Trabalho 1 - Ordenação

Luiz Gustavo Dalmaz Paquete – GRR 20211794
Universidade Federal do Paraná
Curitiba, Brasil - 2022

INTRODUÇÃO

Este relatório apresenta os testes realizados no programa ”trab”que contém algoritmos recursivos de ordenação e busca em vetores ordenados, e uma breve análise dos mesmos, para a matéria de Algoritmo e Estrutura de dados 2 do curso de Bacharelado em Ciência da Computação.

Os testes foram realizados em um notebook utilizando o sistema operacional Linux, com as seguintes especificações:

- **OS:** Linux Mint 20.2 x86 64
- **CPU:** Intel i5 U 470 (4) @ 1.334GHz
- **GPU:** Intel Core Processor
- **Memoria:** 2642MiB / 3604MiB

ANÁLISE SOBRE OS TESTES

Como foram realizados:

Os testes foram realizados executando o arquivo por meio do comando ”./trab”após o mesmo ser criado com o comando ”make”, para evitar interferências nos resultados, o programa foi executado com exclusividade no computador, com entradas predefinidas e saídas anotadas conforme abaixo.

Teste 1

Para o primeiro teste, a entrada definida será o vetor de tamanho 10 = [0..9] descrito abaixo para que possamos testar alguns casos.

Posição	0	1	2	3	4	5	6	7	8	9
Valor	1	2	3	4	5	6	7	8	9	10

Sendo ”Comparação”o número de comparações entre elementos do vetor e ”Tempo de clock”o tempo inicial - o final do algorítmo em seg, os resultados obtidos com os algoritmos de ordenação foram:

Algoritmo	Comparações	Tempo de Clock
Insertion sort	9	0.000005
Selection sort	45	0.000003
Merge sort	34	0.000005
Quick sort	54	0.000007

Neste caso, pode se observar que o Insertion sort foi o algoritmo menos ”custoso”entre os demais, pois chegou em seu Melhor caso $C(n)^{-} = (n-1)$. E utilizando os algoritmos de busca neste mesmo vetor, para obtermos a localização de **1** temos os dados a seguir:

Algoritmo	Comparações	Tempo de Clock
Busca sequencial	10	0.000002
Busca Binaria	3	0.000001

Após analise, os dados mostram uma vantagem do algoritmo Busca Binária, em tempo e comparações.

Outro teste no mesmo vetor, agora buscando o valor **10** temos:

Algoritmo	Comparações	Tempo de Clock
Busca sequencial	1	0.000001
Busca Binaria	4	0.000001

E assim vemos um caso em que o algoritmo de Busca Sequencial se destaca, com menos comparações pois seu método de funcionamento se resume a procurar de trás para frente o valor dentro do vetor.

Teste 2

Utilizando da mesma metodologia do teste anterior, foi realizado outro teste, agora com o vetor de mesmo tamanho, com os valores abaixo:

Posição	0	1	2	3	4	5	6	7	8	9
Valor	10	9	8	7	6	5	4	3	2	1

E de resultado, teremos os seguintes dados:

Algoritmo	Comparações	Tempo de Clock
Insertion sort	45	0.000006
Selection sort	45	0.000003
Merge sort	34	0.000006
Quick sort	54	0.000004

Com os resultados acima, observamos que o algoritmo Selection sort, mantém seu numero de comparações do teste anterior pois possuí um numero fixo de $C(n) = \frac{n^2}{2}$ comparações, e nesta ocasião o algoritmo menos custoso foi o Merge sort, porém um pouco mais lento.

Teste 3

Neste teste foi utilizado de uma aleatoriedade maior entre valores e posições no vetor, para que sejam obtidos resultados fora do ”padrão”, o vetor utilizado e os dados obtidos foram estes a seguir:

Posição	0	1	2	3	4	5	6	7	8	9
Valor	24	19	45	95	79	30	112	51	87	66

Algoritmo	Comparações	Tempo de Clock
Insertion sort	22	0.000005
Selection sort	45	0.000004
Merge sort	34	0.000009
Quick sort	27	0.000004

Novamente analisando os dados acima, temos como mais ”balanceado”o algoritmo de busca Insertion Sort, e notamos que o algoritmo Merge sort, levou um tempo considerável para realizar a ordenação.

Teste com vetores de tamanhos diferentes

Agora o teste realizado irá medir a eficiência de cada algoritmo para diferentes tamanhos de vetores, e o impacto em tempo e comparações.

Teste 1

Teste realizado sob as mesmas condições dos anteriores, utilizando um vetor[0..19] com os valores a seguir: **VETOR = [1, 3, 2, 5, 9, 10, 20, 17, 4, 16, 18, 6, 13, 8, 7, 11, 14, 12, 15, 19]**

Algoritmo	Comparações	Tempo de Clock
Insertion sort	74	0.000008
Selection sort	190	0.000007
Merge sort	88	0.000010
Quick sort	97	0.000005

Observamos um aumento grande nas comparações em todos os algoritmos listados, porém como em quase todas os testes, o algoritmo Selection Sort se sobressai como o mais custoso, e o algoritmo Merge Sort como o mais demorado.

E para os algoritmos de busca, dado o valor **11** para ser encontrado, obteram os resultados a seguir:

Algoritmo	Comparações	Tempo de Clock
Busca sequencial	10	0.000002
Busca Binaria	4	0.000002

Se observa que a busca binaria teve uma vantagem em questão de comparações, e que o tempo de execução dos algoritmos foram iguais.