PROJECT

DOCUMENT

Team Members:

1) Biswesh Mohapatra IMT2016050

2) Sapparapu Rahul IMT2016036

3) Sriram Marisetty IMT2016120

TABLE OF CONTENT

Topic:

NXN Sliding Puzzle Solver

Overview:

Our project aims to solve a nXn sliding puzzle in real time, providing an easy method for people to understand the logic behind solving a sliding puzzle.

Inspiration:

We decided to select the above topic as it was related to Artificial Intelligence(AI) which was completely based on logic. It was a fascinating topic which led us to select it as our project.

Specifications:

1. Our program is quite flexible and is designed to solve for any NXN sliding puzzle.
2. Initially it takes an integer value as an input from the user to determine the size of the square matrix.
3. Later it takes in two inputs in the form of matrix in text based form which denote the starting and the ending state of the puzzle.
4. The only restrictions are that the user must give a valid puzzle which is solvable, both the states must contain a zero representing the blank space and that the zero should come in the right-bottom corner of the final state.
5. The program shows the number of moves completed before going for the next move.
6. The best part of our program is that it has been done by our team without any help from the internet. Hence the logic used by us is easy to understand and is implemented in a way which is more realistic in nature.
7. The above point makes our program a very nice tool for beginners to master this game within few days.

Design:

- **Concepts used:** recursion, API

- **Approach:** To recursively put the required element and solve the top most row and leftmost column and shortening the matrix size after every step.

- **Implementation:**
    1) Initially our team sat together to decide the logic for the puzzle solver.
    2) It was quite a challenging task and we wanted to approach it in the right direction.
    3) The first and the most important logic that struck us was the use of recursion.
    4) It could be seen that use of recursion makes the program way shorter to implement.
    5) The next challenge was to identify the way to use the recursion as it could be row wise, column wise , upper row and leftmost column wise etc.
    6) Finally we decided to stick with the concept of solving the uppermost row and leftmost column repeatedly till we reach a puzzle of 2X2 dimensions which could be solved

using clockwise movement of zero till we reach the final state(if possible).

7) Hence our method for 4X4 would look like the following-

```
X X X X      - - - -          - - - -
X X X X  ->  - X X X    ->    - - - -        X X <-  2X2
X X X X      - X X X          - - X X        X X     base case
X X X X      - X X X          - - X X                left
```

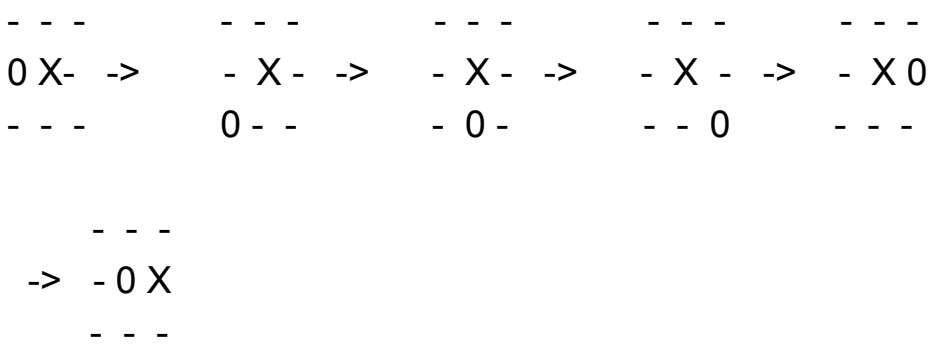
Here (-) means those elements which have come to their right position. We put the elements in their respective placed on by one.

8) Since our project was completely based on logic and mostly used various functions to do the work , we tried to take complete advantage of this fact to implement APIs.
9) Since we had to use lots of functions to do different works, we divided the task of coding them among ourselves.
10) Initially we wrote the main code by which gave us the idea of the functions which we would require, we wrote just the function names along with their arguments and proceeded in the direction of our logic.
11) This created a great platform for us to implement APIs as we now had function names, their functions and the arguments taken already present known to us and all we needed was to make that function work in the way it was required.
12) The project was divided by three of us due to which we had to use three different files module1, module2, module3 and later import them and use them together.
13) These files contain different functions used by us in the program.

14) Using these files also created some problems such as maintaining a global variable for keeping the record of the number of moves.
15) Hence to tackle this problem we had to make some other modules to store such kind of functions and variables.
16) Some of the functions used by us are(arguments have not been given)-
    a) display() - to display the puzzle after every move
    b) swap() - to change the positions of 2 numbers
    c) zero_up() - to move the zero upwards
    d) zero_down() - to move the zero downwards
    e) zero_left() - to move the zero leftwards
    f) zero_right() - to move the zero rightwards
    g) shift _zero() - to move the zero to a
    h) shift_col_left() - to move a particular element to left
    i) shift_col_right() - to move a particular element to right
    j) shift_row_up() - to move a particular element to up
    k) shift_row_down() - to move a particular element to down
    l) first_row() - it arranges the first row elements in the right order
    m) first_column() - it arranges the first column elements in the right order
    n) puzzle_solver() - the main function which gets called recursively to make the solver work properly.

17) The key to implement the puzzle solver accurately and in minimum time was to bring the zero near the element to be moved in minimum possible moves.

18) To do this we had to take into consideration various possible ways in which the zero can reach near the element and then to take the minimum path out of them in such a way that it also makes the element reach its desired position in the least possible moves.
19) The above was the trickiest part of the question which even caused some corner cases to pop up which we had to take care off.
20) We moved the element to be moved by revolving the zero around it in such a way that it moves in the correct direction.



This is the method used by us to move the required element to its right.

21) But again created many exception like we had to check if This move doesn't cause any fixed element to move from its path.
22) Similarly we also have to check if it is not the last row or column
And accordingly we have to decide the correct move.

If possible, we have plans to make a sliding puzzle using graphics. This would be done in another file which would later be added as a module.

Source Used:

As already mentioned, our entire logic was made by ourselves without any internet help,
But for the GUI based program we have taken help of Google, youtube for learning the basics.

Experience Gained:

It was a great experience working as a team and getting to know how things might be functioning in the real life.We learnt to cooperate and also help each other out in the time of need.