Midterm 1 Part B, CS512 Spring 2025

Name:

Section:

NetID:

Do all problems, in any order.
Show your work. An answer alone may not receive full credit.

Full Question List

Problem Number	Possible Points	Points Earned Part A	Points Earned Part B
1 (Primality testing)	50		
2 (Graph)	50		
Total Points	100		

Part A to be solved in class. Part B to be solved by groups.

Honor Code

Students may discuss and work on homework problems in groups. This is encouraged. However, for individual assignments, each student must write down the solutions independently. For group assignments, it is expected that each individual student will be able to show an understanding of the solution well enough in order to reconstruct it by him/herself. Students should clearly mention the names of all the other students who were part of their discussion group. Using code or solutions from the web is not allowed unless it is an open source library that is required for completion of the class project. However, explicit credit should be given with a clear specification of your contributions. We check all the submissions for plagiarism. We take the honor code seriously and expect students to do the same.

I/We acknowledge and accept the Honor Code.

Group Leader (Signed) Apoorv Bankey

Group Member (Signed) Herik Patel

Group Member (Signed) Krish Jetly

If you are not printing this document out, please type your initials above.

1) **(50 points. Primality testing)**
Given three binary numbers. One of these is a prime, and the other two are composite numbers. Your task is to:

- Write a program to determine whether each of the given numbers is a prime number.
- Explain the logic behind the program you wrote.
- Describe the algorithm used in your solution.

Your **inputs** are accessible in OneDrive folder Named "`Group-[Your_Group_Number]`".
What to submit:

- Your program source code.
- A README.txt file explains your algorithm and your implementation.
- A result.txt file of three lines, indicating the primality of the three input numbers.
- A short (one-minute) demo.mp4 video shows how your code works.

2) **(50 points. Graph)** You are given a transcript from a lecture recording. Your task is to generate a directed graph based on the sentences in the transcript.

Each node represents a set of words from a sentence. Nodes are connected if their sentences share common words, with directed edges labeled by the shared words.

Your task is to:

- Build a directed graph based on the sentences, where:
  - Each node is a collection of words appearing in a sentence.
  - Two nodes are connected if their corresponding sentences share some common words, directed from the earlier sentence to the latter one.
  - The edge is labeled by the shared words, weighted by the number of shared words.
- Print the number of vertices, the number of edges, and the number of connected components (for this part you may want to treat the graph as undirected. You may also want to compute the undirected degree distribution).
- Plot the weighted in-degree and out-degree distribution (weighted by edge weights) of the graph.
- Find the strongly connected components (SCCs) in the graph. Draw the DAGs of SCCs in non-decreasing order of the number of vertices.
- Plot the distribution of the SCCs by size in the number of vertices and the number of edges.
- Convert the graph to an undirected graph. Compute the bi-connected components (BCCs) in the graph. Draw the BCCs forest.

Note:

- Punctuation marks, spaces, line breaks, and stop words[1] are ignored when constructing the graph.

[1]spaCy stop_words.py

- You could use Python modules like `pypdf`[2] for extracting texts from PDF files, `NLTK`[3], `spaCy`[4] for natural language processing, `igraph`[5], `NetworkX`[6] for graph processing, and JavaScript library 2D `force-graph`[7], `3D Force-Directed Graph`[8]. Of course, you are welcome to use any other libraries of your choice.
- A Python script for natural language preprocessing is available in OneDrive.

Your **inputs** are accessible in OneDrive folder named "`Group-[Your_Group_Number]`".
What to submit:

- Your program source code for all tasks.
- A size.txt file indicates the number of vertices, the number of edges, and the number of connected components.
- A degree-dist.png file plots the undirected degree distribution.
- An in-deg-dist.png and an out-deg-dist.png files plot the weighted degree distributions.
- An scc-dag.png file draws the DAGs of SCCs.
- An scc-size.png file plots the size of SCCs.
- A bcc-forest.png file draws the BCCs forest.
- A short (two-minute) demo.mp4 video file shows the interaction with your drawings.

[2] https://github.com/py-pdf/pypdf
[3] https://www.nltk.org/
[4] https://spacy.io/
[5] https://igraph.org/
[6] https://networkx.org/
[7] https://github.com/vasturiano/force-graph
[8] https://github.com/vasturiano/3d-force-graph