BT 3051 — Data Structures and Algorithms for Biology

Jul–Nov 2019

Assignment 3

2nd November 2019

Due date: 13th November, 2019 @ 17:00 **Maximum marks: 60**

Instructions: Write Python codes to solve the problems mentioned below. If you need any assistance, feel free to write to me or the TAs via Piazza (private note). Evaluation will be based on the codes and the logic.

Academic Integrity: You are allowed to discuss the problems verbally with your friends, but copying or looking at codes (either from your friend or the Web) is not permitted. Transgressions are easy to find, and will be reported to the "Sub-committee for the Discipline and Welfare of Students" and will be dealt with very strictly. Mention any collaboration (discussions only!) in your solutions.

Late submission penalties: 1 second – 24 h: 20%; 24–48 h: 40%; > 48h: 60%

Early submission bonuses: > 24h: 5%, > 48h: 10%, > 72h: 20%

Evaluation: Assignments will be evaluated by the TAs within two weeks of the due date. You can check out your marks and contest them, if needed, for at most one more week post-evaluation, i.e. three weeks from the due date of the assignment.

Problem Statement

A. (15 marks) **Generate a regular lattice.**

Write a Python program to create a Regular Lattice graph. You are allowed to use the NetworkX package, but only the basic functions — you cannot use any graph generation routines that are built in. A regular lattice is one in which each node is connected to k nearest neighbours on either side – connecting each node to 2k other nodes effectively. Construct a regular lattice graph with 50 nodes and k=3. Plot the graph within the function, as given in the template.

B. (45 marks) **Maze.**

Given a matrix of '1's and '0's that represents a maze, and the indices of the starting and finishing points, generate a NetworkX graph where each node represents a '1' in the input matrix, and print:

a) A list of indices in the shortest path solution to the maze (in the order of the path)

b) A list of lists of indices in the isolated components of the maze (order doesn't matter)

- '0's in the input represent walls (cannot be part of a path) and '1's represent open points (allowed in paths). (Visualise the matrix like a crossword grid, where '0's are the darkened out blocks.)
- Traversal is allowed up, down, left, and right - diagonally adjacent '1's cannot form a path.
- The matrix is indexed from (0,0).
- Label the nodes of the graph as tuples of their indices in the input matrix.
- A path is a series of '1's that can be reached with the allowed travel directions.
- An isolated component is a group of '1's that can't reach other '1's via allowed travel directions.

(a) Use a **BFS** algorithm on the graph data structure. Assume the maze has at least one solution. The algorithm should work even if there are loops, as below.

(b) Use a **DFS** algorithm on the graph data structure. *Hint: iterate this repeatedly until all the components are identified.*

You are allowed to use only the basic functions in NetworkX, you cannot use built-in graph generation routines or traversal/shortest-path functions.

Follow the template attached without modifying the function names and arguments.

Example:

```
Input:        [[1 , 0 , 1 , 1 , 0 , 1],
               [1 , 1 , 0 , 0 , 0 , 0],
               [0 , 1 , 0 , 1 , 1 , 1],
               [0 , 1 , 1 , 1 , 0 , 1],
               [1 , 0 , 0 , 1 , 1 , 1],
               [1 , 1 , 0 , 0 , 0 , 1],
               [0 , 0 , 1 , 1 , 0 , 1]]

             Start: (0,0)
             Finish: (6,5)
```

Output:

```
a) [(0, 0), (1, 0), (1, 1), (2, 1), (3, 1), (3, 2), (3, 3), (4,
3), (4, 4), (4, 5), (5, 5), (6, 5)]

b) [ [(0, 0), (1, 0), (1, 1), (2, 1), (3, 1), (3, 2), (3, 3), (4,
3), (4, 4), (4, 5), (5, 5), (6, 5), (3, 5), (2, 5), (2, 4), (2,
3)], [(0, 2), (0, 3)], [(0, 5)], [(4, 0), (5, 0), (5, 1)], [(6,
2), (6, 3)] ]
```

How to Submit your Homework

- Use the template files provided
- Submit your assignment ONLY via the submission link: http://tinyurl.com/bt3051-submit.
- You should not be signed into Dropbox while uploading this file (or use an incognito window to open the link), so that you can enter the following details during submission, instead of Dropbox auto-filling it:
  - First Name: Roll Number
  - Last Name: Your Full Name
  - E-mail: Your `smail` id
- Save your solution files as `hw3a.py` and `hw3b.py`. Do not use different filenames!
- Each of your submission files, `hw3a.py` and `hw3b.py` should begin with the usual **header information** — the number of the assignment, your roll number, your collaborators' roll number(s), and approximately how much time you took to solve the problems in that part of the assignment.
- **Submissions not adhering to any of the above instructions will not be evaluated**.
- Also do not send the files by e-mail — obviously, they will not be evaluated.

```
#BT3051 Assignment 1a
#Roll number: BE13B001
#Collaborators: CH12B001, EE13B001
#Time: 1:15
```