Algoritmos de Ordenação

Documentação do Trabalho Prático 2
Universidade Federal de Minas Gerais – Sistemas de Informação

Estrutura de Algoritmos e Dados II
Professores: Dorgival Neto & Adriano Veloso

Tatiana Santos Camelo de Araujo – 2015086298

1. Introdução

O programa feito neste trabalho foi feito a fim de testar e fortalecer os conhecimentos adquiridos ao longo do semestre. O trabalho foca no uso de algoritmos de ordenação.

Para compilar o programa, usa-se o comando "gcc *.c -Wall -o main".

2. Implementação

Não foram usados TADs para esta implementação. Como a ordenação é de vetores de inteiros, foi possível fazer uma manipulação direta do valor de cada posição.

O código implementado é composto por 5 arquivos.

O arquivo *main.c* é o gerenciador do código, recebe e salva as entradas (algoritimo, tamanho de vetor, tipo de ordenação, impressão) e executa três switch para execução das escolhas do usuário. Primeiramente o programa cria um vetor de tamanho e tipo de ordenação escolhido e em seguida o ordena com o algoritmo. Por último, gerencia a escolha do usuário de imprimir ou não os vetores de entrada e saída.

O arquivo *ordenadores.c* é onde as funções de ordenação estão implementadas. Os algoritmos de ordenação retornam o vetor ordenado, assim, se o usuário quiser imprimir, ambos vetores são salvos separadamente. Todos algoritmos de ordenação usam a função *clock()* para salvar o tempo de execução e usam uma variável que salva o número de movimentações. Para a ordenação, foram implementados os métodos de ordenação de Bolha, Seleção, Inserção, ShellSort, QuickSort, HeapSort, MergeSort, RadixSort.

O arquivo *gerador.c* é onde estão implementadas as funções de gerar vetores. O gerador de vetores serve para quatro casos diferentes de vetores, sendo estes tipos aleatório, quase aleatório, inverso e ordenado. O gerador é feito ao receber do usuário a informação do tamanho escolhido. O vetor, então, possui o mesmo tamanho do que a informação inserida no sistema. Nos espaços de 0 a n-1 ficam os números inteiros escolhidos e no local n o finalizador de string (\0).

Para evitar a escolha de opções que não são as que interessam previamente o código, só há chamas de funções se o usuário inserir as informações corretamentes, como as mostradas antes de cada escolha. No caso de uma entrada diferente, o programa avisará o usuário que a entrada escolhida não é aceita. São quatro perguntas feitas ao usuário: qual algoritmo de ordenação ele quer usar, qual tipo de ordenação do vetor, qual o tamanho do dado vetor e se ele deseja que se imprima na tela o vetor antes e após ser ordenado. Todas perguntas recebem de resposta um número referente a opção que o usuário queira executar. Assim, o uso

do programa se torna fácil e direto, sem que o usuário tenha que, por exemplo, digitar o nome do algoritmo de ordenação todo.

3. Estudo de Complexidade

A tabela abaixo mostra todas as principais funções implementadas ao longo do código, tempo de execução e complexidade da **função**, com notação O.

Ordenadores	**Tempo**	**Compexidade**
bubble	-	$O(n^2)$
selecao	-	O(nˆ2)
insercao	-	O(nˆ2)
shell	$O(n^{\wedge 1,25})$	O(n*lg(n))
quick	$O(n^2)$	$O(n^2)$
heap	O(n*lg(n))	O(n*lg(n))
merge	O(n*lg(n))	O(n)
radix	O(lg(n))	O(n log n)
Gerador de vetor		Complexidade
ordenado	-	O(n)
aleatorio	-	O(n)
quaseOrdenado	-	O(n)
inverso	-	O(n)
Código total		Complexidade
Main	-	O(nˆ2)

4. Listagem de testes executados

Os testes foram executados de forma que o tempo de execução e movimentação de cara ordenador seja salvo.

Nãoa chei uma forma de implementar o vetor quase ordenado e por algum motivo o programa parou de gerar o executável após ter terminado. Compila sem erros e warnings na forma em que está sendo enviada.

5. Conclusão

O trabalho foi uma ótima forma de aplicar os conhecimentos de algoritmos e estruturas de dados no geral, com ênfase em ordenação. Uma das maiores dificuldades ao longo do código foram alguns problemas de conceitos dos algoritmos de ordenação. Também foi um desafio criar um gerador de vetores com números aleatorios, e aprender e usar corretamente a função de bibliotecas menos usuais, como clock().

6. Bibliografia

https://www.toptal.com/developers/sorting-algorithms
https://www.tutorialspoint.com/
http://www.cplusplus.com/