Revision Notes: Introduction to Arrays and Array Manipulations

1. Introduction to Arrays

- **Definition**: An array is a collection of elements, all of the same type, stored in contiguous memory locations. Arrays allow easy access to their elements via indices.
- **Declaration**: The declaration of an array involves specifying the data type of the elements, the name of the array, and the size (number of elements it can hold). Example for an integer array: `int ages[5];`.
- **Usage Scenario**: Arrays are useful when multiple values of the same type need to be managed, for instance, ages of students or scores in a game【6:19†transcript.txt】.

2. Array Access and Traversal

- **Accessing Elements**: Elements in arrays are accessed via indices starting from 0 up to (n-1), where n is the size of the array. For example, `ages[0]` would give you the first element of the array.
- **Traversal**: Arrays can be traversed using loops such as `for` loops. Iterating over an array allows operations to be performed on each element【6:19†transcript.txt】.

3. Reversing an Array

- **Concept**: Reversing an array involves swapping elements from the start to the end until you reach the center.
- **Algorithm**:
  - Use two indices, one starting at the beginning (`i=0`) and the other at the end (`j=n-1`).
  - While `i < j` , swap elements at position `i` and `j` , then increment `i` and decrement `j` .
  - Continue the process until `i` reaches or exceeds `j` .

【6:8†transcript.txt】【6:9†transcript.txt】.

4. Array Rotation

Right Rotation by K Elements

- **Steps**:
  1. Reverse the entire array.
  2. Reverse the first `K` elements.
  3. Reverse the remaining `N-K` elements.
- **Example**: Given an array `[1,2,3,4,5,6]` and `K=4`, after performing the above steps, the array becomes `[3,4,5,6,1,2]`.
- **Complexity**:
  - **Time Complexity**: O(N), since each element is involved in constant-time operations.
  - **Space Complexity**: O(1), as the operation is done in place with no extra space【6:1†transcript.txt】【6:12†transcript.txt】.

Left Rotation by K Elements

- **Conceptualization**: Equivalent to rotating right by `N-K` elements.
- **Implementation**: Similar approach to right rotation as described above but reversed【6:15†transcript.txt】.

5. Importance of Constraints

- **Understanding Limits**: Knowing the constraints, such as maximum possible size of the array or time allowed for computation, helps in developing efficient algorithms.
- **Real-world Relevance**: Predicting the complexity (Big O notation) allows determination of whether an algorithm is fit for given constraints, e.g., will the algorithm handle up to `10^5` operations?【6:18†transcript.txt】.

6. Additional Concepts

- Examples across languages include Python's list, Java's `ArrayList`, and C++'s `vector` 【6:10†typed.md】.

This session covered a foundational understanding of arrays, how to manipulate them with operations such as reversing and rotating, and the importance of constraints in algorithm design.