Jose Diego - 1761, Vitor - 3049

Trabalho de Projeto e Análise de Algoritmos

Brasil

20 de Setembro de 2018

Jose Diego - 1761, Vitor - 3049

Trabalho de Projeto e Análise de Algoritmos

Trabalho referente a algoritmos de ordenação para a disciplina de Projeto e Analise de Algoritmos, aonde será feito um estudo sobre algoritmos fortemente conhecidos na literatura.

Universidade Federal de Viçosa

Campos Florestal

Ciência da Computação

Brasil

20 de Setembro de 2018

Resumo

Os algoritmos de ordenação representam um fundamental campo tanto no estudo teórico de algoritmos como também em aplicações práticas. Na literatura podemos obter uma variedade de algoritmos de ordenação propostos, desta maneira estudar o comportamento destes algoritmos e suas complexidades computacionais se torna necessário na determinação de algoritmos mais eficientes ou adequados para determinados contextos. Este trabalho aborda um estudo comparativo de dois algoritmos de ordenação, com a intenção de observar experimental a complexidade computacional dos mesmos.

Palavras-chaves: algoritmos, ordenação, complexidade.

Lista de ilustrações

Figura 1 – Funções para calculo dos tempos. 8
Figura 2 – Algoritmo de Busca. 9
Figura 3 – Tempo do Quick Sort. 9
Figura 4 – Tempo do Select Sort. 10
Figura 5 – Uma comparação entre o SelectSort x QuickSort. 11

Sumário

1 Introdução **5**

2 Metodologia **6**

3 Desenvolvimento **7**

3.1 Algoritmos de Ordenação 7

3.1.1 Escolha dos algoritmos 7

3.1.2 Desafio Proposto pela Dupla 7

3.2 Implementação 7

3.3 Resultados 9

Conclusão **12**

Referências **13**

1 Introdução

Como já foi dito a utilização de algoritmos de ordenação é muito usada na área da computação. E seus estudos se mostram importantíssimos para maior compreensão dos mesmos, para que desta maneira se possa melhor utilizá los para os vários contexto práticos possíveis. Para isso o trabalho foi desenvolvido seguindos as especificações encontradas para consulta para sua realização, como também iremos ver ao longo do desenvolvimentos que novos desafios foram propostos e de posse deles métodos na literatura foram utilizados para a sua resolução.

2 Metodologia

A metodologia adotada consiste,primeiramente, no estudo teórico da complexidade e comportamento de cada algoritmo, em seguida a implementação destes na linguagem C e a execução e medição de tempo em algumas instâncias de problemas , que consistem de vetores de n tamnhos que precisam serem analisados. As medições de tempo serão apresentadas em gráficos de tempo x tamanho do vetor. Todas os exemplos e medições foram feitas em um computador com um processador Intel i5 (2.4GHz), 6GB de memória em uma distribuição Windows 10.

3 Desenvolvimento

3.1 Algoritmos de Ordenação

O trabalho proposto consiste na realização da comparação de algoritmos de ordenação dos quais foram pedidos que os mesmo serem de características distintas para que dessa maneira tivéssemos um maior escopo de problemas, onde então poderíamos entender melhor a gama de detalhes dos quais os estudos em ordenação de algoritmos estão sujeitos.

3.1.1 Escolha dos algoritmos

Diante do item acima foi escolhido os algoritmos de ordenação Quick Sort que tem uma complexidade O(nlogn) no melhor caso e Select Sort que tem uma complexidade quadratica, estes foram escolhidos por atenderem as necessidades pedidas no trabalho já que se tratam de algoritmos de complexidades distintas, além de bastantes conhecidos na literatura, como em Ziviani (2010) para mais informações.

3.1.2 Desafio Proposto pela Dupla

O Desafio proposto pela a dupla foi a implementação de um vetor de lista emcadeada ordenado por hash. Permitindo o método *BuscaHash*, O(1) seja qual tamanho determinado em suas busca pela soma de dois naturais igual a X. Apesar da ordenção de um dado vetor em uma tabela hash ter um custo expressivo, este mesmo custo é valido quando se coloca o método pesquisa da hash em comparação com os demais, no qual é percepitivel um ganho muito considerável da hash.

