Nome: Pedro Henrique Moreira Caixeta Ferreira

Exercício 01 - Revisão: tempo de execução de algoritmos

Preparação: Especificações do PC

Modelo: Galaxy Book2 360
Modelo do processador: Intel Core I5 1235U Evo
Velocidade: 4.40GHz
Quantidade de memória principal / RAM: 16GB 4267MHz
Sistema Operacional: Windows 11 Home

TESTES

ArrayList

P = 2.500.000, N = 20.000 - 4106,83 ms ou 4,1068 seg

P = 5.000.000, N = 20.000 - 5683,81 ms ou 5,6838 seg

P = 10.000.000, N = 20.000 - 8161,94 ms ou 8,1619 seg

Considerações: A complexidade do algoritmo com a busca em arraylist é de O(n)

Em comparação com os testes anteriores do HashMap, esses resultados sugerem que o desempenho do ArrayList é pior do que o desempenho do HashMap em termos de busca aleatória.

Esses resultados sugerem que o tempo de execução aumenta linearmente com o aumento de P, o que é consistente com uma complexidade de tempo O(P) para a operação de inserção.

Quanto maior o número de pessoas, mais memória o sistema utiliza.

Quando se compara os resultados com as complexidades teóricas o desempenho do ArrayList para buscas aleatórias é mais próximo de uma complexidade linear (O(n)), enquanto o HashMap mantém uma complexidade constante (O(1)), como esperado. Isso destaca a importância de escolher a estrutura de dados certa para o problema específico que está sendo resolvido.

Teste Final

P = 2.500.000, N = 40.000 - 6124,12 ms ou 6,1241 seg

Considerações: A criação dos dados, feita através da inserção de pessoas em um ArrayList, tem uma complexidade linear (O(P)), onde P é o número de pessoas.

A busca ArrayList pode ser ineficiente para buscas aleatórias em grandes conjuntos de dados devido à possibilidade de recuperações lineares.

Comparado aos testes anteriores com um número de busca de 20000, o aumento para 40.000 buscas resultaram em um aumento perceptível no tempo de execução. Isso sugere que a eficiência a busca aleatória se torna mais desafiadora à medida que o número de buscas aumenta.

Concluindo, o resultado destaca as limitações do ArrayList para buscas aleatórias eficientes em grandes conjuntos de dados, enfatizando a importância de escolher a estrutura de dados adequada para as operações pretendidas e que possam utilizar menos memória.

TabelaHash

P = 2.500.000, N = 20.000 - 5292,39 ms ou 5,2924 seg

P = 5.000.000, N = 20.000 - 5490,33 ms ou 5,4903 seg

P = 10.000.000, N = 20.000 - 9918,55 ms ou 9,9185 seg

Considerações: Este algoritmo tem uma complexidade de tempo de O(1) que pode ser O(N) também.

A tabela hash pode sofrer colisões, e a complexidade pode aumentar se muitas colisões ocorrerem, exigindo uma busca linear na lista de colisões.

Sempre quando aumenta o P a probabilidade de colisões também é maior, impactando negativamente o desempenho.

À medida que o número de elementos aumenta, a tabela hash pode precisa ser redimensionada, o que é uma operação custosa.

O aumento de P pode levar a um maior carregamento da tabela, afetando o desempenho geral.

Quanto maior o número de pessoas, mais memória o sistema utiliza.

Teste Final

P = 2.500.000, N = 40.000 - 7429,69 ms ou 7,4297 seg

Considerações: Aumentar o número de buscas (N) resultou em um tempo de execução maior. Ouvisse também aumento de colisões podendo levar a uma busca mais lenta. Com um maior número de elementos, é mais provável que a tabela hash precise ser redimensionada. Podendo concluir também com mais elementos, a carga da tabela hash aumenta. Isso pode levar a uma menor eficiência na busca, pois a probabilidade de colisões aumenta.