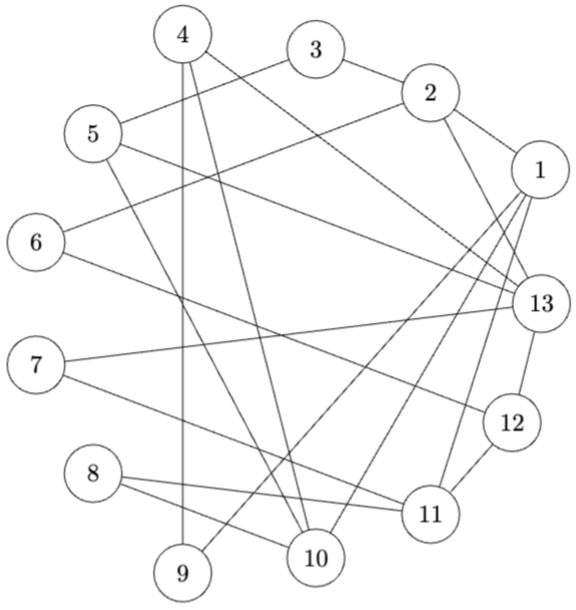
Intro Algorithms EN.601.433, Spring 2024, Hw 2

ID:D5F9AC

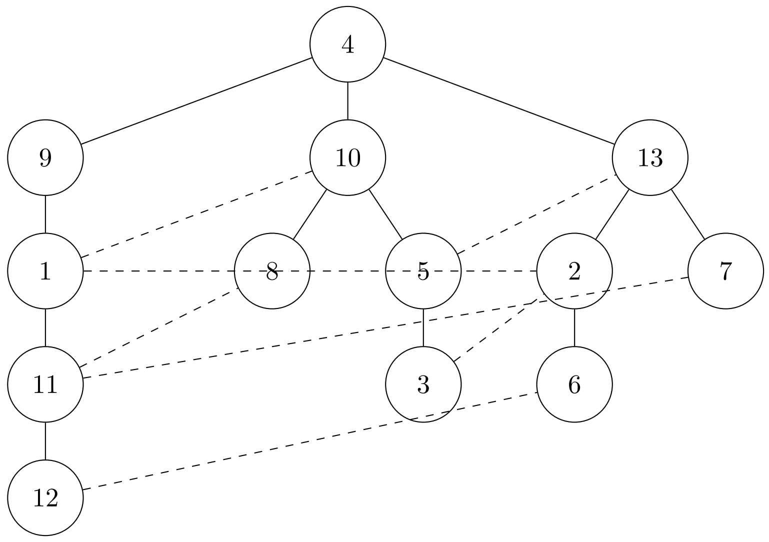
February 22, 2024

1 Determine Bipartite using BFS (3 points)

Create a Breadth-First-Search (BFS) tree for the following graphs and deduce whether or not it is a bipartite graph.



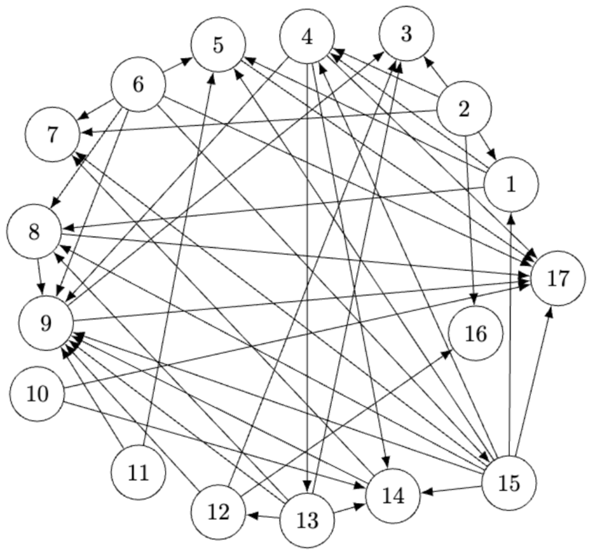
Choosing 4 as my starting node to run BFS, the graph I get is:



Using the definition of a bipartite graph, since nodes 2 and 1 are connected and on the same level and G contains an odd length cycle, the graph is not bipartite

2 Topological Ordering (2 Points)

Find a topological ordering of the Directed Acyclic Graph (DAG) shown below. Use numerical order with the smallest numbers first to break ties.



