A GUI implementation of puzzle generating six congruent parts in a unordered set of tiles.

Anand Dhandhania (B10053)
School of Computing & Electrical Engineering
Indian Institute of Technology Mandi

03/08/2012

Contents

1 Introduction

In the project, i designed an algorithm to find six congruent parts in a unordered set of tiles using Depth-First search. The algorithm was implemented in C++ and is available in file "structure.cpp". The GUI was written in python using tkinter and the code is availabe in file "gui.py". The runtime of algorithm was observed and it was found that the performance declined steeply after the size of the tiles matrix was increased to 8. In the following section, i will show how the GUI works.

2 How it works !

- The GUI shows N x N buttons with a value "0" which means the tiles are present in this manner. To remove a tile, click on it and the button will now show a value "1" which means the slot is empty.
- The "Execute" button will run the algorithm on input, "Clear" will clean the previous input and "Quit" ends the application.
- If the number of tiles are not a factor of 6, that means the input is not correct and the first two buttons will show a message (Error Occured).
- If the number of tiles are a factor of 6, but there are no solutions to the given input, the first two buttons will show a message (No Solution).

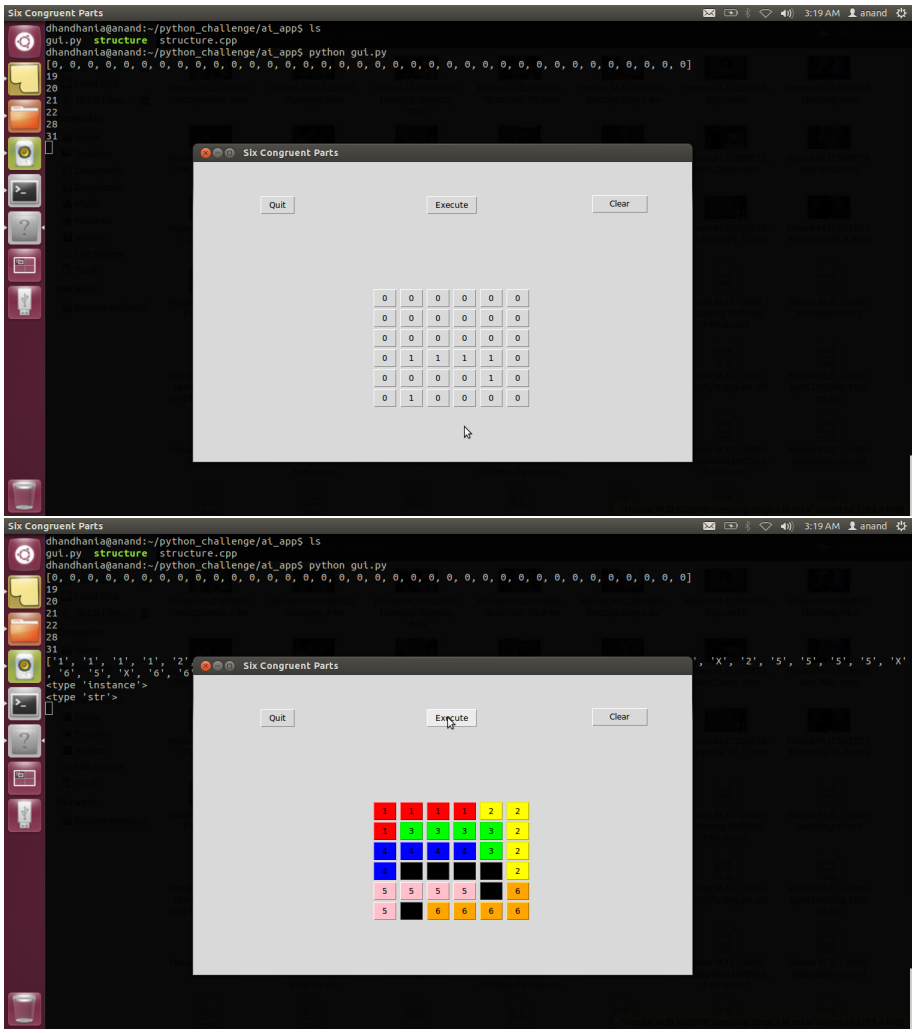

Figure 1: Here we have removed 6 tiles from the 6 x 6 board and the output is shown.

