INSTITUTO FEDERAL SUL-RIO-GRANDENSE - câmpus Passo Fundo

Entrega Parcial Trabalho Impostor

Estudante: Artur Mariano da Silva
Período: 5M1
Disciplina: Estrutura de Dados III
Professor: Élder Francisco Fontana Bernardi

Relatório

Até o presente momento, foi analisado o cenário em que devemos estar imersos, o qual consiste em um planeta que cometeu o delito de sabotar uma nave Sborniana. Para isso, agentes do serviço secreto devem se infiltrar no planeta X9 e realizar uma operação a fim de encontrar o impostor que cometeu tal crime. Para isso, eles devem resgatar os logs da nave, os quais foram embaralhados pelo inteligente criminoso, e posteriormente ordená-los para encontrar o culpado.

Nesse cenário, considerando a solução primária proposta pelos agentes sem muitos conhecimentos tecnológicos e de otimização de algoritmos, o resultado não foi o esperado, o algoritmo levou tempo suficiente para eles serem descobertos. Sendo assim, devemos buscar uma alternativa que utilize menos recursos e que resolva o problema mais rapidamente, evitando a descoberta deles por parte dos alienígenas.

Para isso, foram realizadas pesquisas a respeito dos algoritmos de ordenação sem o uso de comparações. Com isso, surgiram duas opções: Radix Sort e Counting Sort. A respeito do primeiro e de acordo com Cormen et al. (2009), o RadixSort é um algoritmo de ordenação rápido e estável que pode ser usado para ordenar itens que estão identificados por chaves únicas e que cada chave é uma cadeia de caracteres ou número. Nos computadores, estas chaves são os números binários usados para representar todo caractere a partir de um conjunto de dados binários.

Já o Counting Sort é um algoritmo que, como o próprio nome diz, baseia-se em contar os elementos. Ele cria outro vetor com o tamanho do maior valor entre aqueles sobre os quais se realiza a ordenação. Sendo assim, ele se torna mais eficiente em situações que se tem conhecimento dos dados de entrada, podendo ter seu desempenho piorado significativamente ao inserir apenas um valor gigantesco.

Há ainda o bucket sort, porém ele também depende da entrada para ter a sua eficiência garantida, tornando essa solução pouco inteligente.

Assim sendo, percebe-se que o Radix Sort deve ser o algoritmo mais adequado para solucionar o problema dos Sbornianos.

Para realizar essa comprovação, estou fazendo testes primeiramente utilizando o Selection Sort e posteriormente comparando com o Radix Sort e o Counting Sort. Vale ressaltar que o teste com o Selection Sort já foi feito, e após dez minutos encerrei o processo, pois ele ainda não havia ordenado os dois milhões de logs.

Agora, estou na parte da implementação do Radix e do Counting Sort, para assim realizar os testes e comparar o desempenho.

Por fim, dei início à escrita do artigo, escrevi parte da introdução - em que contextualiza o leitor da situação fictícia em que fomos inseridos - e dei início às soluções propostas, para depois servir de base para a escrita do restante do artigo.

Referências Bibliográficas

https://joaoarthurbm.github.io/eda/posts/selection-sort/

https://www.geeksforgeeks.org/selection-sort/

https://www.geeksforgeeks.org/counting-sort/

https://home.unicruz.edu.br/seminario/anais/anais-2017/XXII%20SEMIN%C3%81RIO%20INTERINSTITUCIONAL%202017%20-%20ANAIS/GRADUA%C3%87%C3%83O%20-%20TRABALHOS%20COMPLETOS_CI%C3%8ANCIAS%20EXATAS,%20AGR%C3%81RIAS%20E%20ENGENHARIAS/M%C3%89TODO%20DE%20ORDENA%C3%87%C3%83O%20DE%20DADOS%20RADIXSORT.pdf

Ricarte L. M. I. Ordenação por contagem. Disponível em: http://www.dca.fee.unicamp.br/cursos/EA876/apostila/HTML/node24.html. Acessado em: 20 Ago. 2016, 2003.

OBS.: ainda não estão todas no formato pedido, para o artigo irei formatar corretamente.