PAOLA TERRAZAS

CS2302

CREATING A MAZE USING DISJOINT SET FORESTS

DR. FUENTES

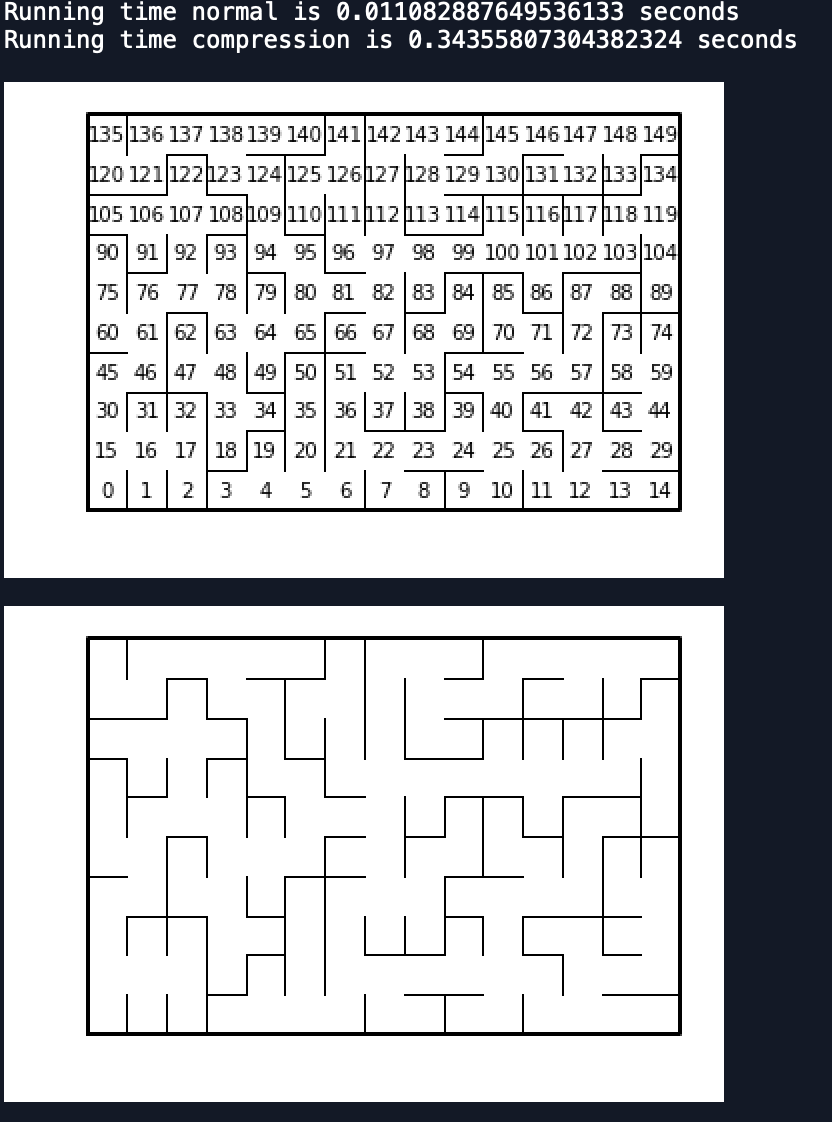
TA - ANINDITA NATH

4.16.2019

LAB 6 REPORT

In our sixth lab we were asked to use a disjoint set foreste to build a maze. Our maze had to contain a collection of cells separated by walls in such a way that there is exactly one simple path separating any two cells. I created my maze by adding a few functions to the sample code Dr. Fuentes gave us. I began with my first method, numOfSets(S) which counted the number of sets in the disjoint set forest, it took the disjoint set forest as a parameter and then I gave it a counter that was increased by 1 as it traversed through the forest, returning the counter. My second method was compMaze, which took the number of columns, rows, wall list and disjoint set forest as parameters. It checks the number of sets in the forest and while it is greater than one set it creates an integer picked at random. If roots aren’t equal, it will give the cells the same root then it will randomly pop a wall between two cells. This was a little challenging method for me, I did have a little bit of trouble figuring it out, going from pseudocode to python but after several tries, it worked. My third method was normalMaze, it has a while loop that uses the method I created known as numOfSets to check if the number of sets in the disjoint set forest are greater than 1, if so, I created a variable known as c to hold a random integer. It will check if the roots are equal, if they are not it uses the union methods to make them equal and then randomly pops a wall between the cells. The codes are similar in both normalMaze and compMaze, once I got one down, it was very simple to implement the second one.

My results are as follows:



For my Big O, my three methods were constant, resulting in O(n) since they are all simple iteration. This lab helped me to understand and work with disjoint set forests better.

