relatório tabalho 1 de alg2

Matheus Ferreira Marquesini
Departamento de Informática
Universidade Federal do Paraná – UFPR
Curitiba, Brasil
matheus.marquesini@ufpr.br

I. Introdução

Esse relatório descreve uma implementação de código baseada em testes, destinada a testar algoritmos de classificação vetorial recursiva ((Insertion Sort, Selection Sort, Merge Sort e Quicksort) e algoritmos de busca vetorial recursiva (busca sequencial e busca binária) implementados em linguagem C. O relatório está dividido em níveis. Primeiramente, tem-se o preenchimento dos vetores com valores inteiros aleatórios para a ordenação de vetores e em seguida busca de um valor inteiro aleatório.

II. Iniciação dos vetores

Para a iniciação dos vetores para a realização dos testes, eu aloquei um vetor, que antes de cada chamada da função, é prenchidos por numeros inteiros aleatorios atraves da função rand() da biblioteca stdlib.h, onde esses valores aleatorios vao de 0 ate 99999.

III. Algoritmos de ordenação

A ordenação de vetores foi feita seguindo a seguinte ordem: Insertion Sort, Selection Sort, Merge Sort e QuickSort e. Esses algoritmos têm como entrada um vetor de inteiros indexado por [a..b], onde $b \leq$, e o tamanho do vetor é definido por n = b - a + 1. Portanto, todos os algoritmos têm como saída o vetor ordenado na forma crescente. Em relação ao custo dos algoritmos de ordenação, a tabela - 1 (custo), retrata o custo dos algoritmos acima mencionados:

Tabela I
Tabela de relaçao de custo de cada algoritimo.

	Tempo de custo	
algoritimo	melhor caso	Pior caso
Insertion Sort	$C(n)$	$C(n)$
Selection Sort	$C(n)$	$C(n)$
Merge Sort	$C(n \log_2 n)$	$C(n \log_2 n)$
Quick Sort	$C(n \log n)$	$C(n)$

Tendo em vista o custo de cada função sendo C, o custo em função de n, e n o tamanho do vetor, a tabela acima demostra as fórmulas para o cálculo do número de comparações entre elementos do vetor de cada algoritmo.

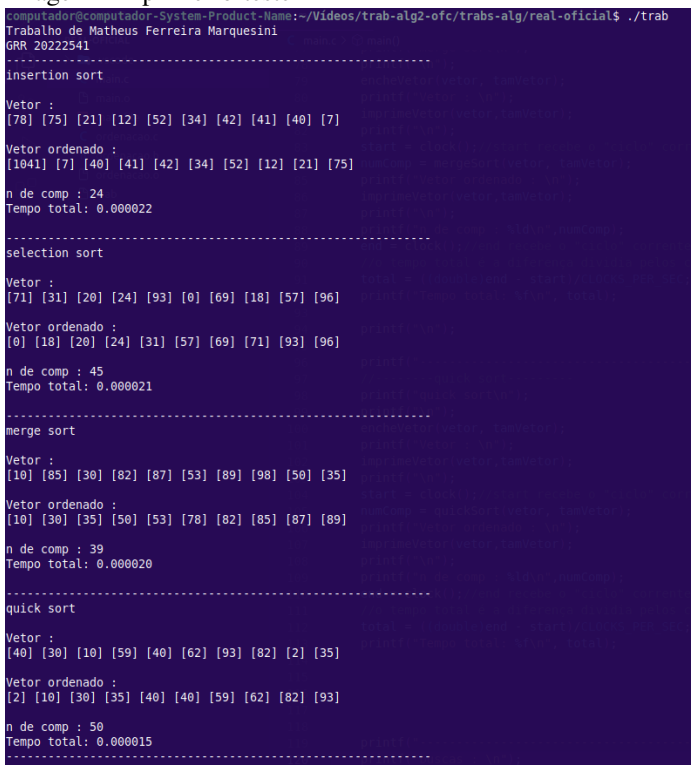
IV. Busca de um valor inteiro

Para fins experimentais dos algoritmos de busca, gerei um valor único aleatório no intervalo de 0 a 99.999, que representa o ser buscado pelos dois algoritmos de busca (Busca Sequencial e Busca Binária) e no caso da busca binaria foi usado para a ordenação o algoritmo QuickSort. Esses algoritmos têm como saída a posição no vetor que o valor buscado e retornara -1 caso o numero nao esteja no vetor. O valor inteiro único aleatório também foi gerado com as função srand e rand da biblioteca de funções stdlib.

V. Experimentos

A imagem - 1 primeiro teste, mostra os algoritmos executados em sequencia, com vetores de 10 posições, para fins de demostração de funcionalidade.

Imagem 1 - primeiro teste



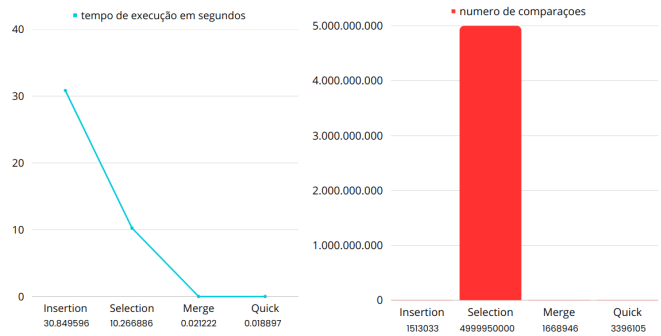
a imagem 2 - segundo teste, mostra os testes que foram citados anteriormente, para um vetor com indice de 100000 posiçoes e numeros aleatorios de 1 a 99999.

Imagem 2 - segundo teste.

```
Trabalho de Matheus Ferreira Marquesini
GRR 20222541
------------------------------------------------------------
insertion sort


n de comp : 1513033
Tempo total: 30.849596

------------------------------------------------------------
selection sort


n de comp : 4999950000
Tempo total: 10.266886

------------------------------------------------------------
merge sort


n de comp : 1668946
Tempo total: 0.021222

------------------------------------------------------------
quick sort


n de comp : 3396105
Tempo total: 0.018897
------------------------------------------------------------
```

```
Trabalho de Matheus Ferreira Marquesini
GRR 20222541
------------------------------------------------------------
buscar o numero 63950 :


busca sequencial
no indice : 17414
n de comp : 82587


busca binaria
no indice : 64145
n de comp : 13
```

Grafico 1 - graficos de tempo de execução e numero de comparações



Devido a relação do gráfico de barras estar em função do número de comparações, o algoritmo do Selection Sort, teve o número de comparações muito elevado em relação com qualquer outro, deixando quase inexpressel os demais.

Imagem 3 - terceiro teste.

VI. Considerações finais

Considerando os esperimentos feitos acima, é importante destacar que, para cada algoritmo de ordenação de vetores, assumiu-se que o vetor estava preenchido com valores inteiros distintos. No que diz respeito à classificação dos vetores usando esses algoritmos, eles permaneceram dentro de um intervalo de custo esperado em termos de tempo de execução. Em relação à Busca Sequencial e à Busca Binária, ambas retornaram a posição no vetor onde o valor buscado está localizado ou retornando -1 caso não encontrado. Portanto, para a busca de valores em vetores ordenados, a Busca Binária demonstrou ser a abordagem mais significativa.

Referências

Playlist no youtube de algoritimos 2 do porf. André Vignatti

slides prod. Gregio selection sort

slides prod. Gregio Insertion sort

slides prod. Gregio busca binaria sort

slides prod. Gregio Merge sort