CS4102 - ALGORITHMS - SPRING 2022

DUE FRIDAY, MARCH 18, 2022 AT **11:30 pm** VIA GRADESCOPE

B1 BASIC(HZ9XS)

Collaboration Policy: You are encouraged to collaborate with up to 3 other students, but all work submitted must be your own *independently* written solution. List the computing ids of all of your collaborators in the `collabs` command at the top of the tex file. Do not share written notes, documents (including Google docs, Overleaf docs, discussion notes, PDFs), or code. Do not seek published or online solutions for any assignments. If you use any published or online resources (which may not include solutions) when completing this assignment, be sure to cite by naming the book etc. or listing a website's URL. Do not submit a solution that you are unable to explain orally to a member of the course staff. Any solutions that share similar text/code will be considered in breach of this policy. Please refer to the syllabus for a complete description of the collaboration policy.

Collaborators: hl8zd, xs7tng

Sources: Cormen, et al, Introduction to Algorithms. *(add others here)*

PROBLEM 1 *Short Questions on BFS*

A. What is the maximum number of vertices that can be on the queue at one time in a BFS search? Briefly explain the situation that would cause this number to be on the queue.

Solution: The maximum number of vertices that can be on the queue at one time in a BFS search is the number of vertices minus one. When all other vertices are the adjacent vertex of the starter vertex, the while loop will enqueue all other vertices to the queue, which makes the queue containing the maximum number of vertices at one time.

B. If you draw the BFS tree for an undirected graph G, some of the edges in G will not be part of the tree. Explain why it's not possible for one of these non-tree edges to connect two vertices that have a difference of depth that's greater than 1 in the tree.

Solution: The basic property of BFS tree is that it will add a vertex to the tree as "early" as possible, as soon as it sees an edge connection. So, if you have an edge that connects two vertices with a depth difference that's greater than 1, it will certainly be used in the BFS tree because it is a shorter path to the node. In this way, non-tree edges can not connect two vertices that have a difference of depth that's greater than 1 in the tree.

PROBLEM 2 *True or False. (You don't have to explain this in your submission, but you should understand the reason behind your answer.)*

A. If you use BFS to detect a cycle in an undirected graph, an edge that connects to a vertex that's currently on the queue or has been removed from the queue indicates a cycle as long as that vertex is not the parent of the current node.

Solution: False

B. If you use DFS-visit on a *directed* graph with $V > 1$ starting at vertex v_1, you will always visit the same number of vertices that you would if you started at another node v_2.

Solution: False

C. If you use DFS-visit on a connected *undirected* graph with $V > 1$ starting at vertex v_1, you will always visit the same number of vertices that you would if you started at another node v_2. (If it were not connected, would your answer change?)

Solution: True. If it is not connected, the statement will be false.

PROBLEM 3 *Finding Cycles Using DFS*

In a few sentences, explain how to recognize a directed graph has a cycle in the DFS-visit algorithm's code we saw in class. How does this need to be modified if the graph is undirected?

Solution: Directed: if an edge is encountered from current vertex to a GRAY vertex, then this edge is a back edge and hence there is a cycle. If the graph is undirected: for every visited vertex v, when we have found any adjacent vertex u, such that u is already visited, and u is not the parent of vertex v. Then one cycle is detected.

PROBLEM 4 *Finding a path between two vertices*

Describe the modifications you would make to DFS-visit() given in class to allow it to find a path from a start node s to a target node t. The function should stop the search when it finds the target and return the path from s to t.

Solution: We need to add something in the for loop of DST-visit(): if the current node is unseen and it is the target node, we will use a while loop to print its parent and its parent's parent until s which is the start node, and end searching after we return the path.

PROBLEM 5 *Labeling Nodes in a Connected Components*

In a few sentences, explain how you'd modify the DFS functions taught in class to assign a value $v.cc$ to each vertex v in an undirected graph G so that all vertices in the same connected component have the same cc values. Also, count the number of connected components in G.
In addition to your explanation, give the order-class of the time-complexity of your algorithm.

Solution: we need to add a counter (initialized to 0) in the DFS_sweep method. In the for loop of DFS_sweep method, for each vertex in the graph, if it is unseen, counter increases by one. Whenever you visit a node using DFS-VISIT method, assign the current value of the counter to $v.cc$, so that all vertices in the same component have the same cc value. The counter counts the number of connected components in graph. The time-complexity of my algorithm is $\Theta(V + E)$.

PROBLEM 6 *BFS and DFS Trees*

Consider the BFS tree T_B and the DFS tree T_D for the same graph G and same starting vertex s. In a few sentences, clearly explain why for every vertex v in G, the depth of v in the BFS tree cannot be greater than its depth in the DFS tree. That is:

$$\forall\, v \in G.V, depth(T_B, v) \leq depth(T_D, v)$$

(Here the depth of a node is the number of edges from the node to the tree's root node. Also, you can use properties of BFS and DFS that you've been taught in class. We're not asking you to prove those properties.)

Solution: One important property of BFS is that the paths in this BFS tree represent the shortest paths from the root to each node in G, which means that for every vertex v in G, the depth of v in the BFS tree is the smallest, so it cannot be greater than its depth in the DFS tree.

PROBLEM 7 *Gradescope Submission*

Submit a version of this `.tex` file to Gradescope with your solutions added, along with the compiled PDF. You should only submit your `.pdf` and `.tex` files.