Using the algorithm provided in lecture to compute a topological ordering of G, I got:
2,6,10,11,15,1,4,5,13,8,12,14,7,9,3,16,17

3 Detect a Cycle (5 Points)

Give an algorithm to detect whether a given undirected graph contains a cycle. If the graph contains a cycle, then your algorithm should output a cycle. (It should not output all the cycles in the graph, just one cycle.) The running time of your algorithm should be O(m + n) for a graph with n nodes and m edges.

Algorithm 1 Detecting a cycle in a graph

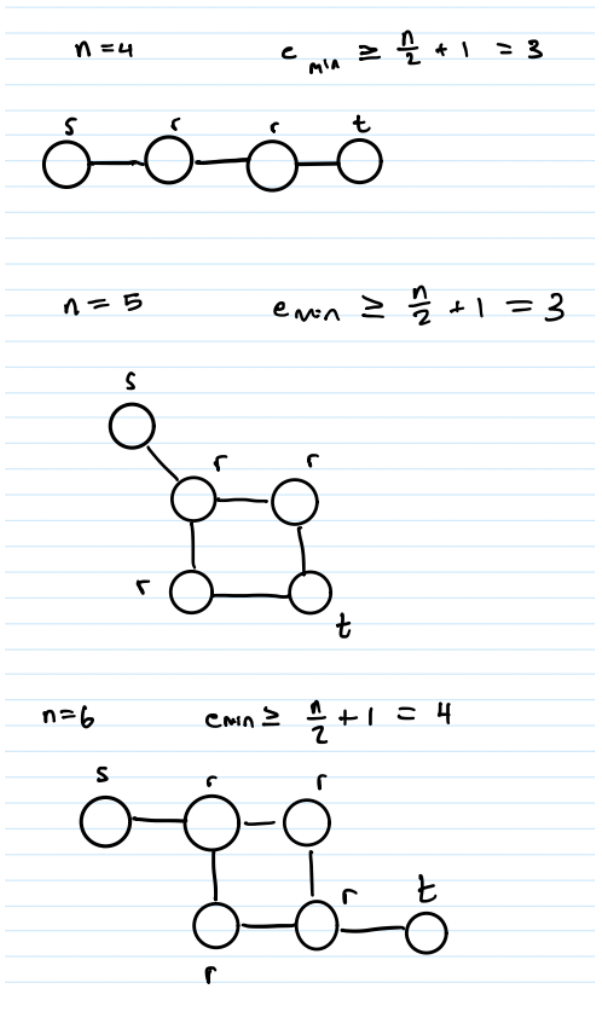
```
Input: Graph $G$
Output: Cycle in $G$, if exists
Choose a starting node $s$
Run modified BFS on $s$
    pop curr node and store in a list to denote current searched path
    search curr node outgoing nodes
        if outgoing node is not a parent (in a level right before current nodes level) of the current node and if outgoing node is explored
            we have found a cycle
            save outgoing node and add to path list
            end BFS
Let $v$ and $w$ denote nodes at the edge where the cycle is determined
Mark all ancestor nodes of $v$ and $w$ as explored 2 until we reach the starting node by running a search on the path list from v or w to the first node in the path(each search will be a runtime of O(n) but the overall runtime will still be O(m+n) from BFS)
During our search, the first instance of an ancestor already marked with explored 2, is the first node of our cycle
Call this first cycle node the least common ancestor and return cycle by returning all nodes between w and least common ancestor node (ie all nodes after least common ancestor and marked explored 2)
```

4 All paths go through r_{ome} (8 points)

There's a natural intuition that two nodes that are far apart in a communication network—separated by many hops have a more tenuous connection than two nodes that are close together. There are a number of algorithmic results that are based to some extent on different ways of making this notion precise. Here's one that involves the susceptibility of paths to the deletion of nodes. Suppose that an n-node undirected graph G = (V, E) contains two nodes s and t such that the distance between s and t is strictly greater than n 2 . Either prove or disprove there exists a node r which is not equal to s or t, such that deleting r from G will destroy all paths between s and t. Your solution should include both:

Draw a n-node graph (for n $\geq$ 4) where the shortest path between s and t is strictly greater than n/2 . You should use this part of the problem to develop an intuitive understanding of the graph's properties. You are not required to type this part of your solution, you can submit a photo of a hand-drawn graph. To get you started, here is an example 3-node graph containing nodes s, t, and r as described above.

