Homework 6

CAS CS 132: Geometric Algorithms

Due: **Thursday October 26, 2023 at 11:59PM**

Submission Instructions

- Make the answer in your solution to each problem abundantly clear (e.g., put a box around your answer or used a colored font if there is a lot of text which is not part of the answer).
- Choose the correct pages corresponding to each problem in Gradescope. Note that Gradescope registers your submission as soon as you submit it, so you don't need to rush to choose corresponding pages. **For multipart questions, please make sure each part is accounted for.**

Graders have license to dock points if either of the above instructions are not properly followed.

Practice Problems

The following list of problems comes from *Linear Algebra and its Application 5th Ed* by David C. Lay, Steven R. Lay, and Judi J. McDonald. They may be useful for solidifying your understanding of the material and for studying in general. **They are optional, so please don't submit anything for them**.

-
-

1 Matrix Algebra

A. (5 points) Suppose A and B are invertible matrices and $AB^T X A^{-1} B = I$. Solve for X in terms of A and B.

B. (5 points) Show that $A + A^T$ is symmetric for any square matrix A. That is, show that $(A + A^T)^T = A + A^T$.

C. (5 points) Show that if A and B are symmetric and $AB = BA$ then AB is symmetric.

Solution.

2 Invertible Matrices

For each of the following statements, argue that they are true or give a counterexample in $\mathbb{R}^{2\times 2}$ showing they are false. All matrices are assumed to be square.

A. (3 points) If A is invertible and B is invertible then $A + B$ is invertible.

B. (3 points) If $A\mathbf{1} = \mathbf{0}$, then A not invertible.

C. (3 points) If A has zeros along it's diagonal, that is

$$A_{11} = A_{22} = \cdots = A_{nn} = 0$$

then A is not invertible.

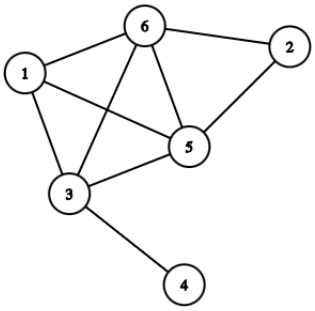
D. (3 points) If $A\mathbf{x} = B\mathbf{x}$ holds only when $\mathbf{x} = \mathbf{0}$, then $A - B$ is invertible.

E. (3 points) If A has a row of all zeros, then A is not invertible.

Solution.

3 Counting Triangles

Consider the following undirected unweighted graph.[1] For this problem, you may use NumPy or solve by hand. If you use NumPy, you must include the code you used to compute each part.



A. (4 points) Write down the adjacency matrix A for this graph.

B. (4 points) Compute A^2 and A^3.

C. (5 points)The *Hadamard product* of two $m \times n$ matrices (denoted by $A \circ B$) is entry-wise multiplication (like '$*$' in NumPy). So $(A \circ B)_{ij} = A_{ij} * B_{ij}$ for any indices i and j. Compute the value of

$$\frac{1}{6}\left(\mathbf{1}^T(A^2 \circ A)\mathbf{1}\right)$$

(Recall that $\mathbf{1}$ is the all-ones vector. In the case of this problem, it must be in $\mathbb{R}^6$.)

D. (5 points) The *trace* of a $n \times n$ matrix (denoted by $\mathsf{tr}(A)$) is the sum of the entries along it's diagonal. So

$$\mathsf{tr}(A) = A_{11} + A_{22} + \cdots + A_{nn}.$$

Compute the value of $\frac{1}{6}\mathsf{tr}(A^3)$.

[1] Image generated with Graph Editor (`https://csacademy.com/app/graph_editor/`)

Solution.

4 Regular Stochastic Matrices

For each of the following stochastic matrices:

- Determine if it is regular.
- Write down a general form solution solution to the homogenous matrix equation $(A - I)\mathbf{x} = \mathbf{0}$).
- If it is possible, find a steady state vector from this solution.

For this problem, you may use NumPy as well as code from previous parts of the course.

A. (6 points)

$$\begin{bmatrix} 1 & 1/2 & 1/3 \\ 0 & 1/2 & 1/3 \\ 0 & 0 & 1/3 \end{bmatrix}$$

B. (6 points)

$$\begin{bmatrix} 0.9 & 0.9 \\ 0.1 & 0.1 \end{bmatrix}$$

C. (6 points)

$$\begin{bmatrix} 0.2 & 0 & 0.3 \\ 0.8 & 0.5 & 0 \\ 0 & 0.5 & 0.7 \end{bmatrix}$$

Solution.

5 Sonnet Generator (Programming)

(15 points) In this problem you will be filling in an implementation of a program which uses random walks to generate poetry. This is based on the idea of the *disocciated press*, a piece of software which does this for arbitrary texts (and which is avaiable in emacs by default).

Large language models like GPT use fancy techniques to predict what word or sequence of words should follow a given piece of text. It's possible to build a very unfancy version of this using random walks. Given a corpus of text, we can build a *weighted directed graph* whose nodes are words, and whose edge indicate word adjacency in the corpus. Each edge can further be labeled with the number of word adjacencies in the corpus. So words which tend to be next to each other will have larger weights on their edges.

As an example, suppose we used the statement "a dog and a cat and a bird" as our corpus. We can build a graph with nodes for each word ("a", "dog", "and", "cat" and "bird") and with edges for each pair of adjacent words ("a" to "dog", "dog" to "and", "and" to "a", and so on). When we look at the edge from "and" to "a", it should have weight 2 because "and a" appears twice in the statement.

Rather than using a single statement for our corpus, we'll use the entirety of Shakespeare's sonnets. We will then use this to generate new (nonsense) sonnets by taking a random walk on this graph and collecting words along the way.

The process is roughly as follows:

- Read in the text of Shakespeare's sonnets and build the adjacency matrix for the graph described above.
- Convert this matrix into a stochastic matrix by dividing each column by its sum. This will make the rest of the implementation easier.
- Perform a random walk on this matrix, keeping track of the nodes you've visited so far.
- Use that list of nodes to generate a list of words which will make up the poem. Then format that poem so that it can be nicely printed.

You are given starter code in the file `hw06prog.py`. **Don't change the name of this file when you submit.** Also don't change any of the names of functions included in the starter code. **The only changes you should make are to fill in the TODO items in the starter code.** Most of the above process in implemented for you. All together you have to fill in two functions:

- `adjacency_to_stochastic` which converts an adjacency matrix into a stochastic matrix by dividing its columns by their sums (Hint. Make sure to look at and understand `numpy.sum`).
- `random_walk` which performs a random walk of a given length and returns the list of nodes visited. There is already an implementation of a *single*

random step in the function `random_step`. You should use this function and collect the outputs into a list.

You will upload a single file `hw06prog.py`. You will also be provided with a text file `sonnets.txt` which contains the text of the entirety of Shakespeare's sonnets. **You do not need to upload this when you submit**, you just need to upload `hw06prog.py`.