UNIVERSIDADE FEDERAL DO AMAZONAS
RELATÓRIO FINAL PIBIC/PAIC 2015-2016

UFAM

FORMULÁRIO PARA RELATÓRIO FINAL

1. Identificação do Projeto

Título do Projeto PIBIC/PAIC

Análise dos Algoritmos de Ordenação: buscando a eficiência em sistemas de supermercados

Orientador

Professor Doutor Jorge Yoshio Kanda

Aluno

Romualdo Costa de Azevedo

2. Informações de Acesso ao Documento

2.1 Este documento é confidencial?

[ ] **SIM** [x] **NÃO**

2.2 Este trabalho ocasionará registro de patente?

[ ] **SIM** [x] **NÃO**

2.3 Este trabalho pode ser liberado para reprodução?

[ ] **SIM** [x] **NÃO**

2. 4 Em caso de liberação parcial, quais dados podem ser liberados? Especifique.

3. Introdução

Algoritmo é um conjunto de instruções que mostra como uma sequência de ações deverá ser executada (ZIVIANI, 1999). Para um sistema computacional ter pleno funcionamento é necessário, que internamente, sejam executadas as instruções conforme as operações desejadas.

Os usuários de um sistema computacional geralmente esperam que o sistema forneça as informações no menor tempo possível. Essa eficiência está diretamente relacionada à qualidade do algoritmo acionado para executar a operação inerente à solicitação do usuário (MEDINA, 2005).

O tempo de busca por uma informação em um conjunto de dados depende de como os dados estão organizados. Se os mesmos estiverem ordenados, haverá maior rapidez no

processo de consulta. Segundo Cormem (1999), para ordenar os dados podem ser utilizados diversos tipos de algoritmos de ordenação, tais como: Merge-Sort, Quick-Sort, Insertion-Sort, Selection-Sort, e Booble-Sort.

A ordenação de elementos é algo essencial em diversos aspectos da sociedade, principalmente para as empresas. Se os dados já estiverem ordenados, mais rápido será o tempo de geração da informação para o usuário. Atualmente, a maioria dos supermercados é gerenciada por meio de sistemas computacionais. E quanto mais rápido for o atendimento ao cliente, maior será a satisfação do mesmo para com o estabelecimento.

Além dos algoritmos de ordenação, realizou-se um estudo detalhado sobre as principais técnicas de busca, que são: a busca sequencial e a busca binária. Segundo Ziviani (2007) a busca sequencial, é preferencialmente aplicada em um conjunto de dados não ordenados, pois pode ser necessária a realização de uma busca por todo o arquivo. Segundo Júnior (2008) a busca binária pode ser utilizada se os dados já estiverem previamente ordenados, porém o tempo de processamento da procura pelo elemento é significativamente menor porque a cada iteração metade dos dados são descartados

4. Justificativa

A área de Projeto e Análise de Algoritmos dispõe de poucas referências na área da computação. Este trabalho visa ajudar os iniciantes nesta área, tornando a comparação dos algoritmos de ordenação, um exemplo crucial para os pesquisadores da área, onde será possível verificar como as análises dos algoritmos podem ser úteis.

Um estudo desse gênero é útil para obter a eficiência de sistemas que controlem estoques de supermercados, usando os Algoritmos de Ordenação e verificando como os mesmos influenciam em um sistema desse porte. Usando as técnicas de Análise de Algoritmos, foram analisados os Algoritmos de Merge, Insertion, Quick, Booble, Selection, onde cada algoritmo tem uma forma diferente de ordenar, porém, uns podem ser mais ágeis ou menos ágeis Cormen (2002).

Almeja-se com este estudo melhorias de custos nos sistemas que contenham os produtos de uma lista de um estoque, como sistemas de supermercados, que tenha cadastrados esses dados. Para que seja eficiente, satisfatória, a venda e alteração dos itens da lista de produtos digitais do estoque ou busca por itens no mesmo sistema dentro dos supermercados, e os funcionários consigam uma busca rápida nesse sistema, tornando o atendimento mais rápido, que implique na satisfação do cliente em relação ao atendimento recebido no estabelecimento, fazendo com que esses clientes voltem, comprem novamente nesse supermercado requerendo lucros gerados com a rapidez do atendimento. Caso uma pessoa necessite de algo e vai a um supermercado realizar compras, mas tem outras atividades para fazer após esta atividade, ela deseja que seu atendimento seja o mais rápido possível para que consiga realizar suas outras tarefas.

