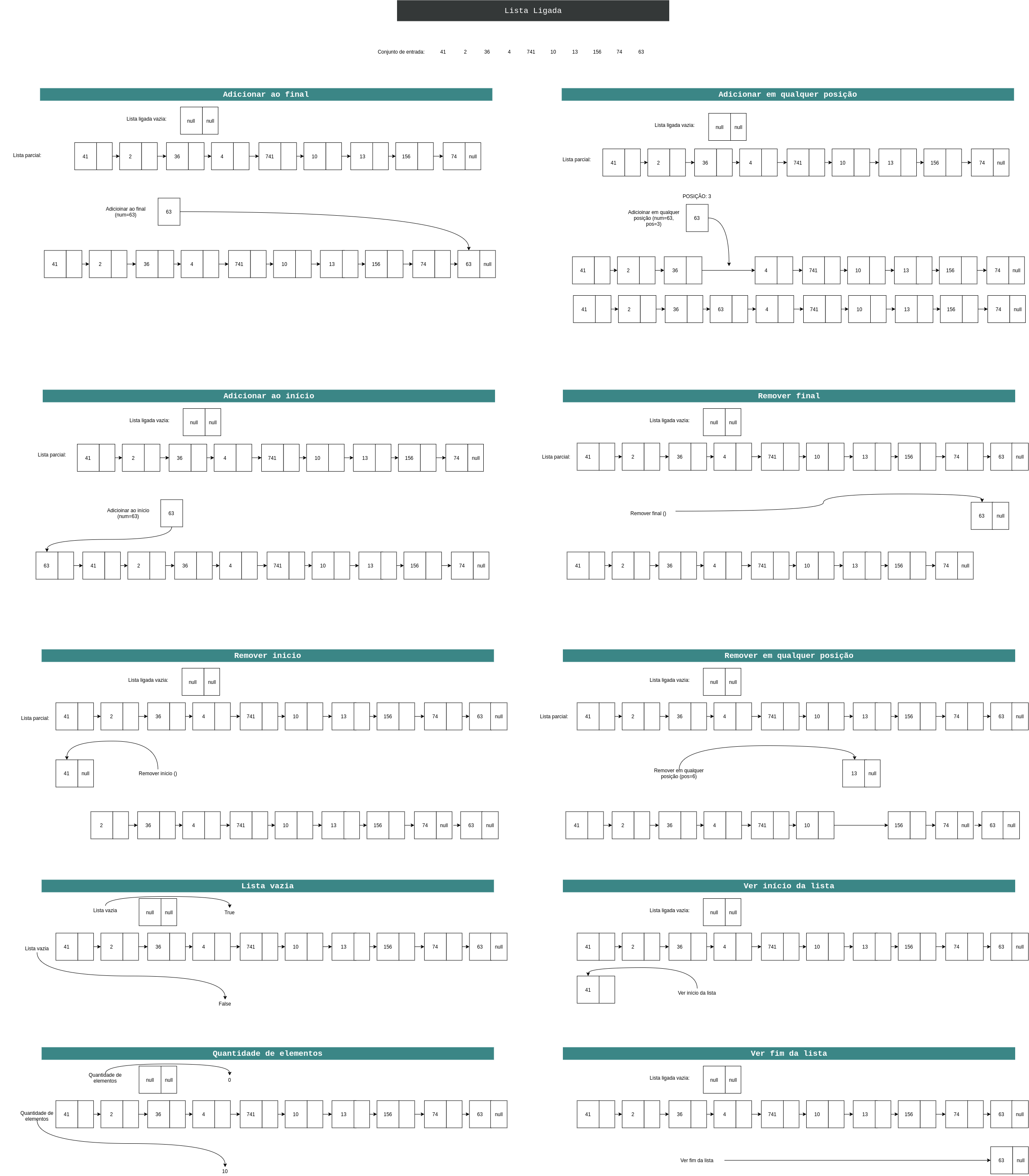
# Exercício 1

## Lista **Ligada**

Estrutura de dados organizada de forma linear e dinâmica. Os elementos de uma lista ligada possuem uma relação de ordem, pois indicam qual é o próximo elemento. Além disso, essa estrutura é dinâmica, uma vez que a alocação de memória é feita em tempo de execução e inexiste limitação de tamanho da estrutura (a limitação é memória física).

Simulações



## **Pilha**

Estrutura de dados embasada no conceito do Último que entra Primeiro que sai (LIFO). Assim, a inserção e remoção de dados dessa estrutura são realizadas em apenas uma extremidade.