Construct a proof or counterexample that exist a node r for all graphs G where distance between s and t is strictly greater than $\frac{n}{2}$ and that removing r will destroy all paths between s and t.



$n=4$
$c_{min} \geq \frac{n}{2}+1 = 3$
s
r
r
t
$n=5$
$e_{min} \geq \frac{n}{2}+1 = 3$
s
r
r
r
t
$n=6$
$c_{min} \geq \frac{n}{2}+1 = 4$
s
r
r
r
t
r

Proof by Contradiction: There exists a node r for all graphs G where distance between s and t is strictly greater than $\frac{n}{2}$ and that removing r will not destroy all paths between s and t.
From the graph drawn above and in general, in order to have a path to t where we don't have an r such that when deleted, destroys the path from s to t, each node in the path from s to t including s and t needs to be connected twice(have at least 2 edges)
From this, there need to be a minimum of 2 paths from s to t so that when an r in between s and t is

deleted, there still is a ”backup” path connected from s to t.

Lets call the two paths from s to t p_{backup} and p_{main}. Our overall path is $p = p_{backup} + p_{main}$. Since $p_{backup} = p_{main}$, $n_{backup} = n_{main}$ where n is the number of nodes in either p_{backup} or p_{main} including s and t, and $e_{backup} = e_{main}$ where e is the number of edges in either p_{backup} or p_{main} including edges to s and t. Thus $p = 2p_{main} = 2p_{backup}$ and $e_{path} = 2e_{backup} = 2e_{main}$.

Lets say our graph G has n total nodes where $n = n_p + 2$, and n_p is strictly the nodes within our path p not counting nodes s and t and +2 are the nodes for s and t.

Each path p_i within p must strictly have edges $e_i > \frac{n}{2}$, or in other words have total number of edges $e_i \geq 1 + \frac{n}{2}$, where e_i is the number of edges within path p_i.

The min number of edges for a path p_i is $e_i = 1 + \frac{n}{2}$, which counts edges to nodes s and t.

Since this is the min number of edges in p_i, the min number of nodes in p_i would be $n_i = e_i + 1 = 2 + \frac{n}{2}$, which also counts nodes for s and t. Thus for 2 paths or for $p = 2pi$, the number of nodes would be $n_p = 2n_i - 2 = 4 + n - 2 = 2 + n$, and -2 comes from removing the double count of nodes s and t when adding nodes p_i and p_i together since they both include nodes s and t.

Since $n_p = 2 + n$ or $n_p = n$ when not counting for nodes s and t. If we plug this for n_p we get $n = n + 2$ which is not possible since we only have n available nodes and the number of nodes required for creating a graph following the proposed property is n + 2 nodes, therefore this is a contradiction.

5 Some Short Paths in a Graph (8 points)

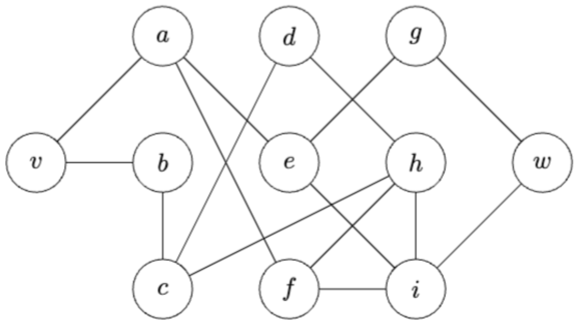
A number of art museums around the country have been featuring work by an artist named Mark Lombardi (1951–2000), consisting of a set of intricately rendered graphs. Building on a great deal of research, these graphs encode the relationships among people involved in major political scandals over the past several decades: the nodes correspond to participants, and each edge indicates some type of relationship between a pair of participants. And so, if you peer closely enough at the drawings, you can trace out ominous-looking paths from a high-ranking U.S. government official, to a former business partner, to a bank in Switzerland, to a shadowy arms dealer.

