UFC – UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS DE SOBRAL
CURSOS DE ENGENHARIA DA COMPUTAÇÃO
ESTRUTURAS DE DADOS
PROFESSOR: JARBAS JOACI
RELATÓRIO: ALGORITMOS DE ORDENAÇÃO
ISRAEL DA SILVA PEREIRA - 497145

1. INTRODUÇÃO

Foram analisados a execução e o desempenho dos métodos de ordenação: BubbleSort, InsertionSort, QuickSort e HeapSort. Os vetores utilizados foram de tamanhos 10^3, 10^4, 10^5, 10^6, preenchidos de formas crescentes, decrescentes e aleatórios. Para analisarmos os resultados, foi utilizada a função clock(), da biblioteca time.h, para calcular o tempo de execução dos métodos. Sobre o preenchimento dos vetores aleatórios, foi utilizado a função rand() com o complemento srand(time(NULL)).

2. RESULTADOS

Os códigos foram executados três vezes e foi registrado, nas tabelas a seguir, o valor com mais frequência ou aquele que era o termo médio entre os três. Em caso de uma análise mais detalhada, encontra-se aqui o link correspondente a uma planilha com todos os tempos encontrados em cada execução, que não foram expostos aqui no intuito de não se prolongar o relatório.

Obs.: Alguns métodos executaram tão rápido, em determinadas características do vetor, que a função utilizada apresentou 0 como o tempo de execução.

2.1. BubbleSort

Vetor crescente	
Tamanho	Tempo (ms)
1000	0,000
10000	201,000
100000	18697,000
1000000	1937862,000

Vetor decrescente	
Tamanho	Tempo (ms)
1000	0,000
10000	391,000
100000	39167,000
1000000	5396726,000

Vetor aleatório	
Tamanho	Tempo (ms)
1000	0,000
10000	353,000
100000	45769,000
1000000	4831961,000

2.2. InsertionSort

Vetor crescente	
Tamanho	Tempo (ms)
1000	0,000
10000	0,000
100000	0,000
1000000	16,000

Vetor decrescente	
Tamanho	Tempo (ms)
1000	0,000
10000	281,000
100000	26417,000
1000000	3438757,000

Vetor aleatório	
Tamanho	Tempo (ms)
1000	0,000
10000	125,000
100000	12392,000
1000000	1640320,000

2.3. QuickSort

Vetor crescente	
Tamanho	Tempo (ms)
1000	0,000
10000	122,000
100000	-
1000000	-

Vetor decrescente	
Tamanho	Tempo (ms)
1000	0,000
10000	131,000
100000	-
1000000	-

Obs.: Em vetores crescentes e decrescentes de tamanhos 10^5 e 10^6, o programa simplesmente parou a execução. Vale destacar que isso não ocorreu no vetor aleatório.

Vetor aleatório	
Tamanho	Tempo (ms)
1000	0,000
10000	0,000
100000	16,000
1000000	178,000

2.4. HeapSort

Vetor crescente	
Tamanho	Tempo (ms)
1000	0,000
10000	0,000
100000	31,000
1000000	375,000

Vetor decrescente	
Tamanho	Tempo (ms)
1000	0,000
10000	0,000
100000	31,000
1000000	328,000

Vetor aleatório	
Tamanho	Tempo (ms)
1000	0,000
10000	0,000
100000	47,000
1000000	531,000

3. CONCLUSÃO

Analisando todos os dados, vemos algumas peculiaridades. O BubbleSort demonstrou ser no geral o mais demorado entre todos os quatros métodos analisados, independente do modo de preenchimento do vetor. O InsertionSort tem claramente seu melhor caso quando o vetor está crescente, tendo uma demora, praticamente, irrelevante. Em relação ao QuickSort demonstrou ser bastante rápido em vetores aleatórios, independentemente do tamanho, porém apresentou inconsistência quando executado nos modos crescentes e decrescentes de tamanhos 100000 e 1000000. Dentre todos, o HeapSort mostrou-se como o mais consistente e eficiente em questão de tempo, de modo geral.