Data Structures: Arrays, Matrices, and Sets

September 10, 2024

1 Array

1.1 Definition

An array is a data structure consisting of a collection of elements (values or variables), each identified by an array index. Arrays are used to store multiple items of the same type together, making it easy to access elements based on their position.

1.2 Main Operations

- **Access:** Access an element using its index.
  Time Complexity: $O(1)$ (constant time).

- **Insert:** Insert a new element at the end or a specific index. Inserting at the end is efficient, while inserting at the beginning or in the middle requires shifting elements.
  Time Complexity:

  – Insertion at the end: $O(1)$

  – Insertion at an arbitrary position: $O(n)$

- **Delete:** Remove an element from the array. If the element is removed from the beginning or middle, shifting elements is necessary.
  Time Complexity: $O(n)$

- **Search:** Find an element by value.
  Time Complexity: $O(n)$ (linear search)

1.3 Diagram

0	1	2	3	4	5	6	7

2 Matrix

2.1 Definition

A matrix is a 2D array, typically organized in rows and columns. Matrices are commonly used in mathematical computations, including linear algebra and graph theory. Each element in a matrix can be accessed by a pair of indices: the row and column numbers.

2.2 Main Operations

- **Access:** Access an element using row and column indices.
  Time Complexity: $O(1)$
- **Insert:** Insert a new row or column in the matrix, which requires shifting elements.
  Time Complexity: $O(m \times n)$ (where m and n are the dimensions of the matrix)
- **Delete:**
  - If implemented as a static 2D array, removing a row or column requires shifting elements, leading to a time complexity of $O(m \times n)$.
  - If implemented as a vector of vectors, removing a row is simpler because each row is an independent vector. Removing a row involves removing a single element from the outer vector, making it $O(m)$, where m is the number of rows.
- **Search:** Find an element in the matrix by value.
  Time Complexity: $O(m \times n)$
- **Transpose:** Flip the matrix over its diagonal (row i becomes column i).
  Time Complexity: $O(m \times n)$

2.3 Diagram

1	2	3
4	5	6
7	8	9

3 Set

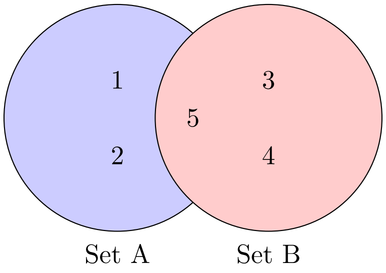
3.1 Definition

A set is an unordered collection of distinct elements. Unlike arrays or matrices, sets do not allow duplicate values. They are commonly used to represent collections where each element is unique.

3.2 Main Operations

- **Insert:** Add a new element to the set.
  Time Complexity: $O(1)$ on average (if implemented using a hash set), $O(\log n)$ if using a balanced binary tree.
- **Delete:** Remove an element from the set.
  Time Complexity: $O(1)$ on average (if using a hash set), $O(\log n)$ for a balanced binary tree.
- **Search (Contains):** Check if an element is present in the set.
  Time Complexity: $O(1)$ on average (hash set), $O(\log n)$ for a balanced binary tree.
- **Union:** Combine two sets into a single set containing all elements from both.
  Time Complexity: $O(n + m)$ (where n and m are the sizes of the sets).
- **Intersection:** Find elements that are present in both sets.
  Time Complexity: $O(\min(n, m))$
- **Difference:** Find elements in one set that are not in another.
  Time Complexity: $O(\min(n, m))$

3.3 Diagram


1
3
5
2
4
Set A
Set B