Um sistema computacional mais rápido pode satisfazer mais pessoas em seu atendimento, mas isso acontece caso o sistema seja eficiente. Assim, neste projeto estudou-se como os algoritmos de ordenação interferem na busca por um registro contido no sistema, já que assim como foi dito, cada algoritmo tem uma forma diferente de realizar a ordenação dos dados. Dessa forma a rapidez do atendimento em determinado supermercado pode refletir NA MELHORIA DOS CLIENTES pode ser que desejem um rápido atendimento, pois isso mostra eficiência do supermercado. Um cliente saindo satisfeito com o atendimento, pode voltar outras vezes e realizando outras compras, aumentando assim os lucros dos donos de supermercados. A forma de estudar como um sistema computacional de

supermercado pode ser mais rápida pode ser esse trabalho, onde será realizado o estudo dos diferentes casos de ordenação, que interferem no processamento do sistema.

Anseia-se que este trabalho possa se tornar torne uma nova referência as pessoas que tenham interesse pelo tema deste projeto. A produção e publicação de um projeto de iniciação científica como este, que explane o tema e os resultados obtidos nesta pesquisa é uma forma de levar o nome do instituto e da provedora de fomento que ajudou indiretamente na elaboração deste projeto, além de tornar público uma nova referência para a comunidade em geral

5. Objetivos

Obter a eficiência em sistemas de supermercado tornando assim, mais um trabalho disponível para consulta na área da Análise de Algoritmos.

Objetivos Específicos:

1- Coletar os dados contidos em um supermercado, ou criar uma lista fictícia com os possíveis dados de um supermercado, para que seja feita a realização dos testes.
2- Implementar os Algoritmos de Ordenação e busca que serão utilizados durante a realização do estudo.
3- Realizar os testes de ordenação de algoritmos em computadores e máquinas de supermercados.
4- Realizar os testes de busca em computadores e máquinas de supermercados sem os itens estarem ordenados.
5- Realizar os testes de busca em computadores e máquinas de supermercados com os itens ordenados.
6- Analisar os testes de ordenação de algoritmos em computadores e máquinas de supermercados.
7- Analisar os testes de busca em computadores e máquinas de supermercados sem os itens estarem ordenados.
8- Analisar os testes de busca em computadores e máquinas de supermercados com os itens ordenados.
9- Comparar os melhores casos e resultados dos efeitos dos Algoritmos de Ordenação.
10- Elaborar o Resumo e Relatório Final (atividade obrigatória)

11- Preparar a Apresentação Final para o Congresso (atividade obrigatória)

6. Metodologia

Durante o desenvolvimento desta pesquisa foram utilizados recursos de hardware e software. Em relação ao hardware, destaca-se o uso de um laptop lenovo composto por um processador i5 e 4 Gb de memória RAM. No que diz respeito aos softwares utilizados, têm-se: a linguagem C usada na implementação algoritmos de ordenação e algoritmos de busca. Essa linguagem foi adotada porque é uma das mais conhecidas pela comunidade científica da área computacional na construção de um sistema baseado programação estruturada, que é a mais conhecida para tal meio de implementação.

A ferramenta devc++ foi escolhida para desenvolver o sistema que contenha os algoritmos citados. Uma das vantagens do devc++ é que erros simples, como erro de sintaxe, podem ser facilmente corrigidos com os recursos embutidos na própria ferramenta, e isso

reduz o tempo alocado no desenvolvimento do sistema. Lembrando que o foco principal do projeto era analisar os testes a serem feitos no sistema após a implementação dos algoritmos.

