & d & b \end{bmatrix}$$

Directions

You will receive an automatic e-mail detailing 16 files you need to construct. These will be files with names such as `++dd+b_d+dabcbad2++dd+b_d+dbacadb.txt` Such a file should specify a sequence of configurations from the start configuration (`++dd+b_d+dabcbad`) to the destination configuration (`++dd+b_d+dbacadb.txt`). The placement of the + cells, which will be the same for all 16 files that you need to generate. Moreover, for the sake of automatic marking, the file must conform exactly to the submission format:

```
++dd ++dd ++dd ++dd ++dd ++dd ++dd ++dd ++dd ++dd ++dd
+b_d +_bd +dbd +dbd +dbd +dbd +dbd +dbd +dbd +_bd +b_d
+dab +dab +_ab +bab +bab +bab +ba_ +b_a +_ba +dba +dba
cbad cbad cbad c_ad ca_d cad_ cadb cadb cadb cadb cadb
```

When you generate such a file, you are required to use the _ to indicate the position of the blank tile and space as the separator. It is also important, that you only write to the first four lines of the file. The file `++dd+b_d+dabcbad2++dd+b_d+dbacadb.txt` can found at the course webpage so that you can carefully check the formatting. Note in particular the position of the blank columns. Your e-mail will also detail the layout of your puzzle, using the digits 1, 2, 3 or 4 to indicate where tiles can be placed. So the layout of the above puzzle, will be illustrated as follows:

```
++34
+234
+234
1234
```

Your mission, should you decide to accept it[1], is to generate solutions from the given start configuration to the destination configuration, as prescribed by the name of the files mailed to you.

Suggestions and mark scheme

Although this puzzle as a large number of states, your files can be generated with the techniques introduced within Parts I, II and III of the course. Marks will only be awarded for a solution to a problem that is your own (everyone has been allocated a different — but equally taxing — set of problems to solve). Sixteen marks will be awarded for a correction solution to each problem and a further sixteen marks will be awarded for optimal solutions, that is, solutions that minimise the number of intermediate configurations. Because optimal solutions are much harder to find than a solution that is merely a path from a start state to an end state, an optimal solution will be awarded twice as many marks than one that is not.

In addition to placing the 16 solutions in the submission directory (not a subdirectory of this directory), you will also need to submit the source code used to generate your solutions, for which sixteen further marks will be awarded. The source code should also not be placed in a subdirectory. No marks will be awarded for the solutions without the code. Some of these marks will be awarded for the brevity of your solution: this will be measured by the number of non-blank lines of code, so as to reward code that is succinct. Particular attention will be paid to your `next_configs`, so make sure that your code for this method is elegant (my version of this method had less than 20 non-blank lines of code). All source files need to be clearly marked with your login and surname. This code must be written in Java and should be clearly (but not overly) documented. The code should include a `main()` method so that it can be executed without using BlueJ. This is important. Furthermore, the code *must* be submitted in addition to the solutions themselves. You are strongly advised to carefully test your `next_configs` method before deploying it. Note that this submission will be automatically marked and therefore you *must* submit your solution in the correct format. Also, for testing, your code should only use language features supported in Java 1.8.

As requested at the Staff Student Liaison Committee, the assessment comes with a challenge extension, for which just eight marks will be awarded (14% of the total marks). These marks will be awarded for a correct implementation of the A* search algorithm. This amounts to designing a heuristic for this problem that is admissible. As a hint, you may wish to count the number of tiles that are out of place and observe that this number never exceeds the number of moves required to reach the destination configuration. You should not even consider starting this aspect of the assessment until you have got a simpler search algorithm running correctly. In fact, to enforce you to follow this advice, marks will only be awarded for the challenge extension if the submitted codebase also includes working code for a simpler search algorithm too. This is because I want to encourage you to check the solutions generated with your A* search against those generated by the simpler search algorithm.

Submission details and deadline

You need to place your solutions in: `/proj/co528c/michie/`xyz on raptor where xyz is your own personal login. Permissions have been set so that only xyz can access files in the directory xyz. The deadline for submission is *2pm* on Shrove Tuesday, that is, the Tuesday of week 19 (February 28th). Section 2.2.1.1 of Annex 9 of the Credit Framework states that, "Academic staff may not accept coursework submitted after the applicable deadline except in concessionary circumstances".

A Frequently Asked Questions document on Plagiarism and Collaboration is available at: `www.cs.kent.ac.uk/teaching/student/assessment/plagiarism.local`. The work you submit must be your own, except where its original author is clearly referenced. Checks will be run on submitted work in an effort to identify possible plagiarism, and take disciplinary action against anyone found to have committed plagiarism. When you use other peoples' material, you must clearly indicate the source of the material.

[1] This is not meant to imply that the assignment is impossible, or that I will disavow any knowledge of it.