	PUSH	POP	REAR	FRONT	INSERT	REMOVE	ELEMENTAT
PILHA	O(1)	O(1)					
FILA	O(1)	O(N)	O(1)	O(1)			
LISTA ENCADEADA	O(1)	O(N)			O(N)	O(N)	O(N)

PILHA

Na pilha a complexidade é mais controlada, uma vez que eu tenho ponteiros definidos. Aqui eu não trabalhei a adição empurrando os itens já adicionando para baixo, mas sim com um ponteiro. Se eu tivesse feito o empurrão, provavelmente obteria uma Pilha com complexidade o(n) e não o(1). As adições são em ordem crescente, porém os métodos pop são decrescentes e isso permite com que a pilha funcione realmente como uma pilha

FILA

Na fila eu trabalhei com um ponteiro apontando para o último. Isso faz com que as adições sejam o(1), uma vez que eu adiciono um elemento ao final da lista e aponto o ponteiro “ultimo” para ele. De forma que eu tenha controle tanto do primeiro que seria o índice 0, quanto do último. As remoções são realizadas em o(n) pois eu preciso reorganizar minha fila. Aqui eu poderia trabalhar com um ponteiro “inicial” para evitar essa reorganização e apenas trabalhar com os ponteiros, mas eu certamente perderia espaço nos índices iniciais a medida que os “atendimentos” da fila fossem realizados. Em produção seria perder memória desnecessária e não vi problemas em deixar a complexidade o(n). Visualizações de último e início são por ponteiros então a complexidade é o(1).

LISTA ENCADEADA

Essa sem dúvida foi mais complicada de implementar, tendo o método push com complexidade o(1) e as remoções, inserções, remoções e visualizações com o(n) uma vez que eu precisava percorrer os nós para encontrar, inserir, remover ou obter os elementos dos nós.