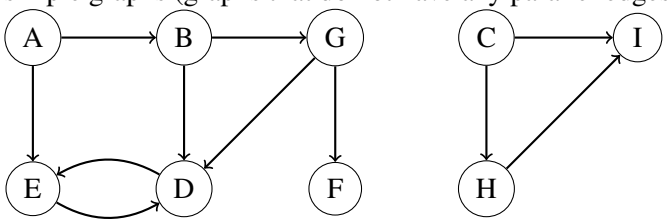
CMPSC 465 Spring 2024 — Data Structures & Algorithms — Mehrdad Mahdavi and David Koslicki

Worksheet 5

Wednesday, February 14, 2024

1. **Graph Basics.** For parts (a) through (c), refer to the figure below. For parts (d) and (e), consider only simple graphs (graphs that do not have any parallel edges or self-loops).



Run DFS at node A, trying to visit nodes alphabetically (e.g. given a choice between nodes D and F, visit D first).

(a) List the nodes in the order you visit them (so each node should appear in the ordering exactly once).

(b) List each node with its pre- and post-number. The numbering starts from 1 and ends at 18.

(c) Label each edge as **T**ree, **B**ack, **F**orward or **C**ross.

(d) Let $|E|$ be the number of edges in a simple graph and $|V|$ be the number of vertices. Show that $|E| = O(|V|^2)$.

(e) For each vertex v_i in an undirected graph, let d_i be the *degree*- the number of edges incident to it. Show that $\sum d_i$ must be even.

Solution

(a) Ordering: ABDEGFCHI

(b) The following table lists each node.

nodes	pre-visit	post-visit
A	1	12
B	2	11
D	3	6
E	4	5
G	7	10
F	8	9
C	13	18
H	14	17
I	15	16

(c) Tree: AB, BD, DE, BG, GF, CH, HI Back: ED Forward: AE, CI Cross: GD

(d) The maximum number of edges a graph can have is when every vertex is connected to every other vertex. This describes the 'complete' graph and has $\binom{|V|}{2} = O(|V|^2)$ edges.

(e) Each edge in the graph belongs to precisely 2 vertices. Thus, the total number of edges is $\frac{\sum d_i}{2}$. Since there are an integer number of edges, $\sum d_i$ must be even.

2. Pouring Water. We have three containers whose sizes are 10 pints, 7 pints, and 4 pints, respectively. The 7-pint and 4-pint containers start out full of water, but the 10-pint container is initially empty. We are allowed one type of operation: pouring the contents of one container into another, stopping only when the source container is empty or the destination container is full. We want to know if there is a sequence of pourings that leaves exactly 2 pints in the 7- or 4-pint container.

1. Model this as a graph problem: give a precise definition of the graph involved and state the specific question about this graph that needs to be answered.
2. What algorithm should be applied to solve the problem?
3. Find the answer by applying the algorithm.

Solution:

(a) Let $G = (V,E)$ be our (directed) graph. We will model the set of nodes as triples of numbers (a_0,a_1,a_2) where the following relationships hold: Let $S_0 = 10, S_1 = 7, S_2 = 4$ be the sizes of the corresponding containers. a_i will correspond to the actual contents of the i-th container. It must hold $0 \leq a_i \leq S_i$ for $i = 0,1,2$ and at any given node $a_0 + a_1 + a_2 = 11$ (the total amount of water we started from). An edge between two nodes (a_0,a_1,a_2) and (b_0,b_1,b_2) exists if both the following are satisfied:

- the two nodes differ in exactly two coordinates (and the third one is the same in both).
- if i, j are the coordinates they differ in, then either $a_i = 0$ or $a_j = 0$ or $a_i = S_i$ or $a_j = S_j$.

The question that needs to be answered is whether there exists a path between the nodes $(0,7,4)$ and $(*,2,*)$ or $(*,*,2)$ where $*$ stands for any (allowed) value of the corresponding coordinate.

