**Resumo das Complexidades**

| **Operação** | **Complexidade de Tempo** | **Complexidade de Espaço** |
| --- | --- | --- |
| Adicionar Elemento | O(n) | O(1) |
| Remover Elemento | O(n) | O(1) |
| Inverter Lista | O(n) | O(1) |
| Obter Nó do Meio | O(n) | O(1) |
| Remover Duplicatas | O(n) | O(n) |
| Imprimir Lista | O(n) | O(1) |

### 1. Adicionar Elemento (addAtPosition)

* **Complexidade de Tempo**:
  + O pior caso ocorre quando você precisa percorrer toda a lista para encontrar a posição de inserção, o que leva O(n), onde n é o número de elementos na lista.
  + Se você estiver inserindo no início (posição 0), a complexidade é O(1).
* **Complexidade de Espaço**: O(1) para a nova instância do nó, pois não estamos usando espaço adicional que cresce com o tamanho da lista.

### 2. Remover Elemento (removeElement)

* **Complexidade de Tempo**:
  + O pior caso ocorre quando você precisa percorrer toda a lista para encontrar o elemento a ser removido, o que leva O(n).
* **Complexidade de Espaço**: O(1), pois não estamos usando espaço adicional que cresce com o tamanho da lista.

### 3. Inverter Lista (reverseList)

* **Complexidade de Tempo**: O(n), pois você precisa percorrer todos os nós da lista uma vez para inverter as referências.
* **Complexidade de Espaço**: O(1), pois a inversão é feita in-place e não requer espaço adicional proporcional ao tamanho da lista.

### 4. Obter Nó do Meio (getMiddleNode)

* **Complexidade de Tempo**: O(n), pois você precisa percorrer a lista para encontrar o nó do meio. O algoritmo usa dois ponteiros (um que avança duas vezes mais rápido que o outro), mas ainda assim, no pior caso, você percorre toda a lista.
* **Complexidade de Espaço**: O(1), pois não estamos usando espaço adicional que cresce com o tamanho da lista.

### 5. Remover Duplicatas (removeDuplicates)

* **Complexidade de Tempo**: O(n), pois você precisa percorrer todos os nós da lista uma vez. O uso de um **HashSet** para rastrear os valores já vistos também é O(n) no pior caso, mas a operação de inserção e verificação no **HashSet** é O(1) em média.
* **Complexidade de Espaço**: O(n) no pior caso, se todos os elementos forem únicos, pois o **HashSet** pode armazener todos os elementos da lista.

### 6. Imprimir Lista (printList)

* **Complexidade de Tempo**: O(n), pois você precisa percorrer todos os nós da lista para imprimi-los.
* **Complexidade de Espaço**: O(1), pois a impressão não requer espaço adicional que cresce com o tamanho da lista.