FACULDADE FEDERAL DE ALFENAS - UNIFAL

CIÊNCIA DA COMPUTAÇÃO

LEONARDO BONARDI MARQUES SILVA E VINICIUS HENRIQUE PIOTTO BOIAGO

RA: 2023.1.08.011 E 2023.1.08.024

RELATÓRIO ENTRE OS MÉTODOS DE ORDENAÇÃO

ALFENAS

2023

FORMA DE ORDENAÇÃO E SEUS RESULTADOS.

Introdução:

Em computação existem vários tipos de ordenação de vetores com diferentes desempenhos e aplicações. Diante disso, selecionamos alguns para estudarmos e decidirmos qual é o mais eficiente de acordo com os resultados obtidos. Como foi dito antes, na computação existem vários meios de ordenação, mas como estudantes dessa área, aplicamos e analisamos alguns desses métodos, a fim de determinarmos como cada um dos meios se comportam e qual deve ser escolhido em relação ao seu objetivo.

Referencial teórico:

Em computação, é fundamental saber sobre os métodos de ordenação e seu funcionamento, pois em uma aplicação, se utilizado do meio correto, o projeto se torna adequado e bem implementado, facilitando o funcionamento do software, além de gastar menos tempo na execução. Entre os métodos existentes, várias formas de ordenação são implementadas, podendo escolher o valor do vetor de forma aleatória ou escolher a menor e encaixá-las corretamente, por exemplo, como na Selection Sort e Bubble Sort.

Sobre o Selection Sort, é um modelo de ordenação que seleciona o menor valor dos vetores e o aloca na primeira array e depois disso, procura o segundo menor valor, o alocando na segunda array e assim por diante, como descreveu João Arthur Brunet da UFCG. Já o Bubble Sort, funciona como um comparador em pares, em que o método analisa os dois primeiros termos, verificando se precisam ser trocados de posição de acordo com o que o algoritimo deseja, se é de forma crescente ou decrescente. Dessa forma, o próximo passo é analisar o segundo termo com o terceiro, verificando novamente se é necessário a troca ou não e assim sucessivamente. Depois de ter analisado todos os termos, o Bubble Sort terá que ser rodado mais uma vez, para certificar que nenhum termo não foi analisado, de acordo com Giulianna Seabra do site betrybe. E por último, o Insert Sort, que funciona como uma rotina de comparação para o alocamento adequado. Para tal, o algoritimo seleciona de forma crescente em ordem os valores do vetores, comparando com as ordens anteriores, de acordo

com o objetivo do algoritimo. Esse valor se encaixaria de forma correta no vetor, realizando esse procedimento diversas vezes, sendo essa explicação formulada mediante ao artigo do site programiz.

De acordo com o tema, esse referencial tem como foco ampliar e clarear para alguns o porquê de ser necessário o estudo sobre as ordenações e seus meios de funcionamento, pois na internet é de extrema dificuldade de encontrar sites ou artigos que comentam de forma clara e objetiva a diferença entre esses meios, em um linguagem simples e de fácil compreensão para o público geral, visto que a linguagem utilizada nesses meios de publicação são focados para o público que já tiveram contato com a programação. Portanto, é importante que essa ponte entre a compreensão para todos do mundo da computação seja realizada, fazendo assim, com que mais pessoas estejam cientes sobre o entendimento da lógica digital.

Por conseguinte, para que todos entendam os diversos métodos e qual é o mais eficiente, uma boa prática é realizar esses métodos de forma manual, para então, entende-los. Nesse caso, será necessário apenas de um lápis ou caneta e um papel, e então, gerar n números aleatórios e aplicar os métodos. Dessa maneira, será possível entender e verificar sobre a eficiência desses meios.

Material utilizado:

Para conseguirmos realizar esse trabalho, utilizamos de algumas fontes como o livros *Como programar em C++*, 5ªEdição e *C Completo e Total*, Terceira edição. Além disso, utilizamos como auxilio de nosso projeto o site https://www.mygreatlearning.com/blog/sort-function-in-cpp/, ambos que nós guiaram para a construção de nossas articulações.

Métodos implementados:

Nesse segundo projeto, como métodos de ordenação utilizamos o Bubble sort, Insert Sort e o Selection Sort.

Resultados obtidos:

Depois de termos implementados os métodos, obtivemos os seguintes resultados: Em vetores com 100 elementos - Insert Sort: forma crescente - 594

repetições, forma aleatória - 2998 repetições, forma decrescente: 5544 repetições. Selection Sort: forma crescente - 99 repetições, forma aleatória - 411 repetições, forma descrescente - 2599 repetições. Bubble Sort: forma crescente - 9801 repetições, forma aleatória - 12205 repetições, forma decrescente - 14751 repetições. Em vetores com 1000 elementos - Insert Sort: forma crescente - 5994 repetições, forma aleatória - 259255 repetições, forma decrescente: 505494 repetições. Selection Sort: forma crescente - 999 repetições, forma aleatória - 6475 repetições, forma descrescente - 250999 repetições. Bubble Sort: forma crescente - 998001 repetições, forma aleatória - 1251262 repetições, forma decrescente - 1497501 repetições. Em vetores com 10000 elementos - Insert Sort: forma crescente - 59994 repetições, forma aleatória - 25175502 repetições, forma decrescente: 50054994 repetições. Selection Sort: forma crescente - 9999 repetições, forma aleatória - 87165 repetições, forma descrescente - 25009999 repetições. Bubble Sort: forma crescente - 99980001 repetições, forma aleatória - 125095509 repetições, forma decrescente - 149975001 repetições.

Conclusão:

Em relação ao segundo método com eficiência mediana, temos o Insertion Sort. Este método apresentou resultados favoráveis quando o vetor estava na forma crescente, demonstrando uma boa performance em relação ao número de leituras e operações realizadas no vetor. No entanto, quando o vetor estava aleatório ou decrescente, o Insertion Sort teve um desempenho inferior ao Selection Sort.

Por último, o Bubble Sort mostrou-se o método menos eficiente entre os três em todos os testes realizados. Ele apresentou um desempenho insatisfatório quando comparado ao número de leituras e operações realizadas no vetor. No entanto, é importante ressaltar que, em comparações de trocas, o Bubble Sort poderia desempenhar-se bem apenas quando o vetor já estivesse ordenado de forma crescente, pois não haveria necessidade de realizar trocas.

Considerando todos os aspectos analisados, o Selection Sort demonstrou ser o método mais eficiente e recomendado para a maioria das situações, especialmente em arranjos aleatórios e decrescentes. O Insertion Sort apresentou um desempenho mediano e pode ser considerado uma opção viável para arranjos já ordenados de forma crescente. Por fim, o Bubble Sort não se mostrou eficiente em nenhuma situação analisada neste projeto.

É importante destacar que a escolha do método de ordenação mais adequado dependerá das características específicas do conjunto de dados, como tamanho do vetor, estado inicial e contexto de aplicação. Os resultados e conclusões obtidos neste projeto servem como base para orientar a seleção do método de ordenação mais eficiente em diferentes cenários. No entanto, é recomendado realizar testes adicionais e considerar outros fatores relevantes antes de tomar uma decisão final.