CS 630, Fall 2024, Homework 7 Solutions
Due *Tuesday*, December 10, 2024, 11:59 pm EST, via Gradescope

Homework Guidelines

Collaboration policy Collaboration on homework problems, with the exception of programming assignments and reading quizzes, is permitted, but not encouraged. If you choose to collaborate on some problems, you are allowed to discuss each problem with at most 5 other students currently enrolled in the class. Before working with others on a problem, you should think about it yourself for at least 45 minutes. Finding answers to problems on the Web or from other outside sources (including generative AI tools or anyone not enrolled in the class) is strictly forbidden.

You must write up each problem solution by yourself without assistance, even if you collaborate with others to solve the problem. You must also identify your collaborators. If you did not work with anyone, you should write "Collaborators: none." It is a violation of this policy to submit a problem solution that you cannot orally explain to an instructor or TA.

Typesetting Solutions should be typed and submitted as a PDF file on Gradescope. You may use any program you like to type your solutions. LaTeX, or "Latex", is commonly used for technical writing (`overleaf.com` is a free web-based platform for writing in Latex) since it handles math very well. Word, Google Docs, Markdown or other software are also fine.

Solution guidelines For problems that require you to provide an algorithm, you must provide:

1. pseudocode and, if helpful, a precise description of the algorithm in English. As always, pseudocode should include
   - A clear description of the inputs and outputs
   - Any assumptions you are making about the input (format, for example)
   - Instructions that are clear enough that a classmate who hasn't thought about the problem yet would understand how to turn them into working code. Inputs and outputs of any subroutines should be clear, data structures should be explained, etc.

   If the algorithm is not clear enough for graders to understand easily, it may not be graded.

2. a proof of correctness,

3. an analysis of running time and space.

You may use algorithms from class as subroutines. You may also use facts that we proved in class.

You should be as clear and concise as possible in your write-up of solutions. A simple, direct analysis is worth more points than a convoluted one, both because it is simpler and less prone to error and because it is easier to read and understand.

Problem 1 *Graph centrality (10 points)*

Recall from class the definition of betweenness centrality. Let $G(V, E)$ be an undirected unweighted graph, you may assume that G is connected.

We denote the *number of shortest paths* between two vertices s and t with $p(s, t)$, note that this is a symmetric notation as G is undirected. We also use the notation $p(s, t|v)$ to denote the number of shortest paths between s and t that contain vertex v. The betweenness of v is the fraction of shortest paths it covers for each pair of vertices

$$B(v) = \sum_{s \neq t \in V} \frac{p(s, t|v)}{p(s, t)}$$

In this problem you will develop an algorithm to efficiently compute the betweenness centrality of vertices.

1. Use Breadth First Search (BFS) to find the distance from a source s to each node. You may treat BFS as a blackbox when calling it as a subroutine in your algorithm. The running time of BFS is $O(|V| + |E|)$. Use the syntax $d_s \leftarrow BFS(G, s)$. Here d_s is an array of lists, such that the list in $d_s[i]$ contains the vertices at distance i from s. With this notation $d_s[0]$ consists of only s itself. (Think about it for yourself: what is the largest index in d_s?)

   Compute $p(s, t)$ for each pair of vertices $s \neq t$. For ease of notation you may want to store these values in an $O(|V|^2)$ hash table P, with $P[(s, t)] = p(s, t)$. *(Write pseudo code* [1] *for your algorithm and analyze its running time. No space complexity is needed. Prove the correctness of your algorithm, you may reference the correctness of BFS without proof.)*

2. Let *path* be a shortest path from s to t. Let x be a vertex on *path*. Then we us the notation $path_{sx}$ and $path_{xt}$ to denote the two sections of *path* from s to x and t to x.

   Prove the following statement: If a vertex v is on *path* then $path_{sv}$ and $path_{vt}$ are shortest path between the respective vertices.

3. Design an algorithm to compute the table B, such that $B[v] = B(v)$. You should use your algorithm from part (a.) as a subroutine. For full credit your algorithm should run in time $O(|V|^3)$. *(Write pseudo code for your algorithm and analyze its running time, no space complexity analysis is needed. The proof from part (b.) serves as the proof of correctness for this algorithm.)*

Problem 2 *Centrality using an adjacency matrix (10 points)*

Let A be the adjacency matrix of an undirected graph with $|V|$ vertices.

write (short) pseudo code, analyze its running time - use k in your analysis - and give a short proof for both parts (a) and (b).

[1] In case you need a reminder, at the end of this document you can find the tex syntax for an algorithm template.

1. Compute the number of shortest paths between every pair of vertices using matrix multiplication. Compute running time if we know that the longest shortest paths in G consists of k edges.

2. Compute the betweenness centrality of the vertices using the adjacency matrix. You may call part (a) as a subroutine.