CS 170 Homework 3

Due **Saturday 2/15/2025, at 10:00 pm (grace period until 11:59pm)**

1 Study Group (1 point)

List the names and SIDs of the members in your study group. If you have no collaborators, you must explicitly write "none".

2 Depth First Search (10 points)

Depth first search is a useful and often efficient way to organize computations on a graph.

Let G be an undirected connected tree, and let $wt : E \to \mathbb{R}^+$ be positive weights on its edges. We show a template for DFS-based computations below.

```
1: Input: Undirected connected tree G = (V, E) and positive weights wt(u, v) for each
   edge (u, v) ∈ E
2:
3: Initialization:
4: visited[v] ← False for all vertices v.
5: A[v] ← 0 and B[v] ← 0 for all vertices v.
6: t ← 1.
7:
8: function Explore(Vertex u)
9:     visited[u] ← True
10:    for each edge (u, v) in E do
11:        if NOT visited[v] then
12:            PreVisit(u, v)
13:            Explore(v)
14:            PostVisit(u, v)
```

DFS can be used for different purposes by defining the procedures PreVisit and PostVisit appropriately.

(a) (4 points) In each of the following cases, PreVisit and PostVisit have been defined for you. After execution, the array $A[v]$ will hold a value for each vertex v. Describe in words what $A[v]$ represents.

(i)
```
1: procedure PreVisit(u, v)
2:     return
3:
4: procedure PostVisit(u, v)
5:     A[u] ← max(A[u], A[v] + 1)
```


(ii)
```
1: procedure PreVisit(u, v)
2:     B[u] ← B[u] + 1
3:
4: procedure PostVisit(u, v)
5:     A[u] ← max(A[u], B[v] + 1)
```

(b) (6 points) In each of the following cases, write down pseudocode for PreVisit and PostVisit routines to perform the computation needed.

(i) For each vertex v, compute the maximum weight of an edge along the path from root r to vertex v and store it in array $A[v]$.

(ii) For each vertex v, compute the maximum weight of any edge in the subtree rooted at vertex v and store it in array $A[v]$.

(iii) For each vertex v, compute the maximum pre-order number of any of its children and store it in array $A[v]$. If v has no children, then $A[v]$ should be 0.

3 Biconnected Components (11 points)

Consider any undirected connected graph $G = (V, E)$. We say that an edge $(u, v) \in E$ is *critical* if removing it disconnects the graph. In other words, the graph $(V, E \setminus (u, v))$ is no longer connected.
Similarly, we call a vertex $v \in V$ critical if removing v (and all its incident edges) leaves the graph disconnected.

(a) (2 points) Suppose that $|V| \geq 2$. Can you always find a vertex $v \in V$ that is **not** critical? What about an edge that is not critical?

(b) (6 points) Give a linear time algorithm to find all the critical edges of G.

(c) (3 points) Modify your algorithm above to find all the critical vertices of G.

4 Topological Sort Proofs (14 points)

(a) (6 points) A directed acyclic graph G is *semiconnected* if for any two vertices A and B, there is a path from A to B or a path from B to A. Show that G is semiconnected if and only if there is a directed path that visits all of the vertices of G. Make sure to prove both sides of the "if and only if" condition.

Hint: Is there a specific arrangement of the vertices that can help us solve this problem?

(b) (4 points) Show that a DAG has a unique topological ordering if and only if it has a directed path that visits all of its vertices.

Remark: This means that a semiconnected DAG always has a unique topological ordering.

(c) (4 points) This subpart is unrelated to the notion of semiconnectedness. Consider what would happen if we ran the topological sorting algorithm from class on a directed graph that had cycles.

Prove or disprove the following: The algorithm would output an ordering with the least number of edges pointing backwards.

5 Distant Descendants (13 points)

You are given a tree $T = (V, E)$ with a designated root node r and a positive integer K. For each vertex v, let $d[v]$ be the number of descendants of v that are a distance of at least K from v. In this problem, we will find an $O(|V|)$ algorithm to output $d[v]$ for every v.

(a) (4 points) Write an $O(|V|)$ algorithm that computes the total size of the subtree (number of descendants plus 1 for the vertex itself) of each vertex v in an array $s[v]$. Give a brief justification that your algorithm is correct and runs in $O(|V|)$ time. Do not just cite an algorithm from class; reproduce anything you use in your solution.

(b) (5 points) Write an $O(|V|)$ algorithm that computes the K-th level ancestor of each vertex v (null if the depth of v is less than K) in an array $a[v]$. Give a brief justification that your algorithm is correct and runs in $O(|V|)$ time. Make sure your algorithm runs in $O(|V|)$ time and not $O(K|V|)$ time.

(c) (4 points) Write an $O(|V|)$ algorithm to compute $d[v]$ for each vertex v using $s[v]$ and $a[v]$. Give a brief justification that your algorithm is correct and runs in $O(|V|)$ time.

6 [Coding] DFS & Edge Classification (6 points)

For this week's homework, you'll implement implement DFS and use DFS to classify edges in a graph as forward/tree, backward, or cross edges. There are two ways that you can access the notebook and complete the problems:

1. **On Datahub**: click here and navigate to the `hw03` folder.
2. **On Local Machine**: `git clone` (or if you already cloned it, `git pull`) from the coding homework repo,

   https://github.com/Berkeley-CS170/cs170-sp25-coding

   and navigate to the `hw03` folder. Refer to the `README.md` for local setup instructions.

Notes:

- *Submission Instructions:* Please download your completed submission `.zip` file and submit it to the Gradescope assignment titled "Homework 3 Coding Portion".
- *Getting Help:* Conceptual questions are always welcome on Edstem and office hours; *note that support for debugging help during OH will be limited.* If you need debugging help first try asking on the public Edstem threads. To ensure others can help you, make sure to:
  1. Describe the steps you've taken to debug the issue prior to posting on Ed.
  2. Describe the specific error you're running into.
  3. Include a few small but nontrivial test cases, alongside both the output you expected to receive and your function's actual output.

  If staff tells you to make a private Ed post, make sure to include *all of the above items* plus your full function implementation. If you don't provide them, we will ask you to provide them.
- *Academic Honesty Guideline:* We realize that code for some of the algorithms we ask you to implement may be readily available online, but we strongly encourage you to not directly copy code from these sources. Instead, try to refer to the resources mentioned in the notebook and come up with code yourself. That being said, we **do acknowledge** that there may not be many different ways to code up particular algorithms and that your solution may be similar to other solutions available online.

