**PARTE 3 : ANÁLISE E MELHORIAS**

ALUNO : Ernane Júnior Barreto Botelho

*LISTA ESTÁTICA X LISTA DIÂMICA*

Os testes de desempenho foram realizados em um programa Python que utiliza quatro tipos diferentes de listas: lista estática com inserção aleatória, lista estática com inserção ordenada, lista dinâmica com inserção aleatória e lista dinâmica com inserção ordenada. como cadastro de elementos, busca e remoção de elementos, com diferentes tamanhos de itens (1.000 e 100.000). Abaixo seguem os resultados obtidos:

1 - Lista estática com inserção aleatória: Para a inserção aleatória de elementos, a lista estática teve um tempo de registro de 22 segundos para 1.000 itens e 2.174 segundos para 100.000 itens. As operações de pesquisa e remoção tiveram média de 0,008125 segundos.

2 - Lista estática com inserção ordenada: Para inserção ordenada de elementos, a lista estática teve um tempo de registro de 17 segundos para 1.000 itens e 1.703 segundos para 100.000 itens. As operações de busca e remoção tiveram um tempo médio de execução de 0,008125 segundos.

3 - Lista dinâmica com inserção aleatória: Para a inserção aleatória de elementos, a lista dinâmica teve um tempo de registro de 17 segundos para 1.000 itens e 1.703 segundos para 100.000 itens. As operações de busca e remoção tiveram um tempo médio de execução de 0,0125 segundos.

4 - Lista dinâmica com inserção ordenada: Para a inserção ordenada de elementos, a lista dinâmica teve um tempo de registro de 16 segundos para 1.000 itens e 1.686 segundos para 100.000 itens As operações de busca e remoção tiveram um tempo médio de execução de 0,016875 segundos.

5 - Para a operação de cadastro de elementos, a lista dinâmica com inserção ordenada teve um tempo de execução para cadastro de pacientes ligeiramente menor em comparação às outras listas, sendo 16 segundos para 1.000 itens e 1.686 segundos para 100.000 itens. Isso acontece porque a estrutura da lista dinâmica para inserção ordenada permite inserções mais fáceis, onde um novo nó é inserido imediatamente antes ou depois do nó existente, dependendo da ordem. Já a lista estática com inserção aleatória apresentou o maior tempo de execução, devido à necessidade de realocar elementos para inserir novos itens em posições aleatórias.

6 - Para as operações de busca e remoção, as listas estáticas, tanto com inserção aleatória quanto ordenada, apresentaram tempos médios de execução semelhantes e menores (0,008125 segundos) em comparação às listas dinâmicas. Isso se deve ao fato de que as listas estáticas possuem acesso direto aos elementos por meio de índices, o que torna a busca e remoção mais rápidas. Por outro lado, as listas dinâmicas requerem a navegação pelos nós, o que aumenta o tempo de execução dessas operações.

7 - Os algoritmos de ordenação avaliados na tabela são Bubble Sort, Selection Sort e Insertion Sort. Para o cenário estático-aleatório, o Bubble Sort teve um tempo de execução de 7769 microssegundos, o Selection Sort teve um tempo de execução de 4087,38 microssegundos e o Insertion Sort teve um tempo de execução de 3062,67 microssegundos.

Para o cenário estático-ordenado, o Bubble Sort teve um tempo de execução de 2692 microssegundos, o Selection Sort teve um tempo de execução de 1179 microssegundos e o Insertion Sort teve um tempo de execução de 0,03 microssegundos.

Para o cenário dinâmico-aleatório, o Bubble Sort teve um tempo de execução de 6379,82 microssegundos, o Selection Sort teve um tempo de execução de 3317,54 microssegundos e o Insertion Sort teve um tempo de execução de 2541,98 microssegundos.

Para o cenário dinâmico-ordenado, o Bubble Sort teve um tempo de execução de 2509,83 microssegundos, o Selection Sort teve um tempo de execução de 1245,09 microssegundos e o Insertion Sort teve um tempo de execução de 0,03 microssegundos.

8 - Avaliando o cenário de testes dos algoritmos de ordenação , percebe-se que o algoritmo de Insertion Sortteve o melhor desempenho em todos os cenários, especialmente nos cenários ordenados, onde seu tempo de execução foi extremamente baixo (0,03 microssegundos). Isso ocorre porque o Insertion Sort é eficiente para listas parcialmente ordenadas, já que ele só faz trocas quando necessário.

Em contraste, o Bubble Sort apresentou o pior desempenho em todos os cenários, devido à sua natureza ineficiente de comparação e troca de elementos adjacentes. O Selection Sort teve um desempenho intermediário, sendo mais rápido que o Bubble Sort, mas ainda mais lento que o Insertion Sort. Isso acontece porque a estrutura do algoritmo Selection Sort busca o menor (ou maior) elemento da lista e o coloca na posição correta, reduzindo assim o número de trocas em comparação ao Bubble Sort. No entanto, ainda realiza várias comparações, o que o torna mais lento que o Insertion Sort. Já o insertion sort apresenta melhor desempenho, pois realiza menos comparações e trocas, especialmente em listas parcialmente ordenadas, onde ele apenas insere os elementos nas posições corretas conforme necessário.

Apesar da eficiência do Insertion Sort neste cenário, há algoritmos com maior eficiência Apesar da eficiência do Insertion Sort neste cenário, existem algoritmos que tendem a serem mais rápidos , como é o caso do Quicksort , que usa a abordagem de dividir para conquistar (dividindo o problema em subproblemas menores repetidamente) e tem uma complexidade média O(n log n), enquanto o Insertion Sort tem complexidade O(n^2). Portanto, em geral, para grandes conjuntos de dados desordenados, é provável que o Quicksort seja mais rápido que o Insertion Sort. Entretanto, para conjuntos de dados pequenos e quase ordenados, o Insertion Sort pode ser mais rápido que o Quicksort. Isso ocorre porque o Quicksort pode ter um custo adicional na etapa de divisão, enquanto que o Insertion Sort é uma escolha mais direta para lidar com dados já parcialmente ordenados.

Os resultados apresentados mostram que as listas estáticas com inserção aleatória e ordenada apresentam tempos de execução menores para as operações de busca e remoção em comparação com as listas dinâmicas, enquanto que para a operação de inserção aleatória, as listas estáticas apresentam tempos de execução maiores.

Já para a inserção ordenada, a lista dinâmica tem um desempenho melhor, pois permite inserções mais fáceis, sem a necessidade de realocar elementos. O tempo de execução das operações de busca e remoção da lista dinâmica com inserção ordenada é maior em relação às listas estáticas, devido à necessidade de percorrer os nós da lista para acessar os elementos.

Os testes realizados para avaliar os algoritmos de ordenação mostraram que o Insertion Sort apresentou o melhor desempenho em todos os cenários de teste, especialmente nos cenários ordenados, onde seu tempo de execução foi extremamente baixo.

Sendo assim, é recomendado o uso de listas estáticas com inserção aleatória ou ordenada para as operações de busca e remoção, e lista dinâmica com inserção ordenada para a operação de inserção ordenada. Para ordenação de elementos, o algoritmo de Insertion Sort pode ser uma boa escolha, especialmente para cenários em que os dados já estão ordenados.