Inversions and Manhattan distances

Gabor – AI – 27 Sep 2019

1) Implement a function to determine whether a slider puzzle is possible or not. Your function should work with at least 3x3 and 4x4 sizes of sliders.

2) Generate 500 random 3x3 slider puzzles and solve them (or exit after 90 seconds, whichever comes first). Hint: you might use `random.shuffle` on `sPzl = [*"12345678_"]`. The first thing that your `solve()` routine should do is check to see whether the slider puzzle is impossible and if so, return right away. You should output (and then record) three statistics:

1. The total number of impossible puzzles (should be around 250)
2. The average length of solvable puzzles (should be around 22)
3. The total time taken to solve all the puzzles

3) Create a separate script that accepts a single argument, a file that holds a list of sliders to solve. These will be 4x4 sliders that Mr. Eckel has generously donated (thanks Mr. Eckel!). Have your code run for 90 seconds where you print out the number of steps it takes to solve each slider puzzle along with the amount of time your script took to solve it. Once your code solves a puzzle past the time just mentioned, your script should end.

Item 2 and 3 above are your baseline scripts. You'll now investigate what happens when you include Manhattan distance in your script. To be clear, the purpose of the below is make sure that you have a plausible Manhattan distance implementation rather than a correct search routine.

```
def solve(root, goal):
  if root == goal: return [root]
  if impossible(root, goal, width): return []

  parseMe = [(ManhattanDist(root,goal,width), root)]
  dctSeen = {root: ""}

  while parseMe:
    md, pzl = parseMe.pop(0)            # pop from beginning
    for nbr in neighbors(pzl):
      if nbr in dctSeen: continue  # already handled

      dctSeen[nbr] = pzl            # else, prepare to handle
      mdn = ManhattanDist(nbr,goal,width)
      parseMe += [(mdn,nbr)]  # it by adding it to end of Q

      if nbr == goal: return getPath(goal, dctSeen)[::-1]
    parseMe.sort()
```

There is a potential gotcha in the code above which is that if you have implemented `getPath()` recursively, you might be in for a bit of a shock. If that happens, rewrite `getPath()` using a while loop.

4 & 5) You should rerun item 2 and 3 using the code above to see how things change. Don't worry about doing things better or faster or different – you're just getting an initial impression. We'll discuss what's hapening on Tuesday of the upcoming week.