(b) We can apply DFS on this graph, starting from node $(0,7,4)$ with an extra line of code that halts and answers 'YES' if one of the desired nodes is reached and 'NO' if all the connected component of the starting node is exhausted and no desired vertex is reached.

(c) DFS can visit the graph this way: $(0,7,4) \to (7,0,4) \to (7,4,0) \to (10,1,0) \to (6,1,4) \to (6,5,0) \to (2,5,4) \to (2,7,2)$. So, we have reached our desired vertex, and the algorithm will answer "Yes".

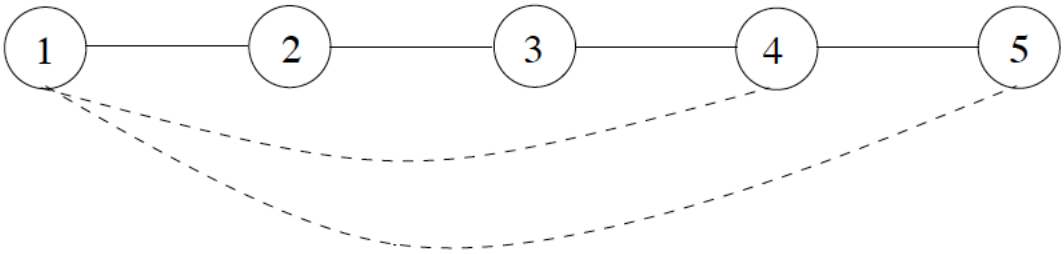
3. Shortest Cycle. Here's a proposal for how to find the length of the shortest cycle in an undirected graph with unit edge lengths.
When a back edge, say (*v*,*w*), is encountered during a depth-first search, it forms a cycle with the tree edges from *w* to *v*. The length of the cycle is $level[v] - level[w] + 1$, where the level of a vertex is its distance in the DFS tree from the root vertex. This suggests the following algorithm:

1. Do a depth-first search, keeping track of the level of each vertex.

2. Each time a back edge is encountered, compute the cycle length and save it if it is smaller than the shortest one previously seen.

Show that this strategy does not always work by providing a counterexample as well as a brief (one- or two-sentence) explanation.

Solution: The graph in the below figure is a counterexample: vertices are labeled with their level in the DFS tree and back edges are dashed. The shortest cycle consists of vertices $1-4-5$, but the cycle found by the algorithm is $1-2-3-4$. In general, the strategy will fail if the shortest cycle contains more than one back edge.



4. Particular Edge Cycle. Design a linear-time algorithm which, given an undirected graph G and a particular edge e in it, determines whether G has a cycle containing e.

Solution: Let u and v be the vertices on either end of edge e. A cycle containing e implies that u can reach v not only through e but also through some other path. To check this, we construct a graph G' that is identical to G, but with edge e removed. Using this new graph, the problem is reduced to "can vertex u reach vertex v in G'?" Since both graphs are undirected, this is equivalent to "are u and v in the same connected component of G'?".
We can answer this question by running DFS Explore on G', starting from u. If at any point we reach v, we can return true - the original graph G does have a cycle containing edge e. If we check every node reachable from u without finding v, we can return false - the original graph G does not have a cycle containing edge e. Because it runs DFS Explore and at worst explores the entire graph before returning, the running time of this algorithm is $O(|V|+|E|)$.

5. Post Number Ancestors. Assume (u,v) is an edge in an undirected graph. Either prove the following statement or provide a counterexample: If during DFS $\texttt{post}(v) < \texttt{post}(u)$, then u is an ancestor of v in the DFS tree.

Solution:
There are two possible cases:
$\texttt{pre}(u) < \texttt{pre}(v) < \texttt{post}(v) < \texttt{post}(u)$ or $\texttt{pre}(v) < \texttt{post}(v) < \texttt{pre}(u) < \texttt{post}(u)$.
In the first case, u is an ancestor of v by definition.
In the second case, v was visited, all of it's neighbors were visited, and DFS returned to v, all before u was visited. However, since there is an edge between v and u and we look at all neighbors of v, this cannot happen. So, the given statement is true.