Stefanie Shidoosh

<u>CS 180 Homework #2</u>

1a. We have seen that an undirected connected graph $G = (V, E)$ such that all the vertices have even degrees has an Eulerian cycle. Give an $O(|E|)$ time algorithm to find it.

Pick an arbitrary vertex, v_0
Starting v_0, find a cycle, c_0, that initiates from v_0
    Mark the edges traveled (and thus the vertices visited)
If there is a vertex, v_x, in c_0 that has an unmarked edge incident on it:
    Find a cycle, c_1 that initiates from initiates from v_x and uses only the unmarked edges thus far in E. Mark the traveled edges.
    Unite c_0 and c_1 to create a cycle that initiates from v_0. This cycle is now c_0.
    Repeat until all edges are marked.

This algorithm takes $O(|E|)$ as it is necessary to visit every edge to conclude that there exists an Eulerian cycle, since by definition such a cycle must visit every edge in G exactly once.

Stefanie Shidoosh

1b. A directed graph is strongly connected if every vertex is reachable from every other vertex. Assume that for every vertex in a directed graph $G = (V,E)$ its in-degree equals its out degree, and G is strongly connected. Prove that G has an Eulerian cycle and give an $O(E)$ time algorithm to find it.

If for every vertex in a directed graph $G = (V,E)$ its in-degree equals its out degree, then for any vertex v there must be a cycle that contains v, as when entering another vertex, there is an unvisited edge leaving that vertex because its in-degree equals its out degree. The only time that this may not true is for v, when the cycle initiates from v, as the cycle used an outgoing edge.
Therefore, to prove that such a graph has an Eulerian cycle:

Pick an arbitrary vertex, v_0, from G_0

Starting v_0, find a cycle, c_0, that initiates from v_0
Remove that cycle, s.t. an in-edge and out-edge of the incident vertices are removed, resulting in G_1
Repeat for G_1, which will be G_0
Stop when all edges are visited

Since it is strongly connected and the in-degree is equal to its out-degree, for the outgoing edge taken there is an incoming edge that allows for the completion of a cycle. Thus, we find cycles that share at least one vertex v and therefore we can combine them to make one whole cycle that includes all edges, hence an Eulerian cycle.

The algorithm for a directed graph is the similar to an undirected graph when given the assumption that for every vertex in a strongly connected directed graph $G = (V, E)$ its in-degree equals its out degree, as this provides that for every edge leaving a vertex v there must be an edge entering v, in a similar way as an even degree offers the same ability. Thus, this algorithm is a modified version of the aforementioned algorithm:

Pick an arbitrary vertex, v_0
Starting at v_0, find a cycle, c_0, with respect to only traveling outgoing edges relative to the current node, until point of return to v_0
Mark the edges traveled (and thus the vertices visited)
If there is a vertex, v_x, in $V(c_0)$ that has an unmarked outgoing edge incident on it (there is a cycle back to it because the in-degree is equal to its out-degree):

Find a cycle, c_1 that initiates from initiates from v_x and uses only the unmarked edges thus far in $E(G)$. Mark the traveled edges.
Unite c_0 and c_1 to create a cycle that initiates from v_0. This cycle is now c_0.
Repeat until all edges are marked.

This algorithm takes $O(E)$ as it is necessary to visit every edge to conclude that there exists an Eulerian cycle, since by definition such a cycle must visit every edge in G exactly once.

Stefanie Shidoosh

2. A celebrity among n persons is someone who is known by everyone but does not know anyone. Equivalently, given a directed graph $G = (V,E)$ with n vertices, a directed edge from v_i to v_j. represents person i knows person j, a celebrity vertex is the vertex with no outgoing edge and $n - 1$ incoming edges. In the class, we have seen an $O(n)$ recursive algorithm that finds whether celebrity exists or not and it does returns it. The graph G is represented by an $n \times n$ adjacency matrix M, which is a (0,1)-matrix such that $M[i, j] = 1$ if and only if there is a directed edge from v_i to v_j. Give an iterative $O(n)$ time algorithm to find the celebrity vertex in G, or output none if no one is.

(1) Choose 2 vertices, v_0 and v_1, arbitrarily

If there exists an edge from v_i directed to v_j, s.t. $M[i, j] = 1$, eliminate v_i from set of potential celebrity vertices, V_p

Otherwise, eliminate v_j from set of potential celebrity vertices, V_p

Repeat (1) for all v in $V(G)$. Once completed, exactly one v will remain in V_p, and thus it is a potential celebrity, v_p.

(2) Check that M[u, p] = 1, where u is all other n-1 vertices, n is total number of vertices in V(G).

(3) Check that M[p, u] = 0, where u is all other n-1 vertices, n is total number of vertices in V(G).

(4) If (2) and (3) are both true, then v_p is the celebrity vertex. Else, there is no celebrity vertex in V(G).

(1) takes n-1 steps, (2) takes n-1 steps, and (3) takes (n-1) steps. Thus, in total, this algorithm takes 3(n-1) steps, and therefore the time complexity is $O(N)$.

Stefanie Shidoosh

3. Given an undirected tree T, the diameter of a tree is the number of edges in the longest path in the tree. Design an algorithm that find the diameter of the tree in $O(N)$ time where n is number of the nodes in the tree.

> Start at root node of T
> Recursively calculate the maximum height of the subtree to the left of the node and the maximum height of the subtree to the right of the node
> To calculate:
>> height of left subtree + height of right subtree + 1,
>> relative to the current node
>
> Stop when terminal nodes are reached
> If the height of the right and left subtrees is equivalent, the diameter is that height
> Otherwise, the diameter is height that is greater with respect to the heights of the left and right subtrees.

This algorithm is $O(N)$, as each node must be visited such that the height of each node's subtrees is calculated.

Stefanie Shidoosh

4. Let $K_n = (V, E)$ be a complete undirected graph with n vertices (namely, every two vertices are connected), and let n be an even number. A spanning tree of G is a *connected* subgraph of G that contains all vertices in G and no cycles. Design a recursive algorithm that given the graph K_n partitions the set of edges E into n/2 distinct subsets such that for every subset E_i the subgraph $G_i = (V, E_i)$ is a spanning tree of K_n.

Recursively remove 2 arbitrary nodes and their unmarked respective edges in K_n.
  This results in (n-2)/2 spanning trees for K_{n-2}
  To extend those trees to be spanning trees for K_n:
    Add one distinct edge back to each spanning tree per removed node
    Connect the nodes that were removed to the trees using these edges
  Mark the edges that were added back, with respect to each subset E_i … E_{n-2}, at each recursive step
  Backtrack to the 2 nodes until all edges have been marked
  This will then include all v in $V(K_n)$, and because it is a complete graph, there are still n-2 edges available to choose from to reach all other vertices upon the next recursive step, until all edges are marked.

The time complexity for this algorithm is $O(V + E)$ as all vertices must be visited and all edges must be delegated to n/2 distinct subsets.