3.2 Implementação

A implementação foi desenvolvida na linguagem C, utilizando ferramentas de desenvolvimento como a IDE CodeBlocks. É importante ressaltar que foi utilizado um vetor de tamanhos distintos, com valores gerados aleatoriamente, e que os mesmos vetores estavam em um intervalos de um menor tamanho de n como 1000 até maiores valores para n como 100000. Para todos eles o tempo médio foi contabilizado diante dos algoritmos dados. Além dos algoritmos de ordenação já mencionados, foi também usado hash.

Este que foi implementado usando um vetor de lista em cadeada, no qual cada posição desse vetor representava um hash(x), para um determinado numero, todavia quando há colisões em uma mesma posição e adicionado uma celula a mais na lista encadeada. Este metodo de ordenção por index permite uma melhor disperçâo dos dados, como também uma pesquisa muito eficiente e curta devido a disperção dos dados segue imagens do algoritimo de ordenaçâo de uma vetor hash:

A seguir podemos ver alguns trechos de códigos referentes às implementações

Figura 1: Funções para calculo dos tempos.

```
145     LARGE_INTEGER tf, ti, freq;
146     float tempTotal;
147
148     QueryPerformanceCounter(&ti);
149     Ordenacao(vetor, 0, tamVetor-1);
150
151     QueryPerformanceCounter(&tf);
152     QueryPerformanceFrequency(&freq);
153     tempTotal = (float)(tf.QuadPart - ti.QuadPart)/freq.QuadPart;
154
```

Para a computação do tempos gastos foi utilizada as funções da biblioteca windows.h as variáveis tf e ti são respectivamente, o tempo final e inicial retornado por estas funções. E estes tempos estão ligados diretamente com a quantidade de instruções executadas neste processo. Agora Freq é uma variável que contém a frequência total de clock que foram necessárias para ser processado todo o código, tanto para ordenação quando para busca presentes no código. O resultado, então, é calculado e armazenado em tempoTotal, que recebe a diferença de tempo dividido pela frequência, além disso o tempo total é baseado em uma média para 5 execuções feitas para um melhor resultado final.

Para a resolução do problema originalmente na lista de exercícios da disciplina de Projeto e Análise de Algoritmos, foi desenvolvido um algoritmo de busca binária para a obtenção dos resultados, abaixo podemos ver este algoritmo.

Outro resolução foi o algoritimo de *Busca2* , havendo um ganho de velocidade quase 2x mais rapido que em relaçao ao primeiro.

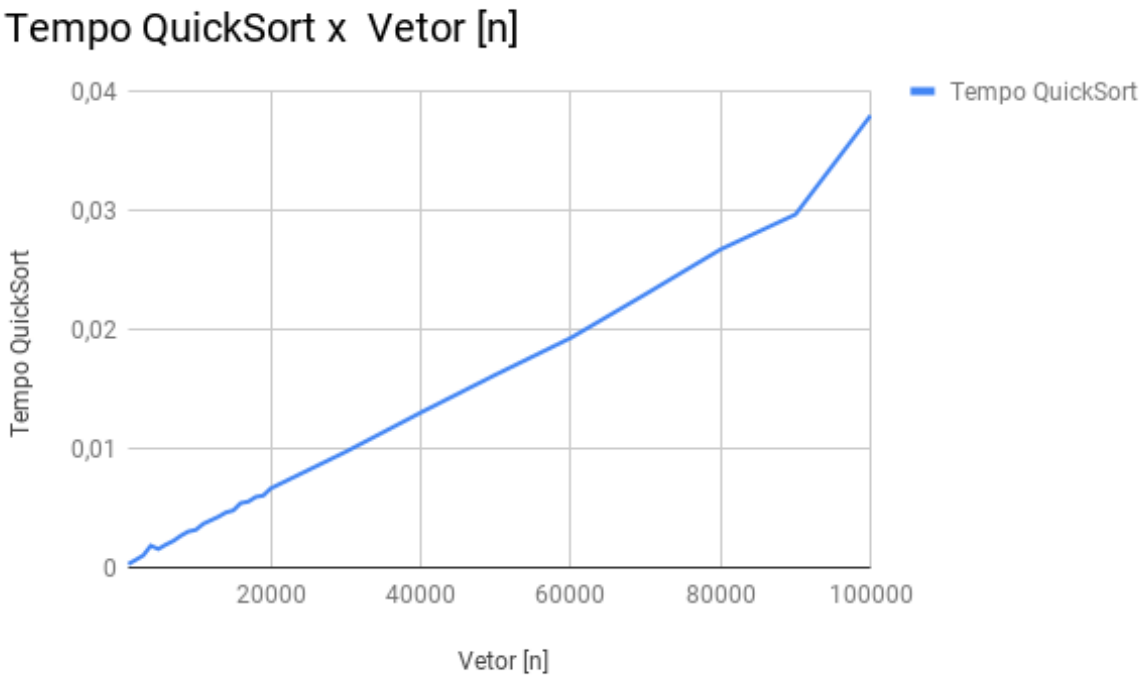
Figura 2: Algoritmo de Busca.

```
221     QueryPerformanceCounter(&ti);
222     while (vetor[i] < chave){
223         inf = i;
224         sup=tamVetor-1;
225         while (inf<=sup){
226             meio=(inf+sup)/2;
227             if (vetor[i] + vetor[meio] == chave){
228                 fprintf(arquivo,"Valor encontrado =  %d + %d = %d\n",vetor[i],vetor[meio],chave);
229                 chave = 10000000;
230                 break;
231             }else if (vetor[i] + vetor[meio]>chave){
232                 sup=meio-1;
233             }else{
234                 inf=meio+1;
235             }
236         }
237         i++;
238     }
239     QueryPerformanceCounter(&tf);
240     QueryPerformanceFrequency(&freq);
241     tempTotal = (float)(tf.QuadPart - ti.QuadPart)/freq.QuadPart;
```