**Simulações**

### 

## **Fila de Prioridades**

Estrutura de dados baseada no conceito de Primeiro que entra Primeiro que sai (FIFO). Os dados são movimentados, restritamente, apenas nas duas extremidas da estrutura, ou seja, a inserção dos elementos ocorre apenas no final da fila e a remoção no início. Em caso de remoção, todos os registros deverão ser movimentados.

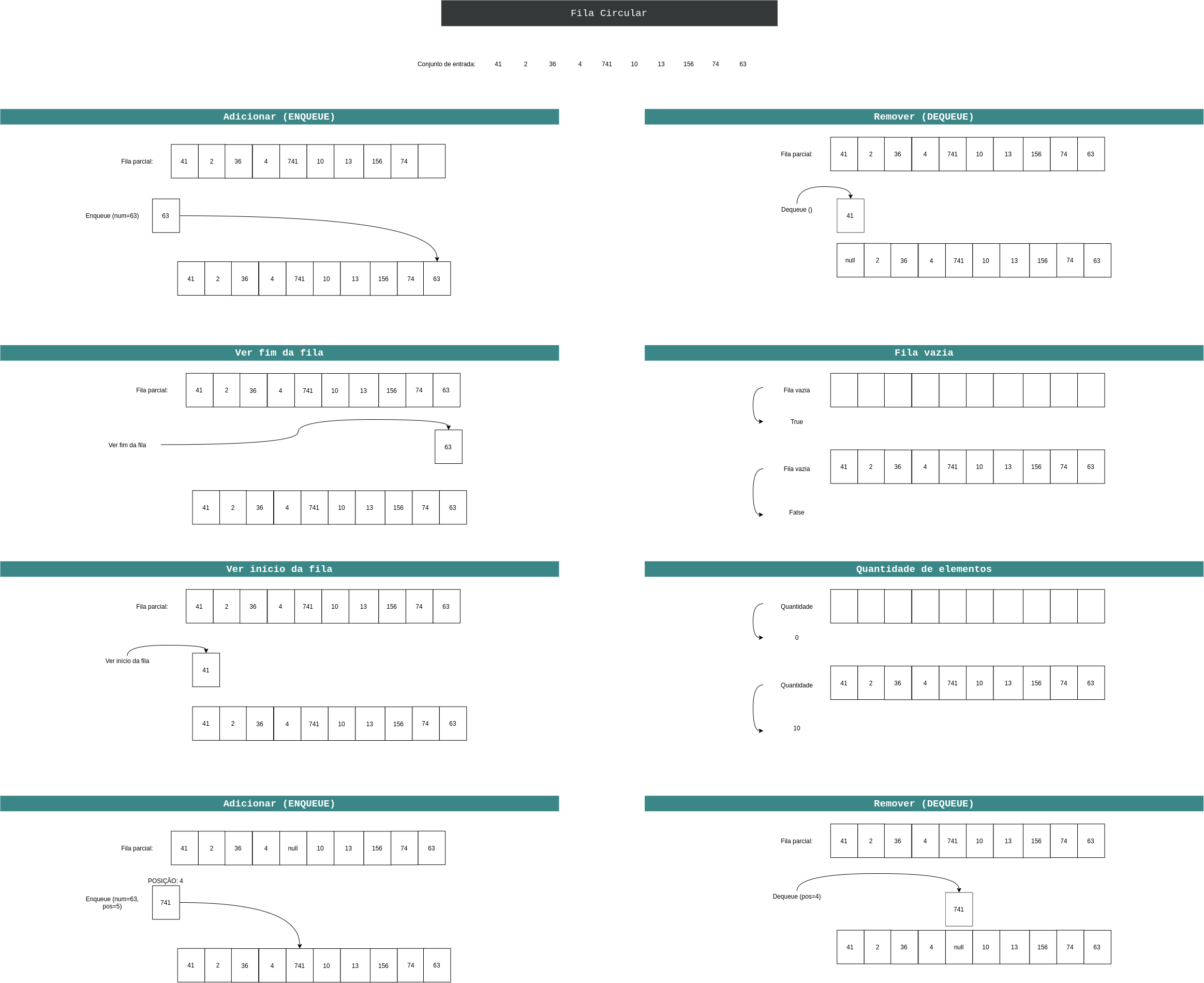
### Simulações



## **Fila circular**

Utiliza o mesmo conceito da Fila de prioridades, entretanto, permite que qualquer elemento em uma determinada posição seja removido, assim, essa posição ficará vazia e disponível para inserçao de um novo elemento. Nessa operação, os outros registros permanecem nas suas posições e não ocorre qualquer movimentação.

