**Time Complexity Analysis:**

* **Add Worker:**
  + **Time Complexity: O(1)**
  + **The worker is added at the next available index, which takes constant time.**
  + **If the array is full, insertion is blocked.**
* **Search Worker (by ID):**
  + **Time Complexity: O(n)**
  + **In the worst case, the entire array must be scanned to find a match.**
* **Traverse All Workers:**
  + **Time Complexity: O(n)**
  + **Each element is printed sequentially.**
* **Delete Worker (by ID):**
  + **Time Complexity: O(n)**
  + **The deletion involves finding the element (O(n)) and then shifting all subsequent elements left by one (O(n)), so overall complexity is O(n).**

**Limitations of Using Arrays:**

* **Fixed Size:  
  Arrays have a predefined capacity. Once full, you cannot add more items unless you manually resize or reallocate.**
* **Costly Insertions/Deletions:  
  Inserting or deleting an element in the middle requires shifting elements, which is time-consuming.**
* **Lack of Flexibility:  
  Arrays do not support dynamic resizing. To handle growth, you would need to use dynamic structures like ArrayList or LinkedList.**
* **Better Alternatives:  
  If frequent insertions, deletions, or dynamic sizing are expected, collections like ArrayList, HashMap, or LinkedList are better suited than raw arrays.**