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 the 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 (as in the two wheels puzzle). If the start and destination configuration are respectively:

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

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

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

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

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

Directions

You will receive an automatic e-mail detailing 16 files you need to construct. These will be files with names such as `db**aab*dc_*bddd2ad**db_*cda*bdbd.txt` Such a file should specify a sequence of configurations from the start configuration (`db**aab*dc_*bddd`) to the destination configuration (`ad**db_*cda*bdbd`). They will also specify 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 format:

```
db** db** db** d_** _d** ad** ad** ad** ad** ad** ad** ad**
aab* aa_* a_a* aba* aba* _ba* dba* dba* dba* dba* dba* db_*
dc_* dcb* dcb* dcb* dcb* dcb* _cb* c_b* cdb* cdb* cd_* cda*
bddd bddd bddd bddd bddd bddd bddd bddd b_dd bd_d bdbd bdbd
```

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 `db**aab*dc_*bddd2ad**db_*cda*bdbd.txt` can found at the course webpage so that you can carefully check the formatting. Note inparticular 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:

```
12**
123*
123*
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. Some Windows users have struggled to generate a file such as `db**aab*dc_*bddd2ad**db_*cda*bdbd.txt` because of the * character. To save grief, I recommend, that you instead generate the file where the name replaces * with +, such as `db++aab+dc_+bddd2ad++db_+cda+bdbd.txt`. My test harness will accept both types of file name.

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). You are advised to not leave this assignment to the last minute, as even a tuned search algorithm is unlikely to be able to solve this sort of problem instantaneously.

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 the submission directory), you will also need to submit the source code used to generate your solutions, for which sixteen further marks will be awarded. 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 both elegant and succinct. Again, these source files should not be buried in a subdirectory of the submission directory. 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 are strongly advised to submit your solution in the correct format. Also, for testing, your code should only use language features supported in Java 1.8.0.

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 *midday* on the Tuesday of week 25 (May 10th). After midday, access to these submission directory will be removed.

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: `http://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.