**Algorithm Analysis**

The performance of both stack implementations array-based and linked list-based can be evaluated primarily in terms of their insertion (push) and deletion (pop) operations.

**Array-Based Stack**

* **Insertion (Push):**  
  In an array-based stack, inserting an element is done by incrementing the top index and assigning the value.  
  This operation takes constant time, O(1), since it does not depend on the number of elements.  
  However, if the array becomes full and needs resizing, that specific operation may take O(n) time temporarily (though this is rare if a static array is used).
* **Deletion (Pop):**  
  Removing an element simply decrements the top index, which also takes O(1) time.  
  There is no need to shift elements, making pop operations very efficient.

**2. Linked List-Based Stack**

* **Insertion (Push):**  
  Each new element is added at the head of the linked list by creating a new node and adjusting pointers.  
  This requires a fixed number of pointer updates, so it runs in O(1) time.  
  However, the operation involves dynamic memory allocation, which introduces a small constant-time overhead.
* **Deletion (Pop):**  
  Deleting an element from the stack involves removing the head node and updating the head pointer.  
  This also takes O(1) time, but with similar memory management overhead.

**3. Performance Comparison**

| **Operation** | **Array-Based Stack** | **Linked List-Based Stack** | **Remarks** |
| --- | --- | --- | --- |
| Push (Insertion) | O(1) | O(1) | Array version faster due to contiguous memory |
| Pop (Deletion) | O(1) | O(1) | Similar complexity, but linked list has overhead |
| Memory Usage | Fixed (predefined size) | Dynamic (grows as needed) | Linked list offers flexibility but higher usage |