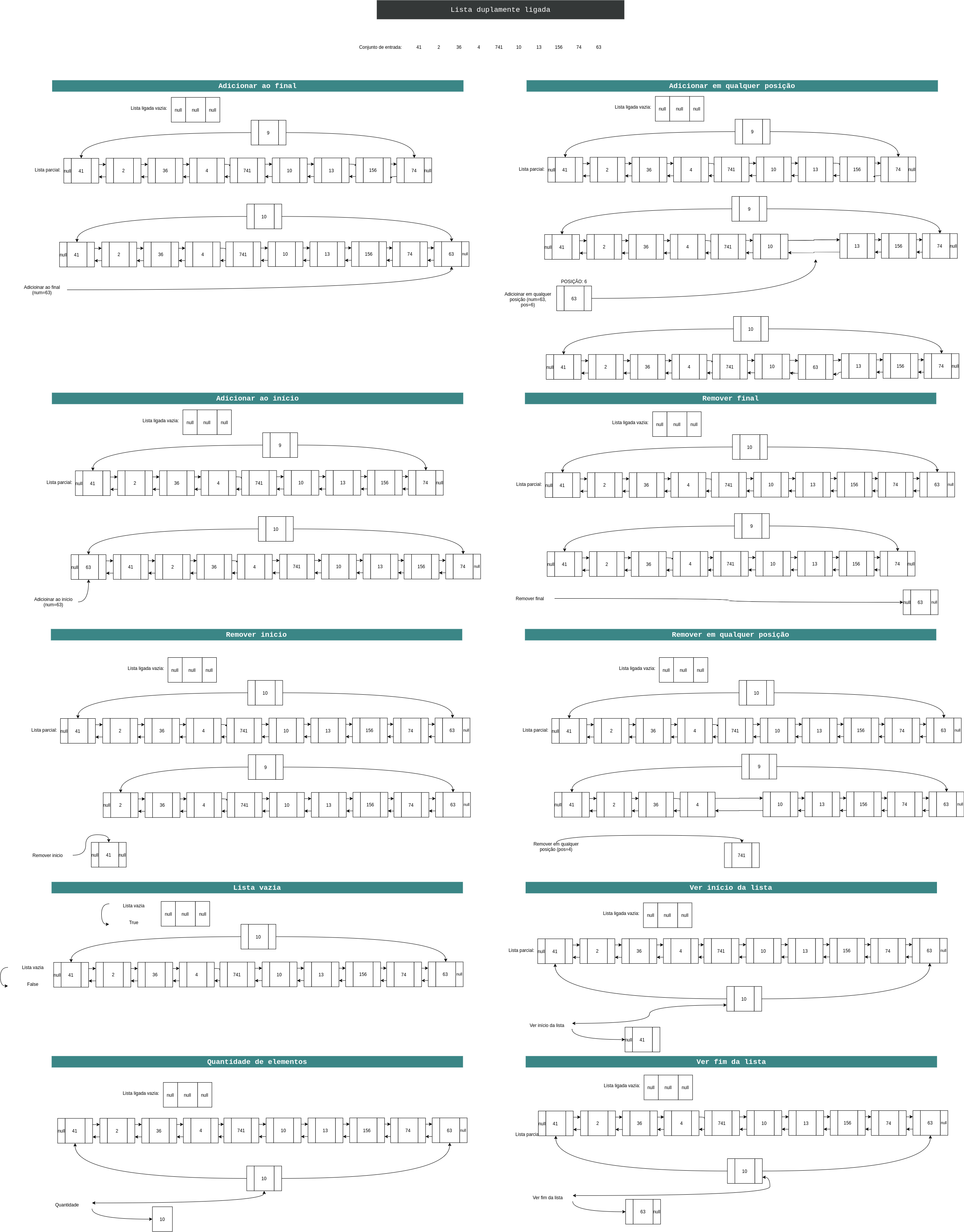
### Simulações



## **Lista Duplamente Ligada**

Estrutura de dados organizada de forma linear e dinâmica. Os elementos de uma lista duplamente ligada possuem uma relação de ordem, pois indicam qual é o anterior e o próximo elemento. Além disso, essa estrutura é dinâmica, uma vez que a alocação de memória é feita em tempo de execução e inexiste limitação de tamanho da estrutura (a limitação é memória física)

