**Array Representation in Memory:**

Arrays are contiguous blocks of memory, meaning that all elements are stored sequentially. The memory address of each element can be calculated using the starting address of the array and the index of the element, making access time constant (O(1)). This representation is efficient for accessing elements but can be inefficient for certain operations like insertion and deletion, especially if they involve shifting elements.

**Advantages of Arrays:**

* **Constant Time Access:** Direct access to elements using their index.
* **Memory Efficiency:** No overhead for storing pointers or additional data structures.
* **Predictable Iteration:** Arrays provide a straightforward way to iterate over elements.

**Time Complexity:**

* **Add Operation:** O(1) if there is space in the array (amortized O(n) if resizing is needed).
* **Search Operation:** O(n) in the worst case, as it requires traversing the array.
* **Traverse Operation:** O(n), as each element needs to be accessed.
* **Delete Operation:** O(n), since finding the element and shifting others require traversal.

**Limitations of Arrays:**

* **Fixed Size:** The array size must be declared upfront. Resizing is costly in terms of time and memory.
* **Inefficient Insertions/Deletions:** These operations can require shifting elements, which is O(n) in time complexity.
* **Memory Allocation:** If the array is not fully utilized, it leads to wasted memory.

**When to Use Arrays:**

* When the number of elements is known and fixed.
* When frequent access to elements is required (due to O(1) access time).
* When memory overhead needs to be minimized, and dynamic resizing is not a concern.