# Estruturas de dados

Dados:

|  |  |  |  |
| --- | --- | --- | --- |
| **Estrutura** | **Inserção\*** | **Listagem\*** | **Remoção\*** |
| Array | 5,1 | 4375,81 | 6,57 |
| Binary Tree | 594,89 | 30741,33 | 5,22 |
| Doubly Linked List | 4,64 | 4908,49 | 7,08 |
| Fila | 3,21 | 4980,62 | 7,05 |
| Grafo | 19,97 | 4774,94 | 13,28 |
| Linked List | 153,77 | 5700,57 | 5,4 |
| Pilha | 3,33 | 5234,66 | 14,55 |

\*em milissegundos

## Conclusões

**Array**

* **Melhor caso de uso:** Aplicações que exigem acesso rápido a elementos por índice.
* **Justificativa:** Tem excelente desempenho em listagem e bom desempenho em inserção e remoção, tornando-a ideal para situações em que o acesso sequencial ou por índice é prioritário.

**Binary Tree**

* **Melhor caso de uso:** Organização e busca hierárquica, como em bancos de dados ou algoritmos de pesquisa.
* **Justificativa:** Embora tenha desempenho relativamente lento em inserção e listagem, é eficiente em organizar dados e em buscas rápidas, como pesquisas balanceadas.

**Doubly Linked List**

* **Melhor caso de uso:** Situações que requerem inserções e remoções frequentes em qualquer posição.
* **Justificativa:** Tem bom desempenho em inserção e remoção devido à capacidade de manipular ponteiros de ambos os lados, tornando-a ideal para manipulações dinâmicas.

**Fila (Queue)**

* **Melhor caso de uso:** Implementação de estruturas FIFO (First In, First Out), como filas de tarefas ou buffers.
* **Justificativa:** Tem desempenho consistente em inserção, listagem e remoção, sendo eficaz para cenários de processamento sequencial.

**Grafo**

* **Melhor caso de uso:** Representação de relações complexas, como redes sociais ou mapas.
* **Justificativa:** Embora tenha um desempenho médio em inserção e remoção, é altamente eficaz para modelar conexões entre objetos.

**Linked List**

* **Melhor caso de uso:** Situações que exigem inserções frequentes e que não priorizam buscas rápidas.
* **Justificativa:** Embora mais lenta na inserção em comparação à doubly linked list, ainda é eficaz em casos onde o crescimento dinâmico é necessário.

**Pilha (Stack)**

* **Melhor caso de uso:** Estruturas LIFO (Last In, First Out), como chamadas de função ou cálculo de expressões.
* **Justificativa:** Tem excelente desempenho em inserção e remoção, tornando-a ideal para cenários de pilha.