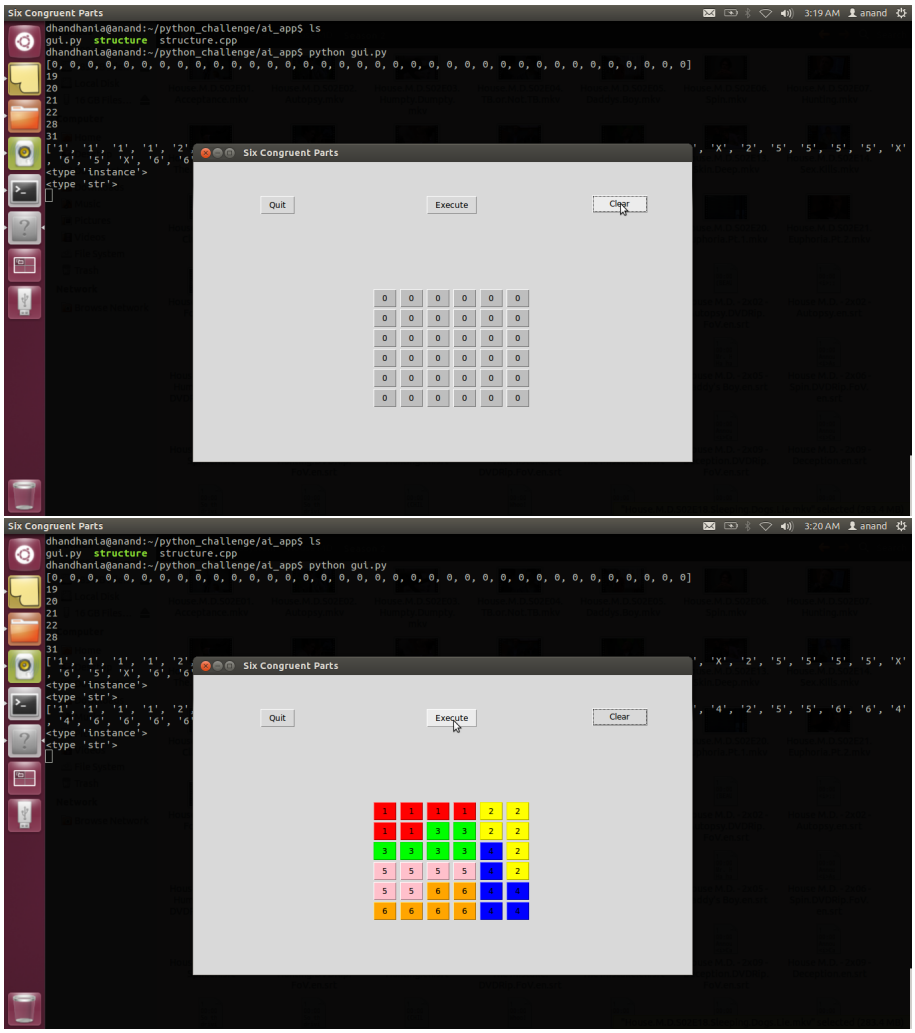

Figure 2: Here all the tiles were present, hence this solution.

