EXERCÍCIO 4

Foram feitas 10 medidas para o tempo de processamento das atividades solicitadas e os resultados estão apresentados nas tabelas abaixo.

Um dado importante é que, no lugar de inserir 10^8 dados nos vetores e acessar 10^7, foram inseridos 10^6 e acessados 10^5 dados. Essas modificações ocorreram devido à capacidade limitada de processamento de dados do computador no qual os testes foram feitos. Se seguíssemos as instruções iniciais do exercício, seria inviável fazer várias medições, pois demoraria muito tempo.

ArrayList			
Inserção e acesso		Remoção e inserção	
ms	s	ms	s
146	0,146	7086	7,086
166	0,166	6890	6,89
142	0,142	7270	7,27
130	0,13	6846	6,846
101	0,101	6911	6,911
114	0,114	7140	7,14
141	0,141	7042	7,042
79	0,079	4623	4,623
78	0,078	4624	4,624
78	0,078	4688	4,688
Média		Média	
117,5	0,1175	6312	6,312
Desvio Padrão		Desvio Padrão	
30,59493422	0,030594934	1097,853	1,097853

LinkedList			
Inserção e acesso		Remoção e inserção	
ms	min	ms	min
196005	3,26675	206340	3,439
192341	3,205683333	207847	3,464117
190775	3,179583333	206295	3,43825
193623	3,22705	206669	3,444483
191577	3,19295	203475	3,39125
192296	3,204933333	206459	3,440983
191512	3,191866667	202520	3,375333
151022	2,517033333	164947	2,749117
146217	2,43695	159298	2,654967
144383	2,406383333	159555	2,65925
Média		Média	
178975,1	2,982918333	192340,5	3,205675
Desvio Padrão		Desvio Padrão	
20897,31	0,348288427	20446,28	0,340771

Das tabelas, percebemos que o tempo gasto para inserir 10^6 elementos em um vetor/lista inicialmente vazio e posteriormente acessar 10^5 elementos (por meio de índices aleatórios) foi, em média:

- 0,118 s para o ArrayList
- 2,982 min para a LinkedList

Após realizar alguns testes, constatamos que essa diferença não se deve ao tempo gasto pela ArrayList ou pela LinkedList para preencher o vetor/lista, sendo esses processos extremamente otimizados. Na realidade, a maior diferença está no tempo de busca/acesso a determinado elemento de índice aleatório i. A explicação para a diferença se baseia no fato de que buscamos um elemento a partir do seu índice i, sendo que por meio de um vetor/array (ArrayList) o tempo de acesso ao elemento é praticamente nulo - $O(1)$ -, entretanto, em uma lista ligada (LinkedList) não temos acesso direto a um elemento de índice i (podemos acessar o elemento somente através de "ponteiros"), sendo necessário percorrer toda a lista, no pior caso, para encontrar o elemento buscado - $O(n)$.

O tempo gasto para remoção de 10^5 elementos de índices aleatórios e posterior inserção de outros 10^5 elementos foi, em média:

- 6,312 s para o ArrayList
- 3,206 min para a LinkedList

Pelos mesmos motivos apresentados anteriormente, essa diferença não se deu pelo tempo gasto para inserir novos elementos no vetor/lista, mas, sim, pelo tempo gasto para a remoção de um elemento de índice i aleatório. Seguindo o mesmo raciocínio apresentado anteriormente, buscamos um elemento de índice i para removê-lo. Porém, a busca por um elemento de um vetor/array (ArrayList) por meio de um índice é praticamente instantânea, enquanto para uma lista encadeada é necessário percorrer toda a lista por meio de ponteiros até encontrar o elemento de índice i.

Há, entretanto, um detalhe importante a ser mencionado. Se buscássemos um elemento de um vetor/lista por meio de seu conteúdo e não por meio de seu índice, a LinkedList teria grandes vantagens sobre a ArrayList. Isso porque, nesse caso, para ambas as estruturas de dados precisaríamos percorrer todo o vetor em busca do elemento de conteúdo especificado para, em seguida, removê-lo. Porém, a remoção em um ArrayList consiste, conforme indica a documentação, em "deslocar quaisquer elementos subsequentes para a esquerda (subtraindo 1 de seus índices)", o que é um processo custoso. Para a LinkedList o processo é mais simples, consistindo em mudar o nodo para qual o ponteiro do elemento anterior ao que está sendo removido aponta (fazemos que o ponteiro do elemento anterior aponte para o elemento posterior ao nodo atual), o que é um processo bem mais rápido.