1. Pilha (Stack)

1.1 Método push(int value)

Primeiro, verifica se há espaço na pilha:

if (top == stack.length - 1)

Depois, adiciona o valor no topo:

stack[++top] = value

Ambas as operações parecem ser simples, uma verificando uma condição e a

outra adicionando um valor no topo.

Complexidade: O(1), o que significa que é muito rápido e não depende da

quantidade de elementos na pilha.

1.2 Método pop()

Verifica se a pilha está vazia:

if (isEmpty()) Depois remove o elemento do topo e retorna:

return stack[top--]

Como só mexe no topo, também parece ser rápido.

Complexidade: O(1), processo simples garante eficiência de processamento.

2. Fila (Queue)

2.1 Método enqueue(int value)

Verifica se a fila está cheia:

if (size == queue.length)

Depois adiciona o valor no final da fila:

rear = (rear + 1) % queue.length; queue[rear] = value

Ambas operações simples e rápidas.

Complexidade: O(1), o que quer dizer que é eficiente em grau de

complexidade.

2.2 Método dequeue()

Verifica se a fila está vazia:

if (isEmpty())

Novamente, parece ser uma verificação direta.

Remove o primeiro elemento e atualiza a fila:

front = (front + 1) % queue.length

Como só mexe na frente da fila, é um processo rápido.

Complexidade: O(1), pois não há possibilidade de complexidade em ambos

processos.

3.2 Método pop()

Verifica se a lista está vazia:

if (head == null)

Verificação básica. Se houver apenas um nó, remove. Se houver mais, percorre

até o penúltimo nó:

while (current.next.next != null) { current = current.next; }

O que pode levar a um maior tempo de processamento.

Remove o último nó:

current.next = null

Operação simples, mas chegar até lá pode demorar.

Complexidade: O(n), porque depende do número de elementos na lista.

Conclusão Geral:

Pilha e Fila: Ambas têm operações eficientes com complexidade O(1), o que as torna ideais para cenários onde a inserção e remoção de elementos é frequente e o desempenho é crítico.

Lista Encadeada: A lista é flexível, mas as operações podem ser mais lentas, especialmente à medida que cresce. As operações parecem ser O(n), o que significa que o tempo para realizar as operações aumenta conforme a lista fica maior.