**Análise da Complexidade Assintótica**

* **Pilha**

|  |  |  |
| --- | --- | --- |
|  | **Push** | **Pop** |
| **Tempo** | O(1) | O(1) |
| **Espaço** | O(n) | O(1) |

1. **Push:** Tempo: O tempo necessário para adicionar um novo elemento à pilha é independente do número de elementos já presentes na pilha. Espaço: O espaço necessário para armazenar a pilha cresce linearmente com o número de elementos n na pilha.
2. **Pop:** Tempo: Remover o elemento do topo da pilha é uma operação simples que pode ser realizada em tempo constante. Espaço: O espaço necessário para armazenar a pilha permanece o mesmo após a remoção de um elemento.

* **Fila**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Enqueue** | **Dequeue** | **Rear** | **Front** |
| **Tempo** | O(1) | O(1) | O(1) | O(1) |
| **Espaço** | O(n) | O(1) | O(1) | O(1) |

1. **Enqueue:** Tempo: Adicionar um novo elemento à fila é uma operação simples que pode ser realizada em tempo constante. Espaço: O espaço necessário para armazenar a fila cresce linearmente com o número de elementos n na fila.
2. **Dequeue:** Tempo: Remover o elemento da frente da fila é uma operação simples que pode ser realizada em tempo constante. Espaço: O espaço ocupado pela fila depende apenas do número de elementos atualmente presentes na fila, não do número máximo de elementos que ela já armazenou.
3. **Rear:** Tempo: Acessar o elemento do final da fila é uma operação simples que pode ser realizada em tempo constante. Espaço: O tempo de execução do método rear não afeta o espaço utilizado pela fila.
4. **Front:** Tempo: Acessar o elemento da frente da fila é uma operação simples que pode ser realizada em tempo constante. Espaço: O tempo de execução do método front não afeta o espaço utilizado pela fila.

* **Lista** **encadeada**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Push** | **Pop** | **Insert** | **Remove** | **ElementAt** |
| **Tempo** | O(n) | O(n) | O(n) | O(n) | O(n) |
| **Espaço** | O(n) | O(1) | O(1) | O(1) | O(n) |

1. **Push:** Tempo: O tempo de execução do método push depende da posição de inserção. Se o novo elemento for inserido no início da lista, o tempo é O(1). Se for inserido no final da lista, o tempo é O(1). Inserir em qualquer outra posição da lista requer percorrer a lista até a posição desejada, o que leva um tempo O(n), onde n é o número de elementos da lista antes da inserção. Sendo assim, o tempo é O(n). Espaço: O espaço necessário para armazenar a lista cresce linearmente com o número de elementos n na lista.
2. **Pop:** Tempo: É necessário percorrer a lista até a posição do elemento a ser removido. Espaço: O espaço necessário para armazenar a lista diminui com a remoção de um elemento.
3. **Insert:** Tempo: O tempo de execução do método insert depende da posição de inserção. Se o novo elemento for inserido no início ou no final da lista, o tempo é O(1). Inserir em qualquer outra posição da lista requer percorrer a lista até a posição desejada, o que leva um tempo O(n), onde n é o número de elementos da lista antes da inserção. Sendo assim, o tempo é O(n). Espaço: O espaço necessário para armazenar a lista cresce linearmente com a adição de um novo elemento.
4. **Remove:** Tempo: Depende da posição do elemento a ser removido. Se o elemento for removido do início ou do final da lista, o tempo é O(1). Se for removido de qualquer outra posição da lista, o tempo é O(n), onde n é o número de elementos da lista antes da remoção. Isso porque é necessário percorrer a lista até a posição do elemento a ser removido. Sendo assim, o tempo é O(n). Espaço: O espaço necessário para armazenar a lista diminui com a remoção de um elemento.
5. **ElementAt:** Tempo: Para acessar um elemento em uma determinada posição, é necessário percorrer a lista até a posição desejada. Espaço: O tempo de execução do método elementAt não afeta o espaço utilizado pela lista.