Trace Of A* Search

1. userInput.py prompts to select search, choice is 2
2. userInput.py prompts to select heuristic (options 1/2/3/4), can choose all 3 (option 4)
3. userInput.py prompts to input string representing solitaire game (ex. < - - 0 0 0 - -, - - 0 X 0 - - , 0 0 X X X 0 0, 0 0 0 X 0 0 0, 0 0 0 X 0 0 0, - - 0 0 0 - -, - - 0 0 0 - - >)
4. Calls AStarSeach.aStarSearch(heuristic, input)
5. Function sees if input is answer (functionsForBothSearches.isanswer(input)) or blank (functionsForBothSearches.checkIfXs(input))
6. Generates start and end locations of for loop based on heuristic value
7. Goes into for loop
8. Generates cost based on i value in for loop (which is based on heuristic)
9. Appends initial input in queue as dictionary with values for input, moves, and depth
10. Goes into while loop which continues until answer does not exist or is found
11. Searches queue for location of minimum cost
12. Checks if this node is the answer (functionsForBothSearches.isanswer(input))
13. If it is, goes to functionsForBothSearches.printSolutions(solution, i, inputAnswer) and functionsForBothSearches.printAdditional(startTime, numberOfNodes, memory)
14. If it's not, produces children of least cost node (functionsForBothSearches.nextMoves(input))
15. Does the following steps for each of the children (steps 16-24):
16. Produce a new modified list from the move of the child (functionsForBothSearches.produceNewInputList(input, move))
17. If the modified list is already in a list of explored nodes, it skips steps 18-24, otherwise it continues
18. Generates cost for each child, given by the depth plus the heuristic value (either manhattanDistance(input), numberOfMoveablePegs(input), or DistanceFromCenter(input))
19. If manhattanDistance(input) will call xYPositions(input) and find the xy position of each x value. After such, it will find the distance from one another and return the sum of such.
20. If numberOfMoveablePegs(input), will call functionsForBothSearches.nextMoves(input) and return count of possible next moves
21. If DistanceFromCenter(input) will go through input, see where an "X" value is and will calculate the difference from the (3,3) center positon.
22. Then will calculate the moves needed to get to the child
23. The will create a dictionary value of the child with attributes "INPUT", "MOVES", and "DEPTH"
24. Will add list representing child to list of explored nodes
25. Will remove the previously lowest cost node in the queue
26. Goes back to step 11 until answer does not exist or is found
27. Goes back to step 8 until all heuristics queried for are completed