#SOURCE CODE

# PAOLA TERRAZAS

|  |  |
| --- | --- |
|  | # CS2302 |
|  | # Create mazes |
|  | # Disjoint set forests |
|  | # DR. FUENTES |
|  | # TA - ANINDITA NATH |
|  | # 4.14.2019 |
|  |  |
|  | import matplotlib.pyplot as plt |
|  | import numpy as np |
|  | import random |
|  | import time |
|  |  |
|  | def draw\_maze(walls,maze\_rows,maze\_cols,cell\_nums=False): |
|  | fig, ax = plt.subplots() |
|  | for w in walls: |
|  | if w[1]-w[0] ==1: #vertical wall |
|  | x0 = (w[1]%maze\_cols) |
|  | x1 = x0 |
|  | y0 = (w[1]//maze\_cols) |
|  | y1 = y0+1 |
|  | else:#horizontal wall |
|  | x0 = (w[0]%maze\_cols) |
|  | x1 = x0+1 |
|  | y0 = (w[1]//maze\_cols) |
|  | y1 = y0 |
|  | ax.plot([x0,x1],[y0,y1],linewidth=1,color='k') |
|  | sx = maze\_cols |
|  | sy = maze\_rows |
|  | ax.plot([0,0,sx,sx,0],[0,sy,sy,0,0],linewidth=2,color='k') |
|  | if cell\_nums: |
|  | for r in range(maze\_rows): |
|  | for c in range(maze\_cols): |
|  | cell = c + r\*maze\_cols |
|  | ax.text((c+.5),(r+.5), str(cell), size=10, |
|  | ha="center", va="center") |
|  | ax.axis('off') |
|  | ax.set\_aspect(1.0) |
|  |  |
|  | def wall\_list(maze\_rows, maze\_cols): |
|  | # Creates a list with all the walls in the maze |
|  | w =[] |
|  | for r in range(maze\_rows): |
|  | for c in range(maze\_cols): |
|  | cell = c + r\*maze\_cols |
|  | if c!=maze\_cols-1: |
|  | w.append([cell,cell+1]) |
|  | if r!=maze\_rows-1: |
|  | w.append([cell,cell+maze\_cols]) |
|  | return w |
|  |  |
|  | def numOfSets(S): |
|  | n =0 |
|  | for i in range(len(S)): |
|  | if S[i] <0: |
|  | n+=1 |
|  | return n |
|  |  |
|  |  |
|  | def DisjointSetForest(size): |
|  | return np.zeros(size,dtype=np.int)-1 |
|  |  |
|  | def find(S,i): |
|  | if S[i]<0: |
|  | return i |
|  | return find(S,S[i]) |
|  |  |
|  | def find\_c(S,i): |
|  | if S[i] <= 0: |
|  | return i |
|  | s = i |
|  | while S[i] >= 0: |
|  | i = S[i] |
|  | root = i |
|  | while S[s] >= 0: |
|  | p = S[s] |
|  | S[s] = root |
|  | s = p |
|  | return root |
|  |  |
|  | def compMaze(row,col,S,walls): |
|  | while numOfSets(S)>1: |
|  | c = random.randint(0,(len(walls)-1)) |
|  | if find\_c(S,walls[c][0])!= find\_c(S,walls[c][1]): |
|  | union(S,walls[c][0],walls[c][1]) |
|  | walls.pop(c) |
|  |  |
|  |  |
|  | def union(S,i,j): |
|  | ri = find\_c(S,i) |
|  | rj = find\_c(S,j) |
|  | if ri!=rj: |
|  | S[rj] = ri |
|  |  |
|  |  |
|  | def normalMaze(row,col,S,walls): |
|  | while numOfSets(S)>1: |
|  | c = random.randint(0,(len(walls)-1)) |
|  | if find(S,walls[c][0])!= find(S,walls[c][1]): |
|  | union(S,walls[c][0],walls[c][1]) |
|  | walls.pop(c) |
|  |  |
|  |  |
|  |  |
|  | plt.close("all") |
|  | maze\_rows = 10 |
|  | maze\_cols = 15 |
|  |  |
|  | walls = wall\_list(maze\_rows,maze\_cols) |
|  | S = DisjointSetForest(maze\_rows\*maze\_cols) |
|  |  |
|  | starttime = time.time() |
|  | normalMaze(maze\_rows, maze\_cols,S,walls) |
|  | endtime = time.time() |
|  |  |
|  | start = time.time() |
|  | compMaze(maze\_rows, maze\_cols, S, walls) |
|  | draw\_maze(walls,maze\_rows,maze\_cols,cell\_nums=True) |
|  | draw\_maze(walls,maze\_rows,maze\_cols) |
|  | end = time.time() |
|  |  |
|  |  |
|  |  |
|  | print('Running time normal is', endtime-starttime, 'seconds') |
|  | print('Running time compression is', end-start, 'seconds')  Scholastic Dishonesty  Any student who commits an act of scholastic dishonesty is subject to discipline. Scholastic dishonesty includes, but not limited to cheating, plagiarism, collusion, the submission for credit of any work or materials that are attributable to another person.   * **Cheating**   + Copying form the test paper of another student   + Communicating with another student during a test   + Giving or seeking aid from another student during a test   + Possession and/or use of unauthorized materials during tests (i.e. Crib notes, class notes, books, etc)   + Substituting for another person to take a test   + Falsifying research data, reports, academic work offered for credit * **Plagiarism**   + Using someone’s work in your assignments without the proper citations   + Submitting the same paper or assignment from a different course, without direct permission of instructors * **Collusion**   + Unauthorized collaboration with another person in preparing academic assignments   Sign \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date: 4/16/2019 |