As execuções dos testes foram realizadas da seguinte forma: o sistema principal, onde estavam as funções de leitura dos dados do arquivo, lia do arquivo de dados.txt, onde continha os registros do arquivo com os campos de descrição, código e categoria do produto, executava buscas (sequenciais) e ordenava os dados com os algoritmos e refazia a busca sequencial e dessa vez realizava-se também a busca binária. Os algoritmos eram construídos externamente em relação ao código principal e eram inseridos apenas na realização dos seus testes.

Os testes foram realizados usando uma lista de dados reais fornecida gentilmente pelo Supermercado Dona Novinha, que possui sua sede na cidade de Itacoatiara. Para o desenvolvimento desse sistema, foi adotada uma metodologia composta pelas etapas descritas a seguir:

6.1 Implementações dos Algoritmos.

A fim de atingir um dos principais objetivos do projeto iniciou-se após o levantamento bibliográfico e a definição da plataforma Dev C++ para implementação do projeto em C, a implementação dos algoritmos de ordenação sendo eles o Booble-sort, Insertion-sort, Selection-sort, Merge-sort e Quick-sort. Os algoritmos foram adaptados para o problema que simulava a ordenação de itens de um supermercado.

No algoritmo que simulava o sistema de supermercado em si, houve a implementação das funcionalidades de ler e inserir do arquivo onde estava a base de dados utilizada para testes, deu-se também a implementação das buscas (binária e sequencial).

Os campos a serem utilizados nesse sistema seriam o código de barras, código do produto, descrição e categoria do produto. Entretanto, tornou-se inviável a leitura do campo de códigos de barras em linguagem c, pois este tinha 13 dígitos e na literatura encontrou-se a possibilidade de ler até 10 dígitos.

Em vista dos presentes fatores, adaptou-se os campos descrevendo-os da seguinte forma: código, descrição e categoria do produto. Categoria não veio na lista original emitida pelo supermercado, mas foi inserida por motivos essenciais, para que se obtivesse uma melhor visão sobre a ordenação de dados.

6.2 Lista de dados para realização dos testes.

Para que os resultados do projeto se aproximassem de valores reais. Foi enviada uma solicitação ao Supermercado Dona Novinha de Itacoatiara, para que fornecesse uma lista de itens de seu estoque, afim de auxiliar na pesquisa.

Foi gentilmente cedido pela assessoria do supermercado uma lista com 55386 itens para a realização de testes. Os respectivos campos originais do sistema eram código de barras, código, e descrição do produto. Houve uma adaptação que se deu por conta de uma restrição a leitura de itens com mais de treze dígitos da linguagem C. Tornando a lista apenas o código do produto, a descrição e a criação de um novo campo que é a categoria.

Essas mudanças sobre os campos de armazenamento foram arranjadas utilizando o Excel, e então gerou-se uma base de dados com essas informações em um arquivo renomeado dados.txt. Na implementação do sistema, usou-se uma struct (vetor dinâmico de dados) para armazenar temporariamente os dados lidos do arquivo dados.txt e com base

nessa estrutura onde estavam armazenados os dados do arquivo que foram realizados os testes de ordenação e busca, respectivamente para o projeto.

7. Resultados e Discussão

Os resultados obtidos por este projeto foram: realização de um levantamento bibliográfico, no qual foram obtidos embasamentos teóricos e ferramentas para a realização do projeto; implementação do sistema (leitura e escrita no arquivo) e dos algoritmos de ordenação em C; obtenção de uma base de dados com 55386 itens para a realização dos testes algoritmos que foram implementados, pois assim a pesquisa buscou sempre chegar o mais próximo do resultado real; A realização e análise dos testes dos algoritmos, levando em consideração a complexidade computacional, o tempo de ordenação, quantidade de trocas realizadas e a quantidade de comparações feitas por cada algoritmo de ordenação. Nesta seção lista-se como se comportava cada algoritmo e posteriormente os resultados tabulados.