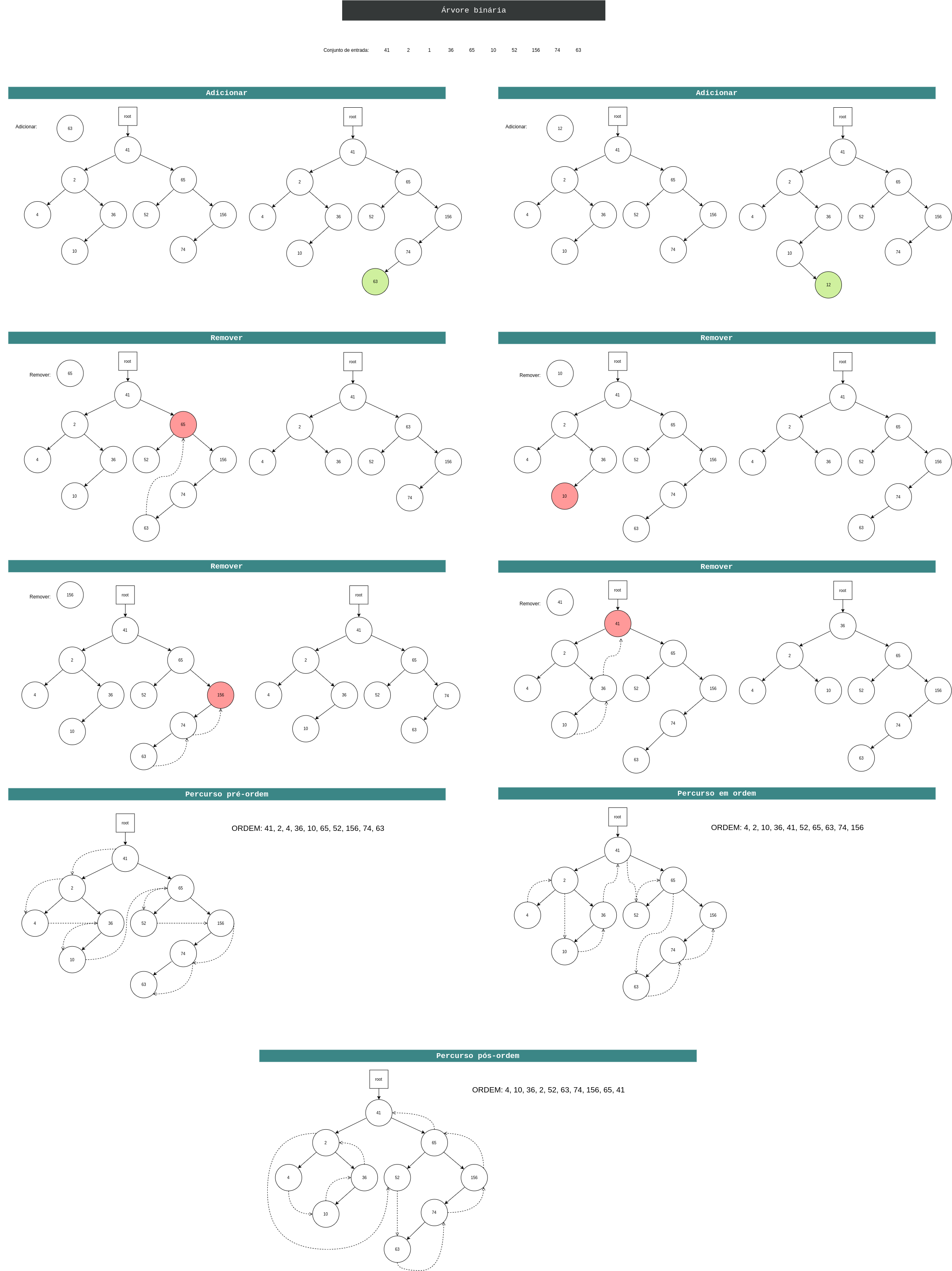
### Simulações



## **Árvore binária**

Estrutura de dados não linear, bidimensional e organizada de forma hierárquica. Cada elemento de uma árvore pode ser denominado como um nó (vértice da árvore), sendo que essa estrutura possui um ponteiro para o elemento alocado no topo da árvore, chamado de raiz. Além disso, cada nó de uma árvore possui dois ponteiros para os nós filhos, denominados esquerda e direita.

### Simulações

1. 