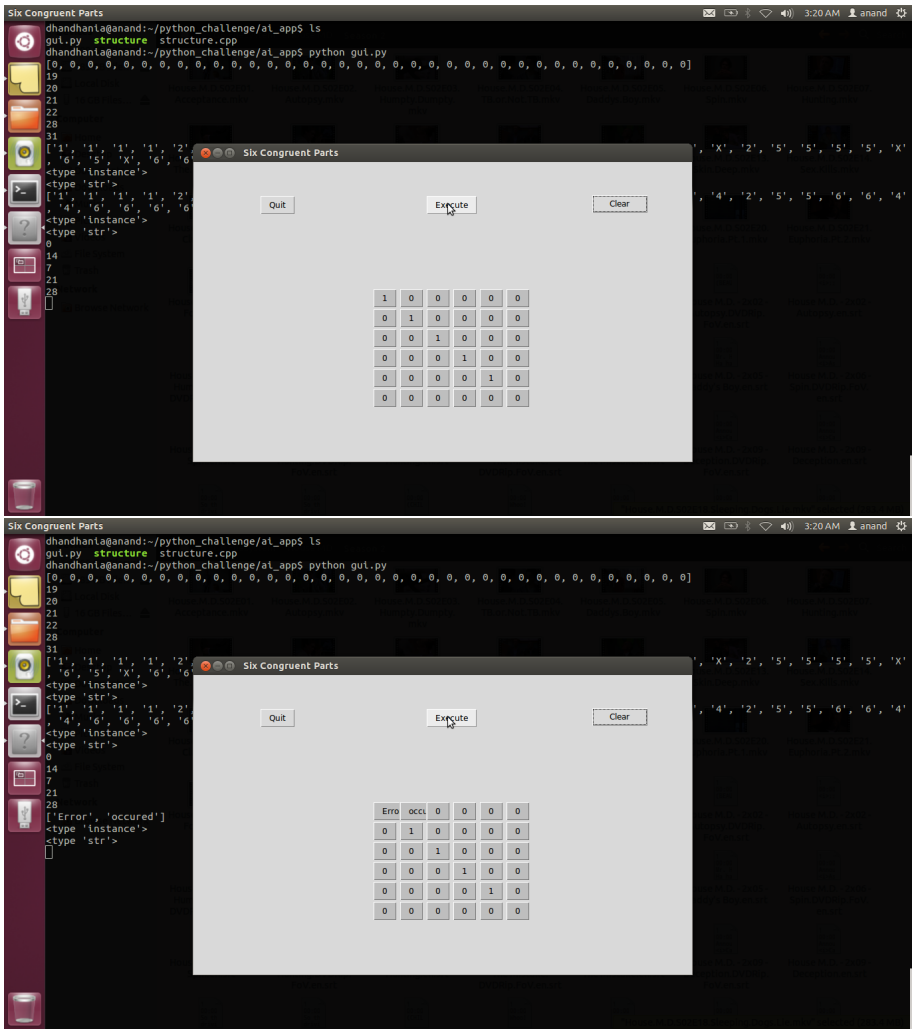

Figure 3: The number of tiles are 31 and hence it's not possible to make 6 congruent parts, hence it shows message "Error occurred"

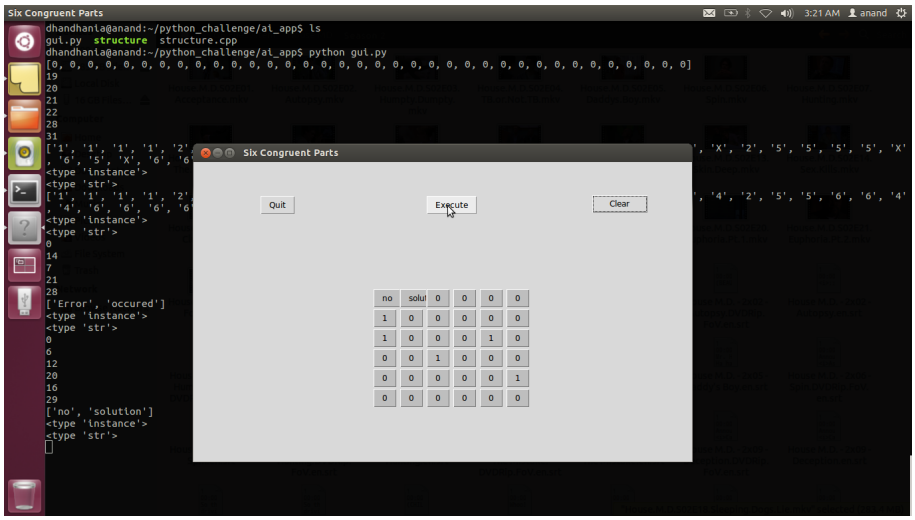

Figure 4: Here the number of tiles are 30, but no solution is possible.