7.1 Funcionamento dos Algoritmos de Ordenação

Booble-Sort, o estudo dos Algoritmos de Ordenação, iniciou pelo mais básico deles, levando em consideração a lógica de implementação, o Booble-Sort (CORMEM, ANO). Neste algoritmo cada elemento da posição i será comparado com o elemento da posição i + 1, ou seja, um elemento da posição 2 será comparado com o elemento da posição 3. Caso o elemento da posição 2 for maior que o da posição 3, eles trocam de lugar e assim sucessivamente. Por causa dessa forma de execução, a estrutura terá que ser percorrida quantas vezes for necessária, tornando o algoritmo ineficiente para quantidade de dados muito grandes. (ZIVIANI, 1999).

Insertion-Sort, considera que o primeiro elemento está ordenado (ou seja, na posição correta). A partir do segundo elemento, insere os demais elementos na posição apropriada entre aqueles já ordenados. O elemento é inserido na posição adequada movendo-se todos os elementos maiores para posição seguinte da estrutura, segundo Ziviani (1999). Torna-se melhor que o Booble-Sort pois após o primeiro passo o algoritmo não repete a comparação para aquela posição da estrutura, ou seja, reduz o número de comparações apesar de continuar custoso.

Selection-Sort, este algoritmo é baseado em se passar sempre o menor valor da estrutura para a primeira posição (ou o maior dependendo da ordem requerida), depois o segundo menor valor para a segunda posição e assim sucessivamente, até os últimos dois elementos. (ZIVIANI, 1999).

Merge-Sort, o problema maior é dividido em problemas menores e os problemas menores obtidos são novamente divididos sucessivamente de maneira recursiva. Então, o resultado do problema é calculado quando o problema é pequeno o suficiente. Com isso os resultados dos problemas menores são combinados até que seja obtida a solução do problema maior, segundo Cormem (1999). Algoritmos que utilizam o método de partição são caracterizados por serem os mais rápidos dentre os outros algoritmos pelo fato de sua complexidade ser sempre a mesma. (ZIVIANI, 2016)

Quick-Sort, considerado neste e em outros trabalhos o algoritmo mais eficiente na ordenação por comparação. Nele se escolhe um elemento chamado de pivô, a partir disto é organizada a lista para que todos os números anteriores a ele sejam menores que ele, e todos os números posteriores a ele sejam maiores que ele. Ao final desse processo o número pivô já está em sua posição final. Os dois grupos desordenados recursivamente sofreram o mesmo processo até que a estrutura fosse totalmente ordenada. (ZIVIANI, 1999)

Busca sequencial, dá-se na procura de um item que está em algum lugar da estrutura não ordenada. Pergunta-se sequencialmente durante a estrutura se a posição detém a chave que está sendo buscada. Exemplo A = {4, 2, 8, 9} e a há uma busca pelo registro 8 contido no arquivo, assim passa-se pelos valores 4, 2 até chegar no 8 sequencialmente (CORMEM, 1999).

Busca binária, é necessário que os itens da estrutura fossem previamente ordenados e ocorre partindo sempre do descarte de metade dos itens, pois verifica se a chave e maior ou menor do que o centro do vetor que está sendo particionado. Exemplo A = {2, 4, 8, 9} e a há uma busca pelo registro 8 desta estrutura, assim particiona-se o vetor em duas part es, obtendo C={2, 4} e D={8,9}. Neste exemplo, o registro 8, seria maior que o item da metade do vetor (4) assim, deveria ser escolhido a partição D, e faria o processo novamente obtendo E={8} e F={9} como o valor chave (8) seria o item central E={8} a busca estaria completa. (CORMEM, 1999)

7.2 Tabelas

Encontra-se na *tabela1* o resultado dos algoritmos de busca sequencial antes e depois dos itens serem ordenados. Mostrando respectivamente sua complexidade, tempo de execução, número de verificações. Estipula-se após a tabela o melhor e o pior caso para uma busca neste gênero. Percebe-se também que o conjunto estar ordenado, para este experimento não necessariamente torna a busca mais eficiente.

TABELA1: Resultados dos testes de Busca Sequencial de complexidade O (n).