3.3 Resultados

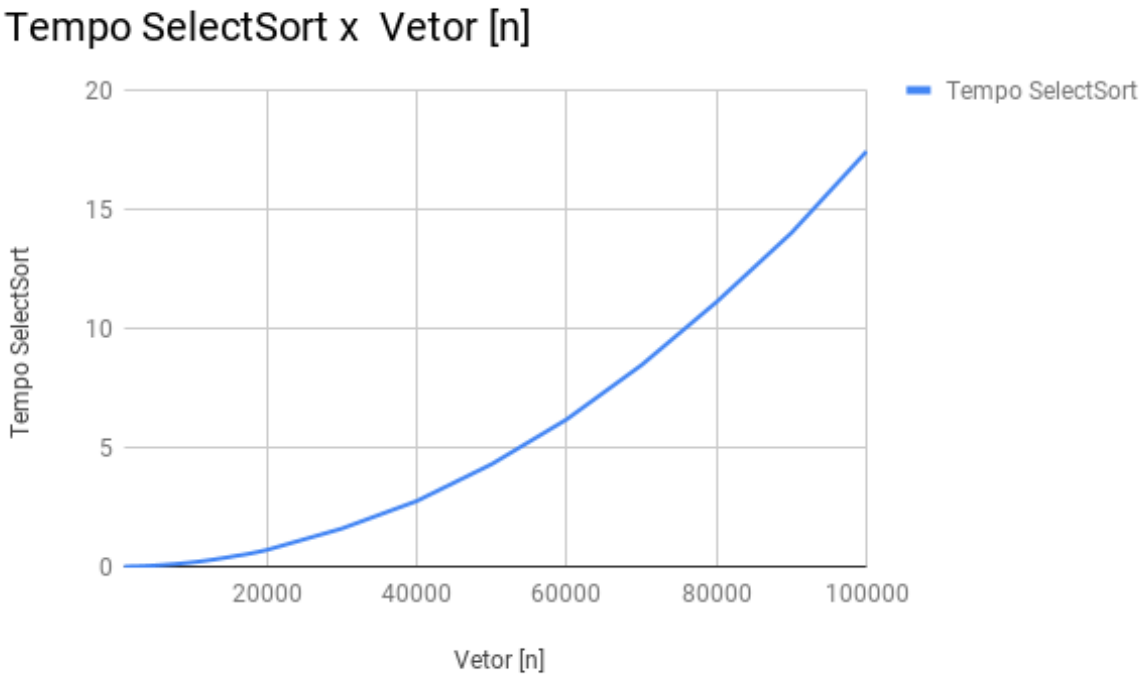
Nesta seção serão descritos os resultados obtidos, como já estudado pela literatura foi observado que o Quick apresentou se como melhor resposta ao problema com uma complexidade inferior, abaixo vemos em forma de gráfico os resultados obtidos, esse gráficos foi gerado a partir de um arquivo de saída que o programa oferece contendo uma lista com os tempos médios totais.

