Relatório *- Ordenação- CI1056*

Universidade federal do Paraná - UFPR

Bruno Corrado Crestani GRR20221240

bruno.crestani@ufpr.br

1. INTRODUÇÃO

Este relatório tem por objeto dissertar sobre o que é possível concluir de acordo com a coleta de dados obtida no trabalho de ordenação sobre os sete algoritmos estudados até o momento na disciplina de Algoritmos e Estrutura de Dados II, sendo dois deles de busca em vetores, e os outros cinco de ordenação.

2. MÉTODOS DE AVALIAÇÃO

Conforme especificado no descritivo do trabalho, foram avaliados principalmente os tempos de duração de execução dos algoritmos e o número de comparações entre elementos do vetor realizados.

3. ALGORITMOS DE BUSCA

De imediato é possível notar uma gigantesca discrepância entre os algoritmos de busca propostos. O Busca Sequencial é possivelmente um dos mais óbvios e primitivos de implementar quando se pensa em busca em vetores, iniciando a procura pelo último elemento do vetor, e conferindo um a um até que o elemento desejado seja encontrado, e sua relação entre o número de comparações e tamanho do vetor pode ser resolvida com uma simples conta de subtração depois de descobrir o índice do elemento do vetor procurado.

Já o Busca Binária, apesar de possuir uma implementação simples e só funcionar

em vetores já ordenados, possui um número de comparações impressionante para vetores de todos os tamanhos, atingindo 27 comparações para um vetor ordenado de 50.000 elementos, por exemplo.

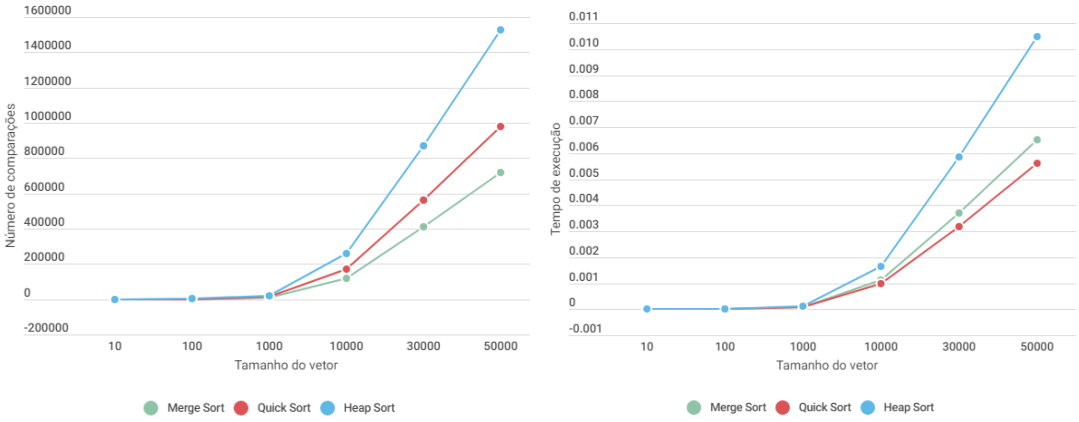
4. ALGORITMOS DE ORDENAÇÃO

INSERTION E SELECTION SORT

Dois Algoritmos básicos de ordenação, que conforme citado pelo professor em sala, possuem qualidades e defeitos opostos, sendo o Insertion Sort “bom de comparação e ruim de troca”, e o Selection Sort “ruim de comparação e bom troca”. Ambos os algoritmos possuem seu pior caso em torno de n^2 (sendo n o número total de elementos) e seus tempos de execução também aumentam de forma quadrática em relação ao aumento do tamanho do vetor.

MERGE, QUICK E HEAP SORT

Devido ao custo desses algoritmos se comportar de maneira próxima a n log2(n), são os algoritmos mais eficientes e complexos da lista.

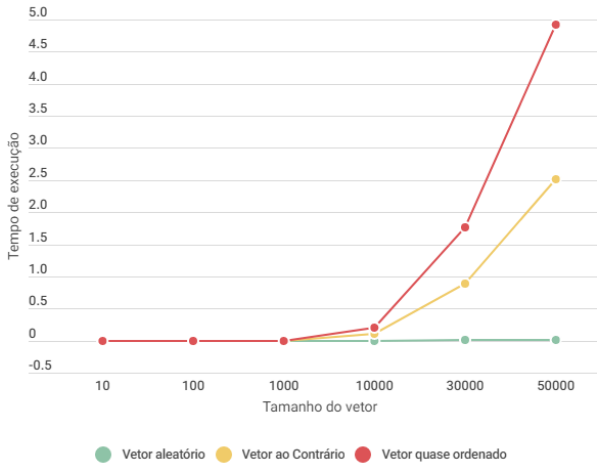


É possível notar que o Heap sort é um pouco mais custoso e lento que os demais, mas já vimos que a estruturação de uma Heap é diferente e pode ser muito melhor aproveitada em construções de fila com prioridades, por exemplo.

O Merge Sort é o algoritmo que se sobressai nos gráficos mostrados, pois ele age em cima do conceito de dividir o vetor maior em inúmeros vetores de apenas um elemento, e reordena-los em um grande vetor novamente. Seu ponto negativo para vetores muito grandes é o alto uso de memória.

O Quick Sort pode ser considerado o algoritmo de ordenação do dia a dia, é extremamente eficiente para casos mais aleatórios, mas não é o recomendado para situações específicas e que necessitem de um elevado nível de precisão e tempo de reação.

O Quick Sort tem problemas com vetores quase ordenados ou completamente desordenados, por exemplo.



5. CONCLUSÃO

Existem diversos tipos e nichos de algoritmos, da mesma maneira que existem inúmeras ocasiões em que ordenações são requisitadas, e junto com elas vem diversas variáveis que serão determinantes para decidir qual será o melhor algoritmo a se utilizar, como tamanho do vetor, nível de ordenação do vetor, número de comparações desejado, capacidade do hardware e diversos outros

Dessa maneira, é necessário dominar todos, e entender a funcionalidade de cada um.