Número de Registros	Antes da Ordenação		Após a Ordenação	
	Tempo	Verificações	Tempo	Verificações
10 mil	0.25s	63	1s	2743
20 mil	0.25	63	4s	5484
30 mil	0.25	63	10s	8225

Obs: O tempo após a ordenação é o tempo de ordenação somado com o tempo de busca do registro no arquivo, esses dados são uma média aritmética de várias buscas.

Na busca sequencial, o pior caso é quando é necessário realizar uma busca pelo item armazenado no último registro da estrutura. Exemplo Vet = {1,2, ..., n-1, n}, e busca-se pelo valor n, pois o algoritmo fará n comparações e demorará o tempo máximo para encontrar o registro. Entretanto, o melhor caso é quando o item pelo qual ocorre a busca encontra-se no primeiro registro do vetor, pois o algoritmo faz uma comparação e atinge um tempo mínimo de retorno.

Na tabela 2, o resultado dos algoritmos de busca binária após a ordenação dos itens, esse teste só ocorreu dessa forma pois no algoritmo de busca binária o funcionamento só ocorre corretamente se os itens estiverem previamente ordenados. Esta tabela mostra respectivamente o tempo de execução, número de verificações, e estipulando também o melhor e o pior caso para a busca por valores contidos nos registros de tamanho 10 mil, 20 mil, 30 mil e 55mil itens.

Descrição de melhor e pior caso:

- **Melhor caso da busca binária**: O valor da busca (registro) estar na posição central do arquivo [Tamanho/2].

- **Pior caso da busca binária**: Estar no Primeiro ou no último registro do arquivo [1 ou n].

TABELA 2: Busca Binária após a ordenação, independente do algoritmo.

Item inexistente	10 mil itens	20 mil itens	30 mil itens
Tempo	5.356s	5.798s	6.025s
Comparações	13	14	15

Item existente	10 mil itens	20 mil itens	30 mil itens
Tempo	1.56s	5.798s	6.025s
Comparações	9	13	14

Dispõe-se na tabela 3 os resultados e comparações entre os algoritmos de ordenação na execução da ordenação dos itens. Este foram feitos um a cada vez, inseridos no código principal quando ocorreria a ordenação dos testes. Esta tabela mostra respectivamente o tempo de execução, número de comparações e número de trocas e a complexidade de cada um, e estipulando também o melhor e o pior caso para a ordenação de acordo com a política de cada algoritmo. Estipula-se após a tabela o melhor e o pior caso para cada algoritmo.

TABELA 3: Análise dos Algoritmos de Ordenação.

10 mil	Tempo	Comparações	Trocas	Complexidade
Booble	1.91s	49995000	36310396	O (n^2)
Insertion	1.32s	49995000	36310396	O (n^2)
Selection	0.98	20366736709	1986750535	O (n^2)
Merge	0.32	27838	5263527	O (lg n)
Quick	0.13	2347	274323	O (lg n)

20 mil	Tempo	Comparações	Trocas	Complexidade
Booble	6.45s	199990000	145654809	O (n^2)
Insertion	4.36s	200010000	99641310	O (n^2)
Selection	1.76s	2108235587	1986760535	O (n^2)
Merge	0.67	62438	2368605	O (lg n)
Quick	0.35	6348	24327	O (lg n)

30 mil	Tempo	Comparações	Trocas	Complexidade
Booble	13.22s	449985000	340104180	O (n^2)
Insertion	9.035s	450015000	226325772	O (n^2)
Selection	2.12s	1858240587	1986770536	O (n^2)
Merge	1.58	2528993	4872389	O (lg n)
Quick	0.85	642823	2767382	O (lg n)

Melhores e piores casos:

- **Booble Sort**, melhor caso: Verifica-se quando os dados da estrutura estão devidamente ordenados. E, o pior caso: Verifica-se quando estão ordenados na ordem inversa.

