Nome: Gonçalo Eduardo Costa dos Santos Henriques

Número: 123422

Turma: CDA2

Informações relevantes a referir

É importante salientar que foram feitas algumas alterações que considerei relevantes fazer ao template original fornecido deste projeto. Nomeadamente, gostaria de reforçar a minha ideologia para a implementação da classe Target, pois sei que apenas foi solicitado (para uma avaliação até 20 valores) a implementação de uma Fila do tipo Numpy para guardar os respetivos tuples com o número do batch processado, o mínimo e máximo encontrado nesse batch, o que seria necessário para este projeto. Contudo, fui além desse critério e nessa mesma classe Target implementei um tipo de fila circular que permitiria uma "reutilização" do espaço de elementos já retirados da fila. Também gostaria de reforçar o porquê de não ter usufruído da função "fetch_value". Este critério foi tomado devido ao que é retornado das funções de ordenação encontradas na classe Main; se a respetiva função (selection_sort, bubble_sort ou merge_sort) ordenasse a lista por ordem crescente retirava o elemento do 1º índice como mínimo e o elemento do último índice como máximo, caso ordenasse por ordem decrescente, faria o inverso. Por último, o código encontra-se todo devidamente comentado explicitamente para uma melhor compreensão.

Tempo de execução do algoritmo em função do tamanho do *batch*

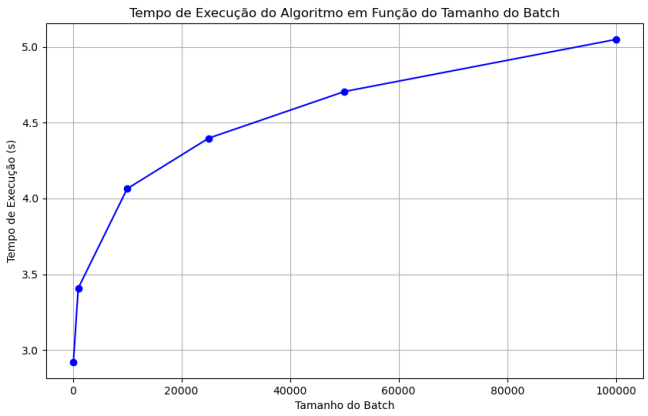

Fig.1 – Gráfico ***Merge Sort***

A partir da análise do gráfico, é notório que o tempo de execução do ***merge sort*** (algoritmo usado) cresce com o aumento do tamanho do batch (100, 1000, 10000, 25000, 50000 e 100000) refletindo a sua complexidade O(n log n). O crescimento é acentuado para batches de "pequena" a "média" dimensão, mas "abranda" para os batches maiores, evidenciando que este algoritmo lida mais eficientemente com volume maior de dados.

Tempo de execução do algoritmo em função do algoritmo de ordenação

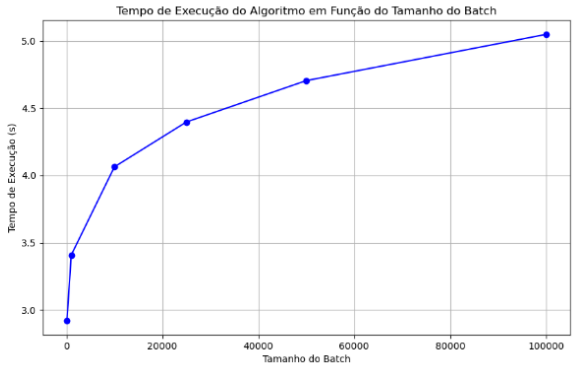

Fig.2 – Gráfico ***Merge Sort***

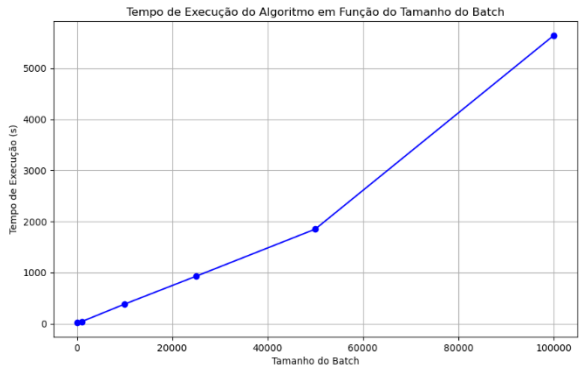

Fig.3 – Gráfico ***Selection Sort***

Analisando os gráficos, o primeiro, que representa o ***merge sort***, mostra um aumento progressivo e suave do tempo de execução, que cresce mais rapidamente para lotes pequenos e tende-se a estabilizar para batches maiores. Já o segundo gráfico, representando o ***selection sort***, exibe uma tendência clara de crescimento linear, onde o tempo de execução aumenta de maneira mais uniforme e íngreme com o tamanho do batch. Isso evidencia a eficiência relativa do merge sort para lidar com grandes quantidades de dados, em contraste com o selection sort, que evidencia escalar o tempo de execução de forma diretamente proporcional ao tamanho do batch.