## **Grafos**

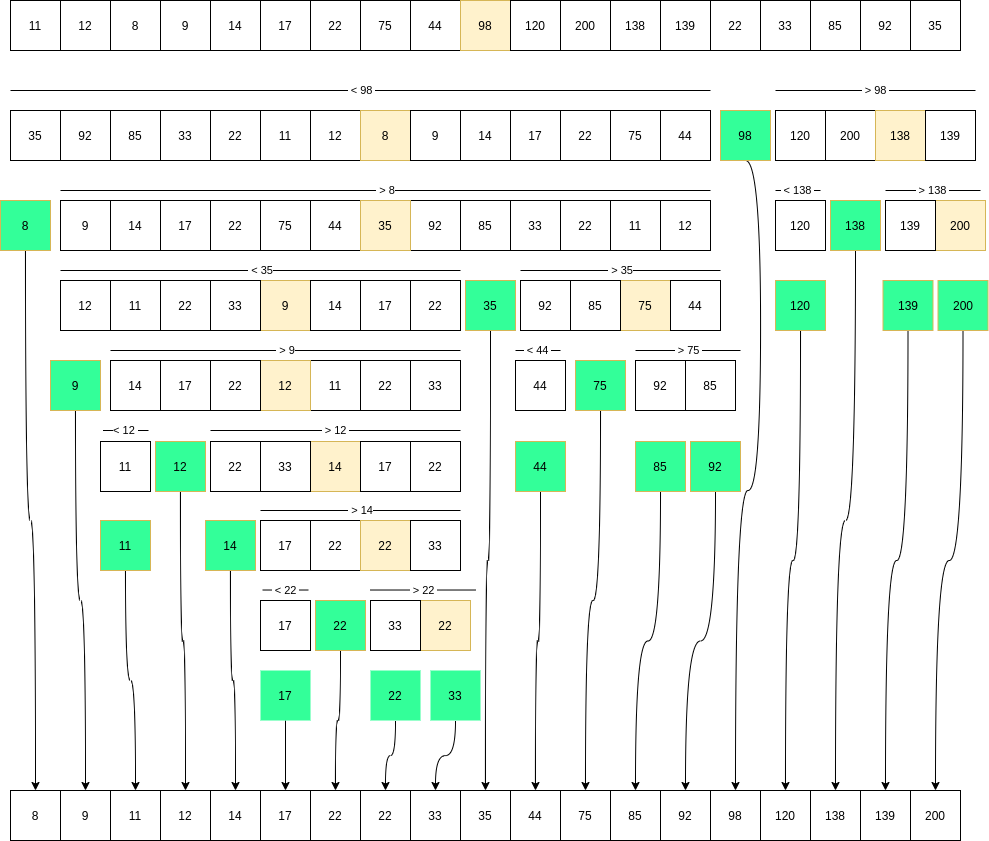
1. Estrutura de dados abstrata correspondente à um conjunto de objetos (vértices) que possuem relacionamentos (arestas) entre si. Além disso, grafos podem ser considerados direcionados (caminho entre duas vértices pode acontecer em apenas uma direção) e não direcionados (caminhos entre duas vértices pode acontecer nas duas direções)

