Tempo de execução de algoritmos – FPAA

Nome: Nikolas Augusto Vieira Louret

Especificações do equipamento:

Processador: Ryzen 5 5600

Memórias: 16GB (2x8GB) DDR4

SO: Windows 11 Pro

Tempo de execução: 8,56min

Questão 1) Segue abaixo a tabela e os gráficos comparando o número total, máximo, mínimo e médio de comparações e trocas dos algoritmos de ordenação *BubbleSort* e *SelectionSort:*

Coluna1	BubbleSort	SelectionSort
TOTAL Comparações	249.997.500.000	249.997.500.000
MAX Comparações	4.999.950.000	4.999.950.000
MIN Comparações	4.999.950.000	4.999.950.000
MEDIA Comparação	4.999.950.000	4.999.950.000
TOTAL Trocas	124.958.077.122	4.999.416
MAX Trocas	2.511.079.276	99.994
MIN Trocas	2.489.089.308	99.979
MEDIA Troca	2.499.161.542	99.988

Tabela 1 - Número total, mínimo, máximo e média de comparações e trocas do BubbleSort e SelectionSort

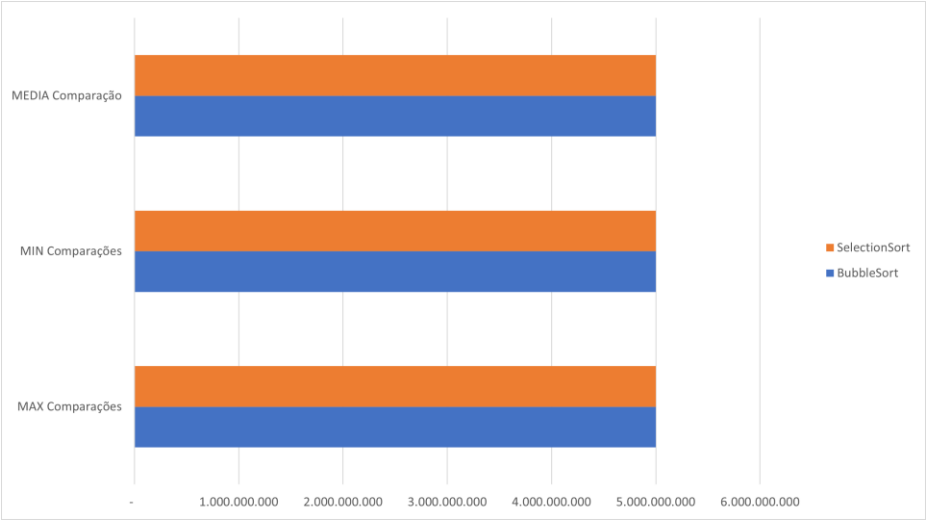

Figura 1 - Gráfico do número da média, mínimo e máximo do número de comparações

Figura 2 - Gráfico do número da média, mínimo e máximo do número de trocas

Tanto o *BubbleSort* quanto o *SelectionSort* são algoritmos de ordenação com complexidade $O(n^2)$, ou seja, eles podem ser bastante ineficientes para ordenar grandes conjuntos de dados. Por isso, é interessante analisar o número de comparações e trocas que cada um realiza para comparar seu desempenho.

O número de comparações que o *BubbleSort* realiza é da ordem de n^2, onde n é o tamanho do vetor. O número de trocas também é da ordem de n^2 no pior caso, ou seja, quando o vetor está completamente desordenado.

O *SelectionSort*, por sua vez, possui o número de comparações da ordem de n^2, assim como o *BubbleSort*. No entanto, o número de trocas é da ordem de n no pior caso, o que é uma vantagem em relação ao *BubbleSort* como é possível observar na *Tabela 1*.

Em termos de desempenho, o *SelectionSort* tende a ser um pouco mais rápido do que o *BubbleSort* em vetores grandes, já que realiza menos trocas. No entanto, ambos os algoritmos são bastante ineficientes para ordenar grandes conjuntos de dados.

Questão 2)

Tempo de execução: 5,34s

Segue abaixo a tabela e o gráfico de barras da busca sequencial:

Contadores	Quantidade
1/500	45437
501/1000	45422
1001/1500	45580
1501/2000	45776
2001/2500	45783
2501/3000	45502
3001/3500	45308
3501/4000	45887
4001/4500	45459
4501/5000	45565
5001/5500	45533
5501/6000	45673
6001/6500	45718
6501/7000	45319
7001/7500	45857
7501/8000	45267
8001/8500	45351
8501/9000	45422
9001/9500	45508
9501/10000	43642
10001/11000	90991

Tabela 2 - Contadores e as quantidades de elementos buscados no vetor

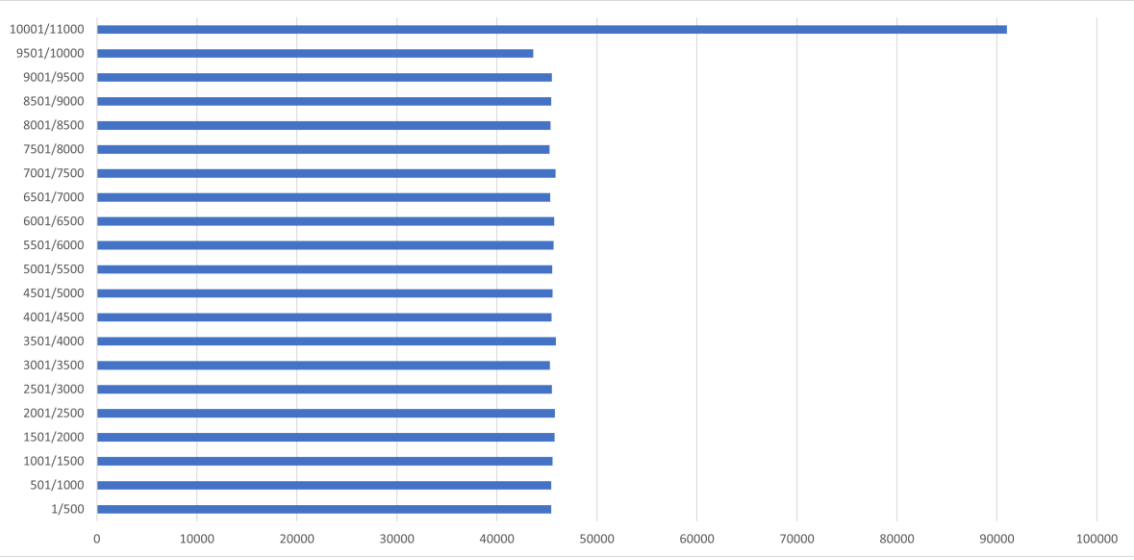

Figura 3 - Gráfico dos intervalos e as respectivas quantidades de itens buscados

Os resultados obtidos indicam a distribuição da frequência dos números sorteados em grupos de 500. É possível observar que a maioria dos números sorteados foram encontrados na faixa de 10.001 a 11.000, indicando que os valores sorteados estavam concentrados nessa faixa, por ser um intervalo com o dobro do tamanho comparado com os demais. Como a busca é baseada em valores aleatórios, existe uma mesma chance de se buscar um número qualquer em um dos intervalos, por isso os contadores são semelhantes, exceto pelo último contador na faixa dos 10.001 a 11.000.

A busca sequencial é um algoritmo simples, porém pouco eficiente em grandes conjuntos de dados, uma vez que ele precisa percorrer todo o vetor até encontrar o elemento desejado.