Figura 3: Tempo do Quick Sort.



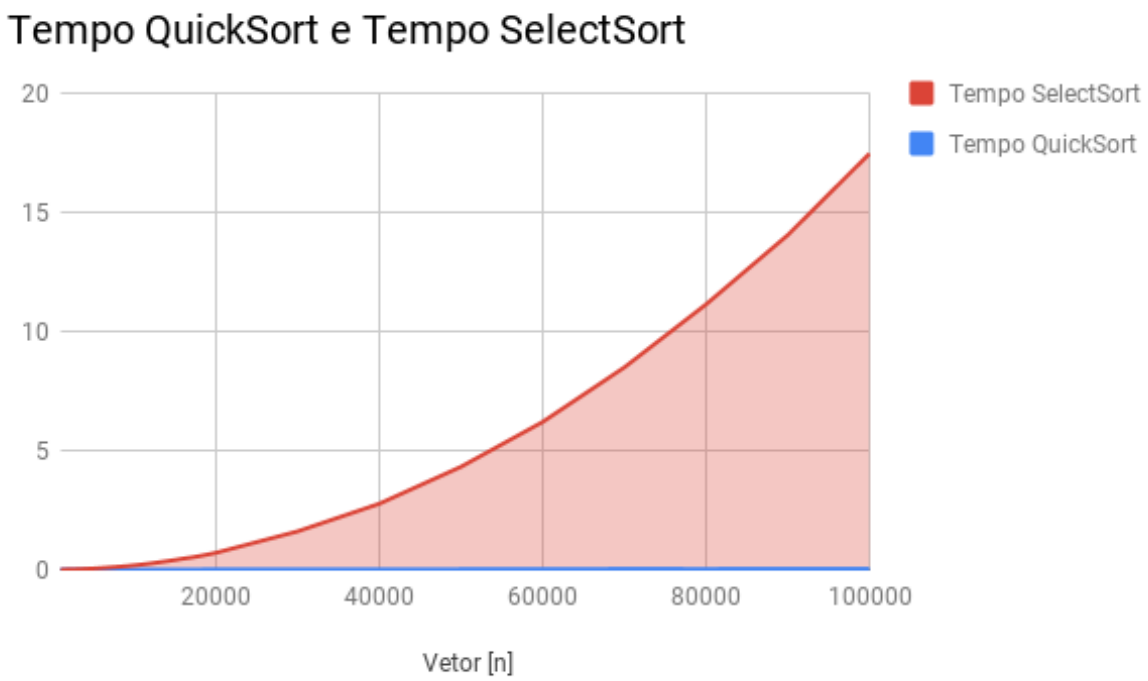
Como veremos também os resultados abaixo para o algoritmo de ordenação Selection Sort que apresentou também como esperado uma complexidade quadrática.

Figura 4: Tempo do Select Sort.



Longo após o grafico do Select Sort, veremos um gráfico comparativos entre os dois algoritmos de ordenação, onde através dele podemos observar o quão grande é a diferença entre os dois algoritmos

Figura 5: Uma comparação entre o SelectSort x QuickSort.



E como podemos observar os resultados obtidos nos dois gráficos se assemelham às funções matemáticas tanto para quadratica ao se tratar do Selectsort , como também de f(x) = n*log(n) ao falarmos do Quick Sort.

Conclusão

A Partir desse trabalho foi possível através de uma maneira prática verificar as complexidades dos algoritmos propostos no início do trabalho, que foram o Quicksort como também o Select Sort, foi também implementado algoritmos de busca para a resolução do problema proposta na disciplina, métodos como busca binária foi utilizado, além também de outros métodos largamente estudados na literatura como a Hash. Dessa maneira o trabalho se mostrou esclarecedor, de maneira que através dele podemos perceber experimentalmente o que já se estava estudando teoricamente através da literatura, aumentando assim o conhecimento sobre o tema algoritmos de ordenação que é de grande importância em ciências da computação.

Referências

ZIVIANI, N. *Projeto de Algoritmos*. [S.l.: s.n.], 2010. Citado na página 7.