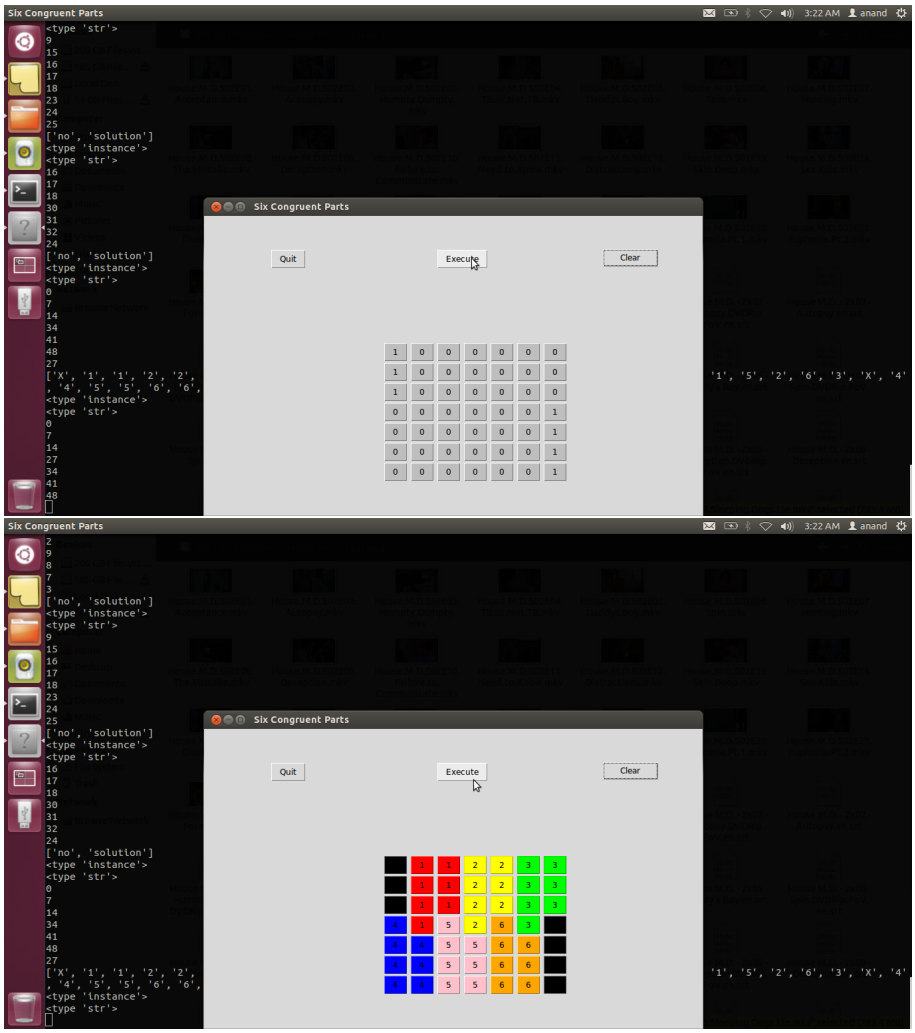

Figure 5: Another implementation with a 7 x 7 matrix

3 Conclusion

Although depth-first search is considered a very fundamental search algorithm, but the problems like this where we need to try a lot of combinations for a particular shape requires a depth first traversal of search tree, where the tree top consists of all the possible shapes that are possible. We could have used iterative deepning which might improve performance for some cases. The codes are attached with the report . For any suggestions / improvements, mail me at dhandhania@live.com . My other projects can be seen at http://students.iitmandi.ac.in/~anand_dhandhania.