Such pictures form striking examples of social networks, which, as we discussed in book chapter 3.1, have nodes representing people and organizations, and edges representing relationships of various kinds. And the short paths that abound in these networks have attracted considerable attention recently, as people ponder what they mean. In the case of Mark Lombardi’s graphs, they hint at the short set of steps that can carry you from the reputable to the disreputable.

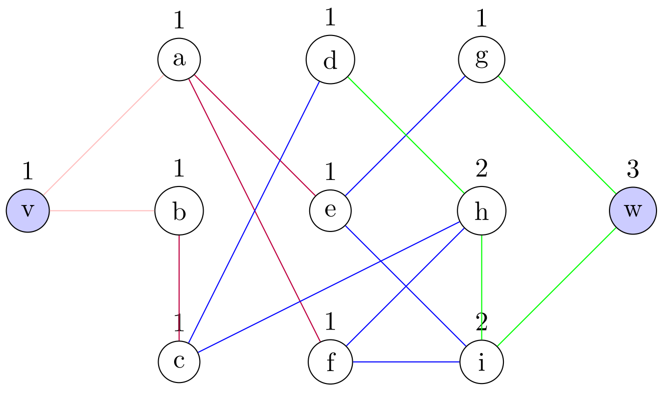
Of course, a single, spurious short path between nodes v and w in such a network may be more coincidental than anything else; a large number of short paths between v and w can be much more convincing. So in addition to the problem of computing a single shortest v-w path in a graph G, social networks researchers have looked at the problem of determining the number of shortest v-w paths.

This turns out to be a problem that can be solved efficiently. Suppose we are given an undirected graph G = (V, E), and we identify two nodes v and w in G. Give an algorithm that computes the number of shortest v-w paths in G. (The algorithm should not list all the paths; just the number suffices.) The running time of your algorithm should be O(m + n) for a graph with n nodes and m edges.

Demonstrate your algorithm working on the graph given below.



Key: pink = layer 1, red = layer 2, blue = layer 3, green = layer 4 Note: V is our source and w is our target node



Algorithm 2 Finding total number of shortest paths in a graph

```
Choose a starting node $v$
Initialize weight of starting node to 1
Run modified recursive breadth-first search on $v$
    For this version of the search we want to recursively find the shortest path to each child node for each parent node and add the parent weight to each child found.
    this value represents how many ways there are to get to the child node.
Stop the algorithm once we assign all child nodes each parent nodes weight
return the count for the number of shortest paths by returning the weight for our target node
```

The runtime for this algorithm will be O(m+n) since it is a recursive BFS. Assigning weights to outgoing nodes will be O(1) since weight is stored as a field within the node. Thus the overall runtime is unaffected and remains O(m+n)

6 Wireless Network Connectivity (5 points)

Some friends of yours work on wireless networks, and they're currently studying the properties of a network of n mobile devices. As the devices move around (actually, as their human owners move around), they define a graph at any point in time as follows: there is a node representing each of the n devices, and there is an edge between device i and device j if the physical locations of i and j are no more than 500 meters apart. (If so, we say that i and j are "in range" of each other.)

They'd like it to be the case that the network of devices is connected at all times, and so they've constrained the motion of the devices to satisfy the following property: at all times, each device i is within 500 meters of at least n/2 of the other devices. (We'll assume n is an even number.) What they'd like to know is: Does this property by itself guarantee that the network will remain connected?

Here's a concrete way to formulate the question as a claim about graphs.

Claim: Let G be a graph on n nodes, where n is an even number. If every node of G has degree at least n/2, then G is connected.

Decide whether you think the claim is true or false, and give a proof of either the claim or its negation.

Proof by contradiction: Let G be a graph on n nodes, where n is an even number. If every node of G has degree at least n/2, then G is not connected.

Without loss of generality assume G is a graph with n nodes, where n is even, and G consists of 2 sub graphs .
Lets say our subgraphs are graphs with the least number of nodes that follow this property and call them g_1 and g_2.
Let g_1 be a single node connected with at least n/2 other nodes, thus there are a minimum of n/2 + 1 nodes.
Since G is a graph of n nodes, g_2 has n - (n/2 + 1) nodes or n/2 - 1 nodes.
This cant be possible given that the number of nodes comprising a node and its connections must be n/2 and itself.
Thus this is a contradiction.