225 final project document

By: Sahil Chand

I apologise in advance for how unorganised this documentation is

- **Hours worked** = 28
- I will be using the Assignment 1 solutions for help with parsing the .txt file to a 2D array
- I will start by using a BFS algorithm
- Used manhattan heuristic because of this video
    - https://www.youtube.com/watch?v=LYkbWAQWGro
- I had trouble reading the file and parsing it into the 2d array
    - I stopped using the solution for Assignment 1 and tried coming up with my own
    - That didn't work so I went back and used your solution for Assignment 1
        - I only used the function that does the actual parsing
        - I store the 2d array in a static variable called board, and a lot of my functions use the said board, so that's why I decided to make it static.
        - I made a function that gets the goal state depending on the size
- Used this to understand how to use the A* algorithm for this project
    - https://www.youtube.com/watch?v=dvWk0vgHijs
- Once I thought i got my code working, I got an error saying I ran out of memory
- Tried representing the board with bytes but that didn't work at all(got multiple error)
- Ran into an issue where the output file contained nothing
    - As of right now I have no idea why this is happening
    - Was able to fix it by having a function that contains the tile moved and then another function that says what direction that tile moved
- I will try using priority queues to hold the states of the puzzle
- I got a working code now but it only adds the direction and not the number being moved
    - I worked for an hour trying to fix it but i was unable to
    - I made a new function that gets the position of the tile being moved, and then when I write the solution to the output file, I add that function + the direction to get something like "8 R"

Goals to accomplish:

1. ~~Read the input file and parse into a 2D array~~
2. ~~Make a goal state of the board no matter the size~~
3. ~~Use manhattan heuristic for now~~
4. ~~Make a priority queue and search for most efficient way to reach goal state~~
5. ~~Run A* algorithm~~
6. ~~Write solution to output file~~\

- I completed my goals but it only solves boards 1-7(all the 3x3 ones)
- I used BFS first but I will try to change it to A*
- Manhattan heuristic seems to be too slow(boards are taking too long)
    - Boards 1-7 solve fast but the others dont

- I plan on using MD if the board is 4x4 or 3x3
- I'll look into using another one if the board is larger

My code now solves 4x4 in about 10 seconds, I changed it so that my states I store the manhattan distance instead of calculating it every single time.

I was able to solve board8 in 3 seconds when I used linear conflict instead of manhattan, after looking at TA's slides, learned that manhattan is the reason why my 4x4 take too long to solve. I will try to optimise my manhattan distance heuristic so that it works better on 4x4
I saw in the discord say they tried MD, Missing tiles and Linear conflict so I will try that to see.if it solves 4x4 faster, I implemented it and it cut the time by a bit but it's still not instant. I decided to keep these 3 heuristics as they work sort of, and instead I will look for another heuristic to add. I settled on the max heuristic and when I used it, I was able to solve some more 4x4 boards.

I changed it so it doesn't recalculate the cost every time and now it solves 4x4 fast, except for boards 8, 12, 13, 14, I tried euclidean heuristic but it made it even slower. Tried using max heuristic and it solved every 4x4 board instantly except for board 14. Tried using inverse count but didn't work.. Right now I have MD, LC, MT and Max and it solves every 4x4 board except 14. I learned when I don't use misplaced tiles, board8 solves instantly but board 10 takes longer.

Right now working with max, MD, LC and misplaced tiles, but for board 10 and 12 im not using misplace tiles, I got board 14 working with max, linear conflict, blocking and gaschnig. I tried changing my algorithm from A* to PHS but I didn't see any noticeable differences, so maybe I didn't do it right.

I will now move onto 5x5 boards: Since I need to use different heuristics to solve 5x5 than 4x4, I have an if statement in my state class saying if size == 5, Board 15 works with lc, blocking, max, gaschnig and walking, however the other 5x5 boards don't work so I need to experiment with other heuristics. I tried using euclidean and edge tiles as ny heuristics and they just slowed down the runtime a lot. I tried combining them with manhattan, lc and max but I got an error saying out of heap space. I saw in the discord that people were trying 0.5manhattan and stuff and it got my eye and I decided to try it out and see if I could get board16-20 working. When I did 0.2manhattan, it made my board15 solve from 1 second to like 10 seconds. I did the max heuristic divided by 2 but it just slowed things down for me. My goal is to at least get 3 more 5x5 boards working so that I can improve my mark. Since my board 15 works with MD, LC, Max, gaschnig and blocking heuristic, I'm assuming boards 16-20 will require something similar.

Final version: All 3x3 boards and all 4x4 boards except board 10, 12 and 14 solve when I use the heuristics: Manhattan distance, linear conflict and max. To solve the board 10 and 12 I just added misplaced tiles and for board 14 I added gaschnig and blocking. For 5x5 boards; I use the heuristics manhattan, linear conflict, gashnig, blocking and max. I used an A* algorithm and I first parse the textfile into a static int[][] called board. From this board, I get the size and now I can use the size to use other functions. For example I have a function that gets the goal state by utilising a nested loop printing numbers from 1 to size*size-1, and

then making the last tile a zero. I constantly compare states to this goal state to see if the puzzle is solved or not. I have a state class that calculates the heuristic value, as well as the moves and previous states. I have many heuristics in this class: manhattan, linear conflict, misplaced tiles, max, blocking ,gaschnig, walking and euclidean. I have a function that gets the next states, by locating the position of the blank tile and then moving it in a way that brings us closer to the solution. It also makes a new board with the updated move, the states are stored in a hashset. In my main function, I call my function to get my board from the textfile, then I compute the goal state. I start with an initial state and goal state. I then have a function that writes the solution to an output file. My solve function has a priority queue that adds in the initial state. Adds the new states until the goal state is reached. I also have a function that just searches for the blank tile, basically a nested loop that searches for zero as that is what the blank tile is represented as. I also have a function that gets the direction depending on the move, so it will either return L R U D.