**1. Understand Array Representation**

**Array Representation in Memory:**

* **Memory Layout:** Arrays are contiguous blocks of memory. This means that each element in the array is stored next to each other in memory. The array's starting address is the address of the first element, and subsequent elements are accessed by their index, which is calculated based on the size of the element type.
* **Indexing:** Accessing an element by its index is very fast (O(1)) because it requires a simple calculation: base\_address + (index \* element\_size).
* **Advantages:**
  + **Fast Access:** Constant-time access to elements using their indices.
  + **Memory Efficiency:** Arrays have low memory overhead because they only store the elements themselves without additional metadata.
  + **Cache Friendly:** Due to their contiguous memory layout, arrays benefit from spatial locality, improving cache performance.

**Analysis**

**Time Complexity:**

* **Add Operation:**
  + **Best Case:** O(1) (when there's space in the array)
  + **Worst Case:** O(1) (constant time, but fails if array is full)
* **Search Operation:**
  + **Best Case:** O(1) (when the first element matches)
  + **Worst Case:** O(n) (when the element is at the last position or not present)
* **Traverse Operation:** O(n) (since each element is visited once)
* **Delete Operation:**
  + **Best Case:** O(1) (when the first element matches and needs to be deleted)
  + **Worst Case:** O(n) (when the element is at the last position or not present, requiring a shift of elements)