Khrystian Clark

CS 325, Assignment 7

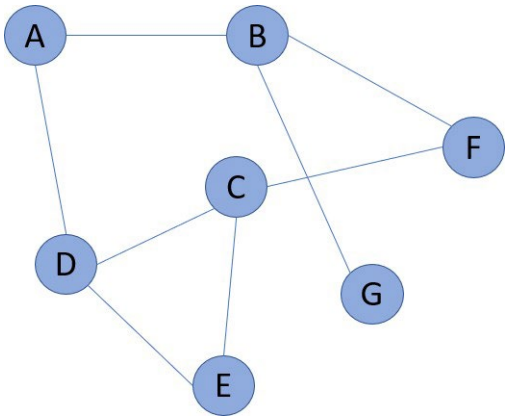
22FEB2022

Assignment: Graph Algorithms – I

Note: The problem 2 is to be discussed as part of the Group Assignment. (Check this week's Group Assignment on Canvas for details).

The questions asked in this assignment – code implementation and time complexity of your code should be done individually based on the problem-solving strategy discussed within your group.

1. **Write BFS and DFS for a graph**: What would be BFS and DFS traversal for the below graphs. Write the nodes for BFS and DFS. Start at node A.



- BFS: A B D F G C E
  - Start: A→B D
  - B →F G (taking away the front B)
  - D → C E (taking away the front D)
- DFS: A B F C D E G
  - A → B → F → C → D → E → Since all that left is G

2. **Apply BFS/DFS to solve a problem**

You are given a 3-D puzzle. The length and breadth of the puzzle is given by a 2D matrix puzzle[m][n]. The height of each cell is given by the value of each cell, the value of puzzle[row][column] give the height of the cell [row][column]. You are at [0][0] cell and you want to reach to the bottom right cell [m-1][n-1], the destination cell. You can move either up, down, left, or right. Write an algorithm to reach the destination cell with minimal effort. How effort is defined: The effort of route is the maximum absolute difference between two consecutive cells.

If a route requires us to cross heights: 1, 3, 4, 6, 3, 1

The absolute differences between consecutive cells is: |1-3| = 2, |3-4|=1, |4-6|=2, |6-3|=3, |3-1|=2; this gives us the values: {2, 1, 2, 3, 2}. The maximum value of

these absolute differences is 3. Hence the effort required on this path will be: 3.

Example:

Input: puzzle[][] = [[1, 3, 5], [2, 8, 3], [3, 4, 5]]

Output: 1

Explanation: The minimal effort route would be [1, 2, 3, 4, 5] which has an effort of value 1. This is better than other routes for instance, route [1, 3, 5, 3, 5] which has an effort of 2.

1	3	5
2	8	3
3	4	5

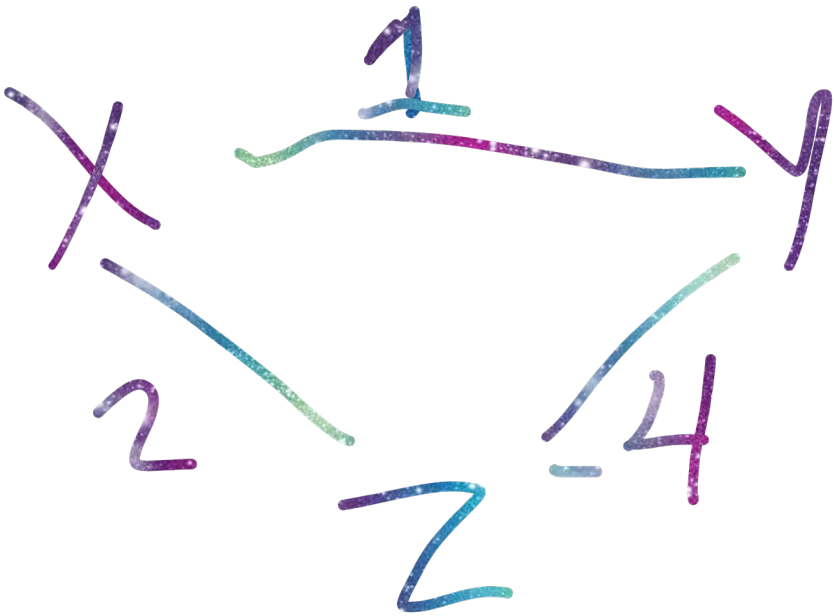
a. Implement the algorithm. Name your function **minEffort(puzzle)**; puzzle will be in the form of an 2D matrix as shown in the above example. Name your file **MinPuzzle.py**

- **See approach code in MinPuzzle.py file**
- **Using DFS:**
  - **minEffort(puzzle)**
    - **startpoint = puzzle[0][0]**
    - **path list = [] #create a holder list for the path we will return**
    - **counter = 0 #create a count placeholder for the amount of "effort" we have.**
    - **Directionally compare which next placement would take more effort.**
      - **If (puzzle[m+1][n]-puzzle[m][n]) > (puzzle[m][n+1]-puzzle[m][n])**
        - **nextPoint = puzzle[m][n+1]**
        - **add nextpoint to path**
        - **increment counter with (puzzle[m][n+1]-puzzle[m][n])**
      - **If (puzzle[m+1][n]-puzzle[m][n]) < (puzzle[m][n+1]-puzzle[m][n])**
        - **nextPoint = puzzle[m+1][n]**
        - **add next point to the path**
        - **increment counter with (puzzle[m+1][n]-puzzle[m][n])**
      - **(need to come up with the case if they are equal)**
        - **Add the placement to the**
    - **Repeat until we reach the bottom right cell (puzzle[m-1][n-1])**
    - **Return path list and counter.**

b. What is the time complexity of your implementation?

- O(|V| + |E|)

3. **Analyze Dijkstra with negative edges**: Analyze with a sample graph and show why Dijkstra does not work with negative edges. Give the sample graph and write your explanation why Dijkstra would not work in this case.
   a. **You run the risk of the algorithm not recognizing the negative value and/or producing the incorrect value as the optimal path.**
   b. **See below: The algorithm would be tricked and not recognize Y connection to Z because it would see the negative value as less than zero.**



4. **(Extra Credit): What would be BFS and DFS traversal in below puzzle. Start at node A.**

A	B	C	
		D	E
	F	G	
	H	I	J

- BFS: A B C D E G F I H J
  - Choosing the next at the most shallow depth/staying on the most convenient level.
- DFS: A B C D G I J H F E
  - Deepest first, then the adjacent.