### Simulações

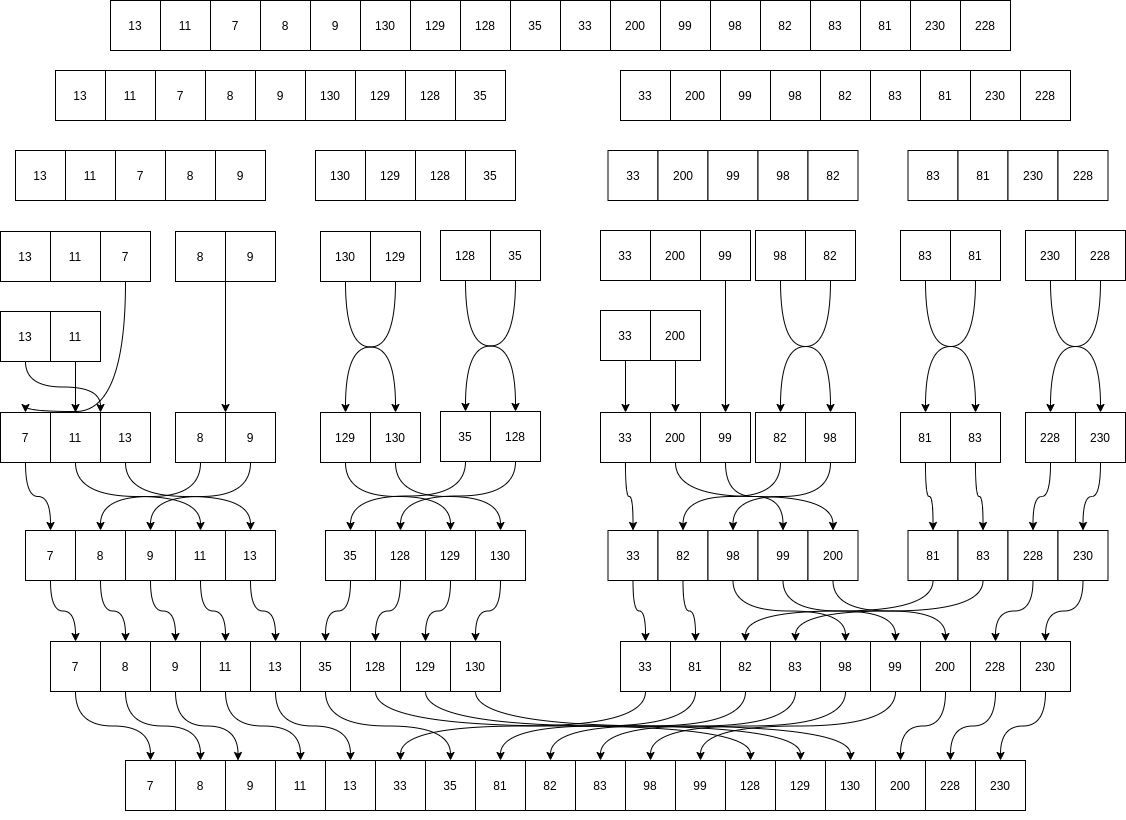
# Exercício 2

Conferir a pasta **exercicio2.**

# Exercício 3



# Exercício 4



# Exercício 5

## **Quick Sort:**

Algoritmo de ordenação que divide a lista em duas partes de forma contínua e move os items de menor valor para um lado e os outros, de maior valor, para outro. Essa comparação para que os itens sejam alocados de acordo com o seu valor é feita a partir de um valor de referência (pivô). Assim, os itens menores que o pivô são movidos para o lado esquerdo do mesmo, enquanto aqueles maiores ou iguais são organizados à direita. Após o primeiro passo, cada metade repete esse processo continuamente até restar apenas um item no grupo.

**Heap Sort:**

Algoritmo que utilizada a estrutura de dados denominada heap binário que corresponde a uma árvore binária organizada em um vetor. Cada elemento inserido na estrutura é imediatamente ordenado de modo crescente ou decrescente (dependerá da escolha por um max-heap ou um min-heap) e, após as inserções, os elementos são removidos de modo sucessivo da estrutura, mantendo-se a ordem desejada e a estrutura max-heap ou min-heap.

**Merge Sort:**

Baseado no conceito de divisão e conquita, este algoritmo segrega o vetor em duas partes de forma contínua até separar totalmente os elementos, uma vez que um veto de apenas um elemento é considerado ordenado. Assim, logo em seguida cada elemento é comparado com o seu adjacente e esses elementos são ordenados para formarem pares de elementos. Esse processo é feito de forma contínua, comparando os pares formados anteriormente com os seus pares adjacentes e combinando-os até que a extinção de todos os subconjuntos do vetor principal.

**Funcionamento dos métodos:**

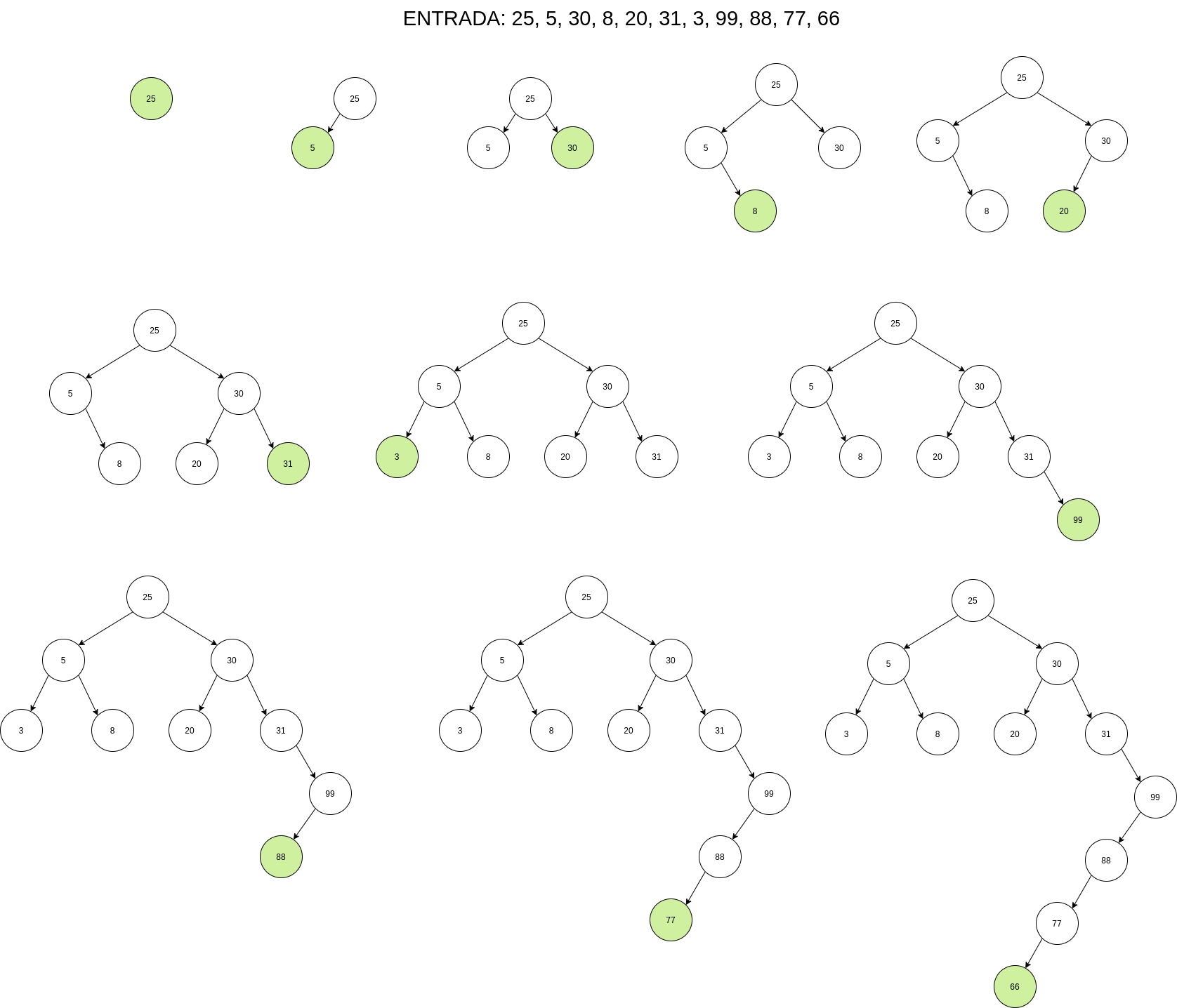
**ordenaA**: este método recursivo determina as posições de início e fim que indicam os locais para a divisão do vetor em duas metades de forma contínua até que o parâmetro início seja maior ou igual que o parâmetro fim. Após duas chamadas de si mesmo, este método passa as variáveis indicativas das posições do vetor para o **ordenaB.**

