Nicolas Carchio
Adam Romano
Professor Ames
CS344

1. It is possible to determine whether G has a cycle containing a particular edge E by using DFS and either rebuilding/modifying the graph before DFS execution. For the sake of the algorithm, suppose the given edge E connects to vertices X and Y (X,Y). The algorithm will go as follows:
    a. **Remove edge E from the graph**. (If the circumstances do not favor modifying the graph, you can rebuild the same graph without E. This will increase the running time of the algorithm, however, it will still yield a linear time algorithm)
    b. **Perform a depth-first search starting from either X or Y**.
    c. **If either X or Y is found** (depending on which vertex DFS is started on) **during the execution of depth-first search, a cycle exists in G containing the edge E**
    d. **If either X or Y is not found, no cycle exists within G containing the edge E**.

    In terms of running time of this algorithm, there are two main aspects to analyze: removing the edge E from the graph and performing a standard depth-first search on G. The running time to remove an edge from G will depend on the representation of G. If G is represented as an adjacency matrix, then it will take O(1) time in order to remove the edge E from G since a 2-D array provides constant time access. If G is represented as an adjacency list, then it will take O(|V|) time in order to remove E from G. Running standard DFS will take O(|V| + |E|) time. Therefore, the total running time is:
    - Adjacency matrix: O(1) + O(|V| + |E|) → O(|V| + |E|)
    - Adjacency list: O(|V|) + O(|V| + |E|) → O(|V| + |E|)

    In both representations, the total running time of the algorithm is simply the running time to run a standard DFS, which is linear in terms of the graph G.

2. In order to calculate the minimum number of semesters required to complete the curriculum, we must perform a DFS and calculate vertices as the recursive calls back up to the starting vertex. Initially, all vertices are labeled with a 0, implying that they take 0 semesters to complete. Once the last vertex in the graph is reached and DFS starts backtracking, we must compare the maximum of current node's value + 1 to the next node in the graph. If the current node's value + 1 is greater than the next node in the graph, then the current node's value increments by 1. This implies that after the current vertex, there is only one semester left before the curriculum completes. By the end, the starting vertex's value will be the minimum number of semesters required in order to complete the curriculum. Since DFS Is only performed once on the graph, the running time of the algorithm is simply going to be the running time for DFS which is O(|V|+|E|).

Nicolas Carchio
Adam Romano
Professor Ames
CS344

3. Using BFS, iterate through the graph. For every vertex, add all of its neighbors to a list and map that vertex to its list of neighbors in a hashmap/dictionary. After BFS completes, every vertex will be mapped within the hashmap/dictionary to a list of its neighbors. If a simple cycle of length 4 exists within the graph, then at least two distinct vertices will share the same pair of neighbors. In order to determine if two distinct vertices share the same pair of neighbors, we must compare. The time for BFS to run is $O(|V|+|E|)$. The time to compare will be at most $O(|V|)^3$ since we have to iterate through a list of vertices for every distinct pair of vertices within the graph. Therefore, since $O(|V|)^3$ dominate $O(|V|+|E|)$, even in the case where $|E| = |V|^2$, the total running time will be $O(|V|)^3$.

4. Let t be the total number of edges in the graph. Since there are k connected components, if we add an edge, that will reduce the number of connected components by 1, meaning there will be k-1 connected components. If we add k-1 edges, the entire graph will be connected. As a connected graph, we know that it has a spanning tree which will have at least t-1 edges. Therefore:
    - t + (k-1) >= t-1
    - t + k >= t
    - t >= t-k

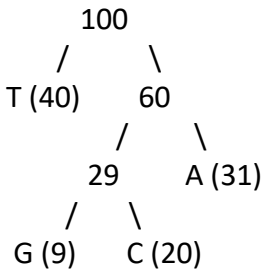
5. Kruskal's algorithm is used to find a minimum-spanning tree, meaning subtree of the graph with every vertex that is connected at a minimum cost. In order to find a maximum-spanning tree, we can invert the edge weights. If an edge weight is 0, leave it alone. Otherwise, multiply the edge weight by -1 and then run Kruskals algorithm. Since the edge weights will be inverted, the algorithm will create a spanning tree with minimum weights, which will really be the maximum weighted edges multiplied by -1. At the very end, multiply the edge weights of the maximum spanning tree by -1 to get their original values and you will have the maximum spanning tree of the graph.

6. A long string consists of the four characters A, C, G, T; they appear with frequency 31%, 20%, 9%, and 40%, respectively. What is the Huffman encoding of these four characters?

    (answer below)

Nicolas Carchio
Adam Romano
Professor Ames
CS344

The Huffman encoding will be:



Now:

Letter	Frequency	Huffman Encoding
T	40	0
A	31	11
C	20	101
G	9	100