EE4483 Heuristics Assignment using 8 puzzle problem

Name: Lim Jia Song John

Matric: U1321554J

# Heuristic measure and search strategy

A heuristic has the evaluation function:

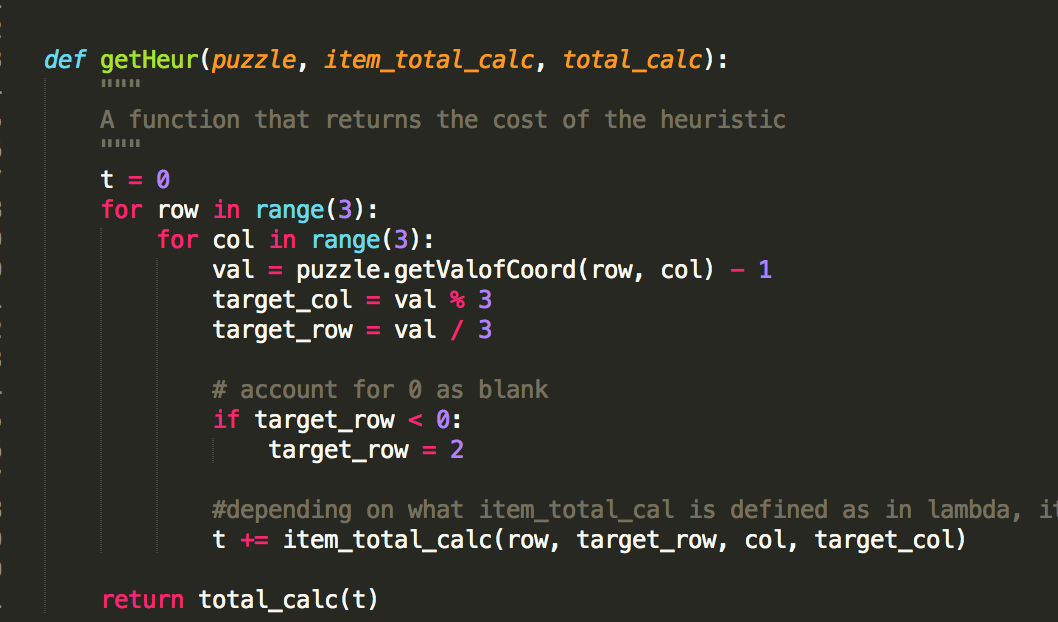
F(n) = g(n) + h(n)

G(n) is the number of steps of the new state from the start state.

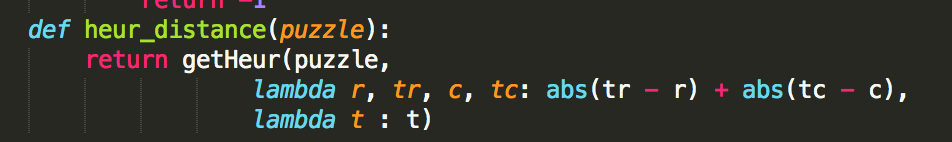
H(n) is the number of tiles out of place.

It considers all possible paths to the goal, and considers the path that appears to have the least computational cost, where computational cost, f(n) is the sum of g(n) and h(n).

By saving each solution state of the 8 puzzle problem as a node, we can calculate f(n) with each path taken by the algorithm. It makes a decision whether or not to continue exploring down the 8-puzzle node, or to begin exploring another possible node that has a lower computational cost. A solution is guaranteed so long as the inputs are valid and can be solved.



In this case, item\_total\_calc is defined by a python in-line function using lambda. The distance heuristic is simply the absolute value of the difference between the target row/column and the actual row/column of the value.



# Time and space complexity

The A\* algorithm is a best first search algorithm which works similarly to a breadth first search, only that the path to the solution is guided by calculating the computational cost at each node. If a constant heuristic were used, the A\* algorithm is no different from a BFS, which has a time complexity of O(v+e), where v is the number of vertexes (nodes) and e is the number of edges.

Consider the following states:

|  |  |  |
| --- | --- | --- |
|  |  |  |
| The number of edges is 4, because the black square can swap in all 4 directions. | The number of edges is 3, because the black square can swap only to the bottom, left and right. | The number of edges is 2, because the black square can swap only to the top and to the left. |

The search tree has a branching factor of 4 in its worst case and 2 in its best case.

To consider the worst case complexity, we take e=4. Yet, the worst case time complexity can only happen when the black square is in the middle.

Let n be the length of the solution path. The worst case time complexity is hence O(n^4).

In this case though, optimal heuristic is used, in which the worst case time complexity is O(n).

The space complexity also depends on how far we traverse down the list. It is exponential as it stores all the nodes as we traverse down the list.

The number of nodes stored in memory is b^n, having a space complexity of O(b^n), or O(4^n) in this case, as the worst case b = 4.