**ordenaB**: método para realizar as comparações dos items que estão nas posições indicadas (início, meio, meio+1 e fim) para verificar qual item é maior que outro. Feita essas comparações, os itens ordenados são combinados e alocados em um vetor auxiliar para que ao fim sejam organizados no vetor original.

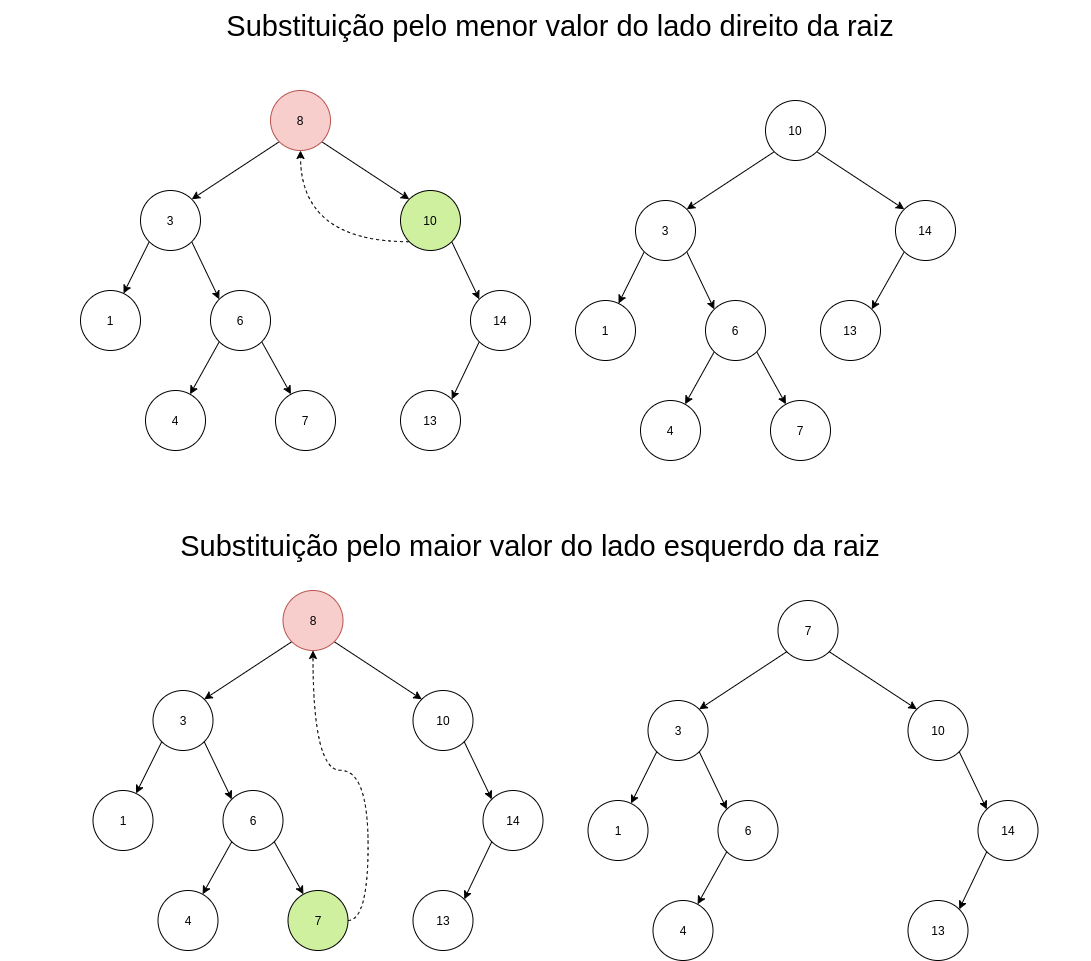
Dessa forma, considerando o funcionamento dos dois métodos descritos acima, o algoritmo de ordenação corresponde ao **Merge Sort**.

