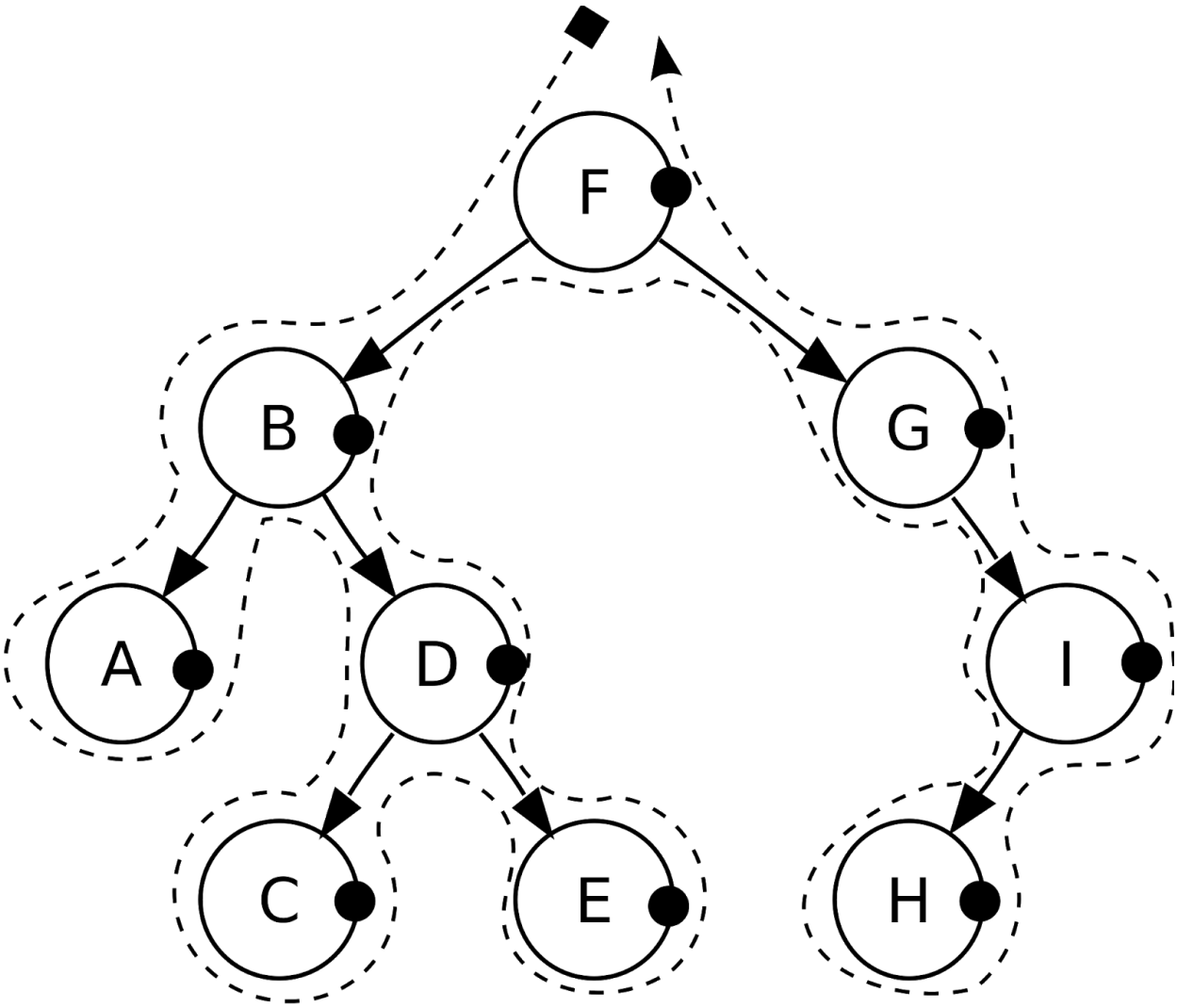

Traversal (DFS, DFS on Edges)

10.25.2018

Omar Vega
17549982

Overview

Traversing weighted graph using the Depth First Search algorithm

Subproblems

1. Iterate over edges in a depth-first-search (DFS)
    a. Given a graph, traverse the graph using DFS with a specified depth limit and return a generator of the edges traversed in order.
2. Return oriented tree constructed from a depth-first-search from source
    a. Use DFS traversal to construct an oriented tree with the edges traversed
3. Return dictionary of predecessors in depth-first-search from source
    a. Create a dictionary whose key is and edge encountered in a DFS and whose value is the key's predecessor (the edge that was encountered directly before it)
4. Return dictionary of successors in depth-first-search from source
    a. Create a dictionary whose key is and edge encountered in a DFS and whose value is a list of the key's successors (the edges that was encountered directly after it)
5. Generate nodes in a depth-first-search pre-ordering starting at source
    a. return a generator of edges traversed in pre - order (in order of the first time an edge was encountered).
6. Generate nodes in a depth-first-search post-ordering starting at source
    a. return a generator of edges traversed in post - order (in order of the last time an edge was encountered).
7. Iterate over edges in a depth-first-search (DFS) labeled by type
    a. return a generator of edges traversed, along with a label indicating the directionality of the edges being traversed. 'forward', 'nontree', or 'reverse'.
8. A directed, depth-first traversal of edges in G, beginning at source
    a. Returns a generator of edges traversed in DFS that does not stop once all the edges are visited, so it provides the edges traversed when "backtracking" after reaching the depth.

Milestones

I. Week 4

A. Proposal for project

II. Week 5

A. Build problem solver using Python:

1. Parse CSV file to build graphs
2. Use NetworkX algorithms to solve subproblems

III. Week 6

A. Complete problem solver

B. Build test cases to test problem solver

IV. Week 7

A. Abstract the interface to handle and “API” to prepare for web interface and group integration

B. Build functionality for returning visualized result

V. Week 8

A. Start to build web interface using Python CGI

1. Allow uploading of CSV graph file
2. Allow manual input of node/edges

VI. Week 9

A. Continue to work on web interface

1. Use JS / CSS to build interactive visual graph result

VII. Week 10

A. Finish web interface,

B. Document procedures for operation, testing