- **Insertion-Sort**: melhor caso: Verifica-se quando os dados da estrutura estão devidamente ordenados. E, o pior caso: Verifica-se quando estão ordenados na ordem inversa.
- **Selection-Sort**: Os dados estarem ordenados não necessariamente significa ser o melhor caso, então, define-se como O (n^2). E, no pior caso: Existe uma troca para cada loop, e para cada troca, três movimentos.
- **Merge-Sort**: A complexidade do merge-sort é a mesma para o pior, médio e melhor caso, sendo O (n log n). Independente da situação dos dados na estrutura, o algoritmo irá sempre dividir e intercalar os dados.
- **Quick-Sort**: Acontece quando as partições têm sempre o mesmo tamanho, balanceadas. E, no pior caso: Quando o pivô é sempre o maior ou menor elemento, ou seja, quando desequilibra-se as partições.

8. Referências

CORMEN, Thomas H.; LEISERSON, Charles E.; RIVEST, Ronald L.; STEIN, Clifford. Introduction to Algoritms. Massachusetss. 2ed. 2002

ZIVIANE, Nivio. Projeto de Algoritmos com implementações em Pascal e C. 2ªed Belo Horizonte: Editora Pioneira Thomson. 2005.

MEDINA, Marco; FERTING, Cristina. Algoritmos e Programação-Teoria e Prática. Editora: Novatec, 1ª ed, São Paulo 2005

CARVALHEIRO, Fábio H.; SETZER, Waldemar W. Algortimos e Sua Análise-Uma Introdução Didática. Caderno de Revista do Professor de Matemática. 2009

JÚNIOR, Antônio C. de N. Métodos de Ordenação Interna. Ouro Preto. 2008

SILVA, Eliezer S. Estudo comparativo de Algoritmos de Ordenação. Universidade Federal do Espírito Santo. São Mateus, 2010

MENDES, Douglas R. Java com: Ênfase em Orientação à Objetos. Editora: Novatec. 1ª ed. São Paulo 2009

PRESSMAN, Roger. Engenharia de Software, 6ª ed. São Paulo, 2006

OKUYAMA, Fábio Yoshimitsu. Desenvolvimento de Software l, Editora: Bookman. Porto Alegre 2014

BALARDIN, E.; FOGLIATTO, F. S. XXXI ENCONTRO NACIONAL DE ENGENHARIA DE PRODUCAO Inovação Tecnológica e Propriedade Intelectual: Desafios da Engenharia de Produção na Consolidação do Brasil no Cenário Econômico Mundial. Belo Horizonte, 2011.

BARBOSA, H.A.L.; CALDAS, G.B.; CRUZ, F.R.B. Proposta de análise de desempenho de algoritmos para otimização de redes de filas M/G/c/K baseada em DOE. UFMG, 2014.

TARÔCO, J.; SANTOS, E. Uma comparação de Algoritmos de Ordenação. Minas Gerais, 2014.

GUEDES, S.B.; PROENÇA, M.C. Simulador Didático de testes de Algoritmos de Ordenação. Porto Alegre, 2018.

9. Cronograma de Atividades

Nº	Descrição	Ago 2015	Set	Out	Nov	Dez	Jan 2016	Fev	Mar	Abr	Mai	Jun	Jul
1	Revisão da Literatura	R											
2	Coletar os dados contidos em um supermercado, ou criar uma lista fictícia com os possíveis dados de um supermercado, para que seja feita a realização dos testes.	R	R										
3	Implementar os Algoritmos de Ordenação e busca que serão utilizados durante a realização do estudo.		R	R	R	R	R						
4	Realizar os testes de ordenação de algoritmos em computadores e máquinas de supermercados						R	R	R				
5	Realizar os testes de busca em computadores e máquinas de supermercados sem os itens estarem ordenados								R				
6	Realizar os testes de busca em computadores e máquinas de supermercados com os itens ordenados									R			
7	Análise dos testes de ordenação de algoritmos em computadores e máquinas de supermercados										R		
8	Análise dos testes de busca em computadores e máquinas de supermercados sem os itens estarem ordenados								R	R			
9	Análise dos testes de busca em computadores e máquinas de supermercados com os itens ordenados									R	R		
10	Comparação dos melhores casos e resultados dos efeitos dos Algoritmos de Ordenação										R	R	
													R

11	- Elaboração do Resumo e Relatório Final (atividade obrigatória) - Preparação da Apresentação Final para o Congresso (atividade obrigatória)												