1.Configurações do computador:

Processador: AMD Ryzen 5 5500 3.80 GHz 6 núcleos e 12 processadores lógicos
Memória: XPG 16 RAM 3200 MHz dual channel
Sistema Operacional: 64 bits, Windows 10 pro

OBS: Meu sistema é modificado para não ficar rodando processos em background e economizar memória/processador

2.Casos de teste:

ArrayList Stream

	Marcações P e N	Tempo
1	P = 2.500.000, N = 20.000	59,1813 s
2	P = 5.000.000, N = 20.000	118,0290 s
3	P = 10.000.000, N = 20.000	236,5028 s
4	P = 2.500.000 e N = 40.000	118,1184 s

ArrayList Loop

	Marcações P e N	Tempo
1	P = 2.500.000, N = 20.000	33,2810 s
2	P = 5.000.000, N = 20.000	66,2419 s
3	P = 10.000.000, N = 20.000	128,1505 s
4	P = 2.500.000 e N = 40.000	64,3920 s

HashMap Stream

	Marcações P e N	Tempo
1	P = 2.500.000, N = 20.000	0,1557 s
2	P = 5.000.000, N = 20.000	0,2370 s
3	P = 10.000.000, N = 20.000	0,4866 s
4	P = 2.500.000 e N = 40.000	0,1689 s

HashMap Loop

	Marcações P e N	Tempo
1	P = 2.500.000, N = 20.000	0,1551 s
2	P = 5.000.000, N = 20.000	0,2772 s
3	P = 10.000.000, N = 20.000	0,4906 s
4	P = 2.500.000 e N = 40.000	0,1510 s

3.Considerações casos de teste:

ArrayList Stream e Loop

Acesso: O(1);
Busca: O(N);
Inserção/Delegação no final: O(1) amortizado; O(N) no pior caso(redimensionar array);

Análise dos Resultados:

Observa-se que o tempo de execução dobra quando o tamanho do ArrayList (P) dobra, o que é esperado, dado que a busca tem complexidade O(N). Isso é evidente tanto para a versão com loop quanto para a versão com stream.

A versão com loop é consistentemente mais rápida que a versão com stream, o que pode ser atribuído ao overhead adicional de abstração e às chamadas de métodos indiretas introduzidas pelos streams.

HashMap Stream e Loop

Acesso/Busca/Inserção/Delegação: O(1) no médio; O(N) no pior caso;

Análise dos Resultados:

Os tempos de execução para operações em HashMap são significativamente menores que para ArrayList, o que está alinhado com a complexidade teórica esperada de O(1) para operações típicas, assumindo uma boa função de hash e distribuição de chaves.

Aumentar o número de operações (N) têm um impacto menor no tempo de execução em comparação com o ArrayList, o que novamente é consistente com a complexidade de tempo constante esperada.

Não há diferença significativa entre a execução com loop e com stream para HashMap, sugerindo que o overhead de abstrações do stream é minimizado pela eficiência geral do HashMap.

Fontes:

- https://pt.stackoverflow.com/questions/361945/por-que-existe-tanta-diferen%C3%A7a-de-performance-entre-stream-e-loops-normais
- https://medium.com/infobipdev/slow-like-a-stream-fast-like-a-loop-524f70391182
- https://pt.stackoverflow.com/questions/56836/defini%c3%a7%c3%a3o-da-nota%c3%a7%c3%a3o-big-o/56868#56868
- https://cursos.alura.com.br/forum/topico-velocidade-da-busca-104920#:~:text=HashSet%20possui%20uma%20complexidade%20de,depende%20do%20n%C3%BAmero%20de%20elementos.
- https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html
- https://docs.oracle.com/javase/8/docs/api/java/util/HashMap.html