Data Structures

Homework Assignment 4 - Dynamic Array

Problem 1 - Intersection and Union - 25 Points
Problem 2 - Mirror and Rotate Matrices - 75 Points

25% of Gradescope Autograder test cases are hidden for this assignment.

Fall 2024 - Revision 1 - *October*

Problem 1 - Intersection and Union - 25 Points

Part A - Intersection - 5 Points

Given two strictly increasing lists, L1 and L2, of length n, write a Python function to compute L1 ∩ L2 using only basic Python operations.

Requirements

- You cannot use Numpy or import other Python libraries.
- You cannot cast L1 and L2 to sets or use Python sets.
- You must use basic Python functions and lists.
- You don't have to sort your results.
- Your function has to be **O(n)** time complex.

Part B - Union - 5 Points

Given two strictly increasing lists, L1 and L2, of length n, write a Python function to compute L1 ∪ L2 using only basic Python operations.

Requirements

- You cannot use Numpy or import other Python libraries.
- You cannot cast L1 and L2 to sets or use Python sets.
- You have to use basic Python functions and lists.
- You don't have to sort your results.
- Your function has to be **O(n)** time complex.

Part C - UserDefinedDynamicArray - 15 Points

Solve the problem of parts A and B using the dynamic array class from Recitation 4. Introduce the function $intersect(self, b)$ and $union(self, b)$ as new member functions in the dynamic array class.

Requirements

- You cannot use Numpy or import other Python libraries.
- You cannot cast b to a set or use Python sets.
- You have to use basic Python functions and lists.
- You don't have to sort your results.
- You cannot use Python lists.
- Your function has to be **O(n)** time complex.

Problem 1 - Mirror and Rotate Matrices - 75 Points

You can not use any third-party libraries to solve these problems. You can not use import.

Part A - Mirror Matrice - 25 Points

Implement the Python function $mirror(M)$, which mirrors the provided nxn matrix M vertically.

Example

Input: [[1, 2, 3, 4, 5], [6, 7, 8, 9, 10], [11, 12, 13, 14, 15], [16, 17, 18, 19, 20], [21, 22, 23, 24, 25]]
Output: [[5, 4, 3, 2, 1], [10, 9, 8, 7, 6], [15, 14, 13, 12, 11], [20, 19, 18, 17, 16], [25, 24, 23, 22, 21]]

Part B - Rotate Matrice - 25 Points

Implement the Python function $rotator(M, a, d)$, which rotates the provided nxn matrix M by a given angle a in the specified direction d.

Important

- $a \in \{ 0, 90 , 180\}$
- $d \in \{clockwise, anticlockwise\}$

Requirements

- You cannot change the input parameters.
- You have to return a new matrix.

Example

Input: [[1, 2, 3, 4, 5], [6, 7, 8, 9, 10], [11, 12, 13, 14, 15], [16, 17, 18, 19, 20], [21, 22, 23, 24, 25]]
Output: [[5, 10, 15, 20, 25], [4, 9, 14, 19, 24], [3, 8, 13, 18, 23], [2, 7, 12, 17, 22], [1, 6, 11, 16, 21]]

Part C - Rotate Matrice In-Place - 25 Points

Implement a Python function to solve the problem of Part B in place.

Requirements

- Your solution has to be completed in place. Compute and return your result in M.
- You are only allowed to use a constant amount of extra space.
- Your solutions for part B and part C can not be identical.