Trinity Belen

Comp 157

Professor Hayes

19 February 2019

Project 1: Tile Puzzle

**Pseudo-Code**

def bfs(tiles, Astar, dim)

#Input: tiles, Astar, dim

#output: A deque of the steps that searches through parents and then their children, and then #children of the children to solve the puzzle and will return “unsolvable” whether or not it didn’t #for reasons specified later on.

tile\_arr <- [] #the array of tiles, which will be appended with tiles

directions <- [] #will be append with tiles

visited <- [] ##will be append with white space

while tile\_arr:

rows, col <- find\_x(tiles, dim)

if (col + 1 > dim):

copytiles <- copy.deepcopy(tiles)

temp <- traverse\_board(copytiles, rows, col, 'D')

if temp not in visited:

tile\_arr <- append(temp)

visited <- append(temp)

directions <- append(dir + "Down ")

#These if statements do the same thing as the if statement right above except they are up, #right, left respectively (not going to copy all the nearly same code right here)

if (rows - 1 < 0):

if (col + 1 < dim):

if (col - 1 >= 0):

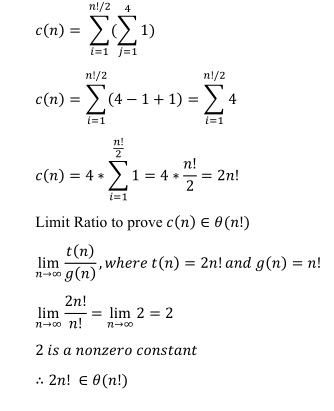
return “Unsolvable”

#Will return unsolvable regardless of whether or not the puzzle was originally solvable since if it #reaches the solution, it will not go through the children anymore, which means it becomes #unsolvable, but shows the solved puzzle

# If it has no solution, it will show the original puzzle and not the solved puzzle

**Efficiency Class**

An instance where it’s a n puzzle and it searches through 4 moves.



Its class is factorial, which is “Typical for algorithms that generate all permutations

of an n-element set,” according to the textbook.