# Exercício 6

## **a)**



**b)**

****

**c)**

**Pré-ordem**

12, 4, 2, 8, 6, 16

**Ordem:**

2, 4, 6, 8, 12, 16

**Pós-ordem:**

2, 6, 8, 4, 16, 12

# Exercício 7

## ( **V** ) A disciplina de acesso da estrutura de dados Pilha determina que o último elemento inserido no conjunto deva ser o primeiro a ser removido.

**Justificativa**: A estrutura de dados Pilha utiliza o conceito de Último que entra Primeiro que sai (LIFO). Assim, o último elemento inserido na estrutura será o primeiro a ser retirado, ou seja, a alteração dessa estrutura é feita apenas em uma extremidade. Portanto, a afirmação é verdadeira.

( **F** ) A implementação de lista utilizando alocação sequencial dos elementos, comparada à alocação encadeada, necessita de mais espaço de armazenamento por elemento do conjunto.

**Justificativa**: Uma alocação encadeada corresponde a alocação da memória para a instância de um objeto. Esse objeto aponta, por meio de um endereço de memória, para o objeto seguinte alocado em outro local. Assim, uma alocação necessita de mais espaço na memória, pois os objetos estão alocados em diferentes espaços. Por outro lado, uma alocação sequencial ocupa o mesmo espaço de memória. Assim, a afirmativa está incorreta.

( **F** ) A pesquisa sequencial é mais eficiente que a pesquisa binária para busca de elementos em listas ordenadas implementadas com alocação sequencial dos elementos.

**Justificativa**: Uma pesquisa sequencial percorrerá cada elemento até que o elemento procurado seja encontrado. Para o pior caso (caso a procura seja para o último elemento), a pesquisa percorrerá todos os elementos da lista. Por outro lado, a pesquisa binária será mais eficiente, pois o elemento procurado será comparado com o elemento central da lista e verificará se o mesmo é maior, igual ou menor. Caso não seja igual, a comparação se repetirá para o elemento central das metades correspondentes (primeira metade caso o valor seja menor e a segunda metade caso o valor seja maior). Dessa forma, a afirmativa está incorreta, pois para uma lista ordenada uma pesquisa binária é mais eficiente.

( **V** ) As estruturas de dados Pilha e Fila podem ser implementadas utilizando tanto abordagens baseadas na alocação sequencial quanto na alocação encadeada dos elementos.

**Justificativa**: Como essas estruturas não exigem que os elementos guardem alguma relação com os outros elementos e, além disso, por se tratarem de estruturas lineares, tanto a alocação sequencial quanto à alocação encadeada podem ser utilizados para essas estruturas. Assim, a afirmação está correta.

( **F** ) A inserção de um elemento no início de uma lista duplamente encadeada implica no deslocamento dos elementos já existentes na memória.

**Justificativa**: A lista duplamente encadeada possui um atributo chamado head que aponta para o objeto inicial da lista. Esse objeto inicial aponta para o próximo elemento e é referenciado da mesma forma pelo seu sucessor e, caso um novo elemento seja adicionado, basta modificar as referências do objeto adicionado (aponta para o antigo primeiro elemento), e daquele que passará a ser o seu sucessor (este deverá referenciar como o seu anterior o elemento adicionado) e atualizar o head da estrutura. Dessa forma, a afirmação está incorreta, uma vez que não há necessidade de deslocar os elementos em memória.

**RESPOSTA CORRETA**:

**C) V F F V F**

# Exercício 8

**RESPOSTA CORRETA**:

**c)** Quando um novo elemento for inserido na lista é necessário que os ponteiros dos elementos envolvidos sejam atualizados, sem a necessidade de qualquer operação de deslocamento físico dos demais elementos da lista.