HOMEWORK 5
LINEAR RECURRENCE AND MATRIX MULTIPLICATION

1. Introduction

Let A be an $m \times n$ matrix and B be an $n \times p$ matrix. The product $C = AB$ is defined as an $m \times p$ matrix where each element c_{ij} is given by:

$$c_{ij} = \sum_{k=0}^{n-1} a_{ik} b_{kj}$$

Here:

- c_{ij} is the element in the i-th row and j-th column of matrix C.
- a_{ik} is the element in the i-th row and k-th column of matrix A.
- b_{kj} is the element in the k-th row and j-th column of matrix B.

Example 1.1. Consider the following matrices:

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, \quad B = \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$$

To compute the product $C = AB$:

$$C = \begin{bmatrix} 1 \cdot 5 + 2 \cdot 7 & 1 \cdot 6 + 2 \cdot 8 \\ 3 \cdot 5 + 4 \cdot 7 & 3 \cdot 6 + 4 \cdot 8 \end{bmatrix} = \begin{bmatrix} 19 & 22 \\ 43 & 50 \end{bmatrix}$$

2. Some helpful resources for this week lecture

The following blog posts explains quite well the relationship between linear recurrence and matrix multiplication.

`https://krinkinmu.github.io/2020/10/23/recurrence-relations-and-linear-algebra.html`

The following chapter gives more explanations for the lattice problem. Please note that the author discusses the move from top left to bottom right but if we rotate the box 90 degree counterclockwise, it is the same as moving from lower left to top right. In other words, the two problems are equivalent.

`https://bradfieldcs.com/algos/recursion/dynamic-programming/`

3. Written questions

Problem 1. Calculate the following matrix product AB. If the product is not well-defined, please explain why.

(1)

$$A = \begin{bmatrix} 1 & -1 \\ 2 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 2 & -3 \\ 4 & 5 \end{bmatrix}$$

(2)

$$A = \begin{bmatrix} 1 & -3 \\ -2 & 4 \end{bmatrix}, \quad B = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

(3)

$$A = \begin{bmatrix} 1 & -3 & 1 \\ -2 & 4 & -1 \end{bmatrix}, \quad B = \begin{bmatrix} 1 & 2 \\ -1 & -3 \\ 0 & 1 \end{bmatrix}$$

Problem 2. You start at the point $(0,0)$ on a 2D grid and aim to reach the point $(n,0)$. You are allowed to move **rightward** along the x-axis, where each rightward movement can cover any positive integer distance. You cannot move upward or downward. Let a_n be the number of all possible paths.

Let take an example with $n = 3$. The following paths illustrate the different valid movements from $(0,0)$ to $(3,0)$:

1. **Direct Move**:
   - Move directly from $(0,0)$ to $(3,0)$.
     - **Path 1**: $(0,0) \to (3,0)$

2. **Intermediate Moves**: You can move to any point $(k,0)$ where $1 \leq k < 3$, and then make the final move to $(3,0)$. The possible paths are:
   - Move to $(1,0)$ first.
     - **Path 2**: $(0,0) \to (1,0) \to (3,0)$
     - **Path 3**: $(0,0) \to (1,0) \to (2,0) \to (3,0)$
   - Move to $(2,0)$ first.
     - **Path 4**: $(0,0) \to (2,0) \to (3,0)$

(1) Show that $a_n = a_{n-1} + a_{n-2} + \ldots + a_0$. Here, the convention is that $a_0 = 1$.

(2) While the above relationship does not look like it has a finite order, show that $a_n = 2a_{n-1}$ for $n \geq 2$.

(3) Show that $a_n = 2^{n-1}$ for all $n \geq 1$.

Problem 3. Let a_n be a sequence given by the following linear recurrence

$$a_0 = 1, a_1 = 2, a_2 = 4, a_3 = 0, a_n = a_{n-1} + 2a_{n-2} - 5a_{n-3} + a_{n-4} \quad \forall n \geq 4.$$

(1) What is the order of this linear recurrence?

(2) Find the matrix A such that

$$\begin{bmatrix} a_n \\ a_{n-1} \\ a_{n-2} \\ a_{n-3} \end{bmatrix} = A \begin{bmatrix} a_{n-1} \\ a_{n-2} \\ a_{n-3} \\ a_{n-4} \end{bmatrix}$$

4. Coding questions

Problem 4. Given two matrices A, B of size $m \times n$ and $n \times p$ respectively (implemented using numpy arrays). Write a function named `matrix_multiplication(A,B)` that returns that product AB. For this problem, please do not use the np.dot() method. The goal of this exercise is to understand how matrix multiplication works.

Problem 5. Let a_n be a sequence described in Problem 3. Find the n-th term of this sequence using matrix multiplication.

Problem 6. The following problem is a modification of the lattice problem that we did in class.

You are on a 2D grid, starting at the origin point $(0, 0)$. Your objective is to reach the point (m, n) using specific movement rules. You can move in the following ways:

- **Rightward Move**: Move one step right from (x, y) to $(x + 1, y)$.
- **Upward Move**: Move one step up from (x, y) to $(x, y + 1)$.
- **Diagonal Move**: Move one step diagonally up and to the right from (x, y) to $(x + 1, y + 1)$.

Let $A(m, n)$ represent the total number of distinct paths you can take to reach the point (m, n) from the origin $(0, 0)$ using any combination of the allowed moves. Find $A(m, n)$. What is the big-O performance of your algorithm?

For example, for $(m, n) = 1$ we have $A(1, 1) = 3$. All distinct paths are

(1) **Path 1**: $(0, 0) \to (1, 1)$ (Diagonal Move)
(2) **Path 2**: $(0, 0) \to (1, 0) \to (1, 1)$ (Right then Up)
(3) **Path 3**: $(0, 0) \to (0, 1) \to (1, 1)$ (Up then Right)