**Exercise 5: Task Management System**

**Understanding the Problem**

1. **Types of Linked Lists:**
   * **Singly Linked List:**
     + **Structure:** Each node contains data and a pointer (or reference) to the next node. It starts with a head node and ends with a node pointing to null.
     + **Traversal:** You can only move forward through the list.
     + **Operations:** Adding, searching, and deleting nodes involve traversing from the head node to find the appropriate position or node.
   * **Doubly Linked List:**
     + **Structure:** Each node contains data and two pointers: one to the next node and one to the previous node. It starts with a head node and ends with a tail node pointing to null.
     + **Traversal:** You can move both forward and backward through the list.
     + **Operations:** More flexible than singly linked lists as you can traverse in both directions, making certain operations easier.

**Analysis**

1. **Time Complexity:**
   * **Add Task:**
     + **At the Head:** O(1) because you only need to adjust pointers to insert a new node at the beginning.
     + **At the End:** O(n) because you need to traverse the entire list to reach the end (unless you maintain a tail pointer in a doubly linked list, which makes it O(1)).
   * **Search Task:** O(n) because, in the worst case, you may have to check every node until you find the task or reach the end of the list.
   * **Traverse Tasks:** O(n) as you need to visit each node once to access all tasks in the list.
   * **Delete Task:**
     + **By Node Reference:** O(1) if you have direct access to the node to be deleted (just adjust pointers).
     + **By Task Value:** O(n) if you need to search for the node before deleting it.
2. **Advantages Over Arrays:**
   * **Dynamic Size:** Linked lists can grow or shrink in size dynamically without needing to resize or reallocate memory, unlike arrays which require resizing operations that can be costly.
   * **Efficient Insertions/Deletions:** Inserting or deleting elements in a linked list involves adjusting pointers and does not require shifting other elements, as is necessary in arrays when inserting or deleting elements in the middle.
   * **Memory Utilization:** Linked lists only allocate memory for the nodes that are currently in use. Arrays, however, allocate a contiguous block of memory, which may include unused space if the array is larger than needed.