CS 170 Homework 3

Due **Friday 9/20/2024, at 10:00 pm (grace period until 11:59pm)**

1 Study Group

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

2 Hadamard matrices

The Hadamard matrices $H_0, H_1, H_2, \ldots$ are defined as follows:

- H_0 is the 1×1 matrix $[1]$
- For $k > 0, H_k$ is recursively defined as the $2^k \times 2^k$ matrix

$$H_k = \left[\begin{array}{c|c} H_{k-1} & H_{k-1} \\ \hline H_{k-1} & -H_{k-1} \end{array}\right]$$

For instance, the first three Hadamard matrices H_0, H_1, and H_2 are:

$$H_0 = \begin{bmatrix} 1 \end{bmatrix} \quad H_1 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad H_2 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}$$

(a) Suppose that

$$v = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$$

is a column vector of length $n = 2^k$. v_1 and v_2 are the top and bottom half of the vector, respectively. Therefore, they are each vectors of length $\frac{n}{2} = 2^{k-1}$. Write the matrix-vector product $H_k v$ in terms of H_{k-1}, v_1, and v_2 (note that H_{k-1} is a matrix of dimension $\frac{n}{2} \times \frac{n}{2}$, or $2^{k-1} \times 2^{k-1}$). Since H_k is a $n \times n$ matrix, and v is a vector of length n, the result will be a vector of length n.

(b) Use your result from the previous part to come up with a divide-and-conquer algorithm to calculate the matrix-vector product $H_k v$, and show that it can be calculated using $O(n \log n)$ operations. Assume that all the numbers involved are small enough that basic arithmetic operations like addition and multiplication take unit time. You do not need to prove correctness.

3 Distant Descendants

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. Describe an $O(|V|)$ algorithm to output $d[v]$ for every v. **Please give a 3-part solution; for the proof of correctness, only a brief justification is needed.**

Hint 1: write an equation to compute $d[v]$ *given the* d*-values of* v*'s children (and potentially another value).*

Hint 2: to implement what you derived in hint 1 for all vertices v*, we recommend using a graph traversal algorithm from lecture and keeping track of a running list of ancestors.*

4 Depth First Search

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 depth-first search based computations below.

```
Input: Undirected connected tree $G = (V, E)$ and positive weights $wt(u, v)$ for each edge $(u, v) \in E$

Initialization:
$visited[v] \leftarrow False$ for all vertices $v$.
$L[v] \leftarrow 0$ and $M[v] \leftarrow 0$ for all vertices $v$.

function Explore(Vertex $u$)
    $visited[u] \leftarrow True$
    for each edge $(u, v)$ in $E$ do
        if NOT $visited[v]$ then
            PreVisit(u,v)
            Explore($v$)
            PostVisit(u,v)
```

DFS can be used for different purposes by defining the procedures PreVisit and PostVisit appropriately. In each of the following cases, write down pseudocode for PreVisit and PostVisit routines to perform the computation needed.

(a) 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 $L[v]$.

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

(c) For each vertex v, compute the depth of the tree rooted at vertex v, i.e., the length of the longest path from v to a leaf in its subtree, and store it in $L[v]$.

(d) For each vertex v, compute the maximum degree among all the children of v and store it in $L[v]$

5 Topological Sort Proofs

(a) 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) 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) This subpart is unrelated to the notion of semiconnectness. 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.

6 [Coding] DFS & Edge Classification

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-fa24-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.