UNIVERSIDADE FEDERAL DA PARAÍBA UFPB – CAMPUS IV
DEPARTAMENTO DE CIÊNCIAS EXATAS - DCX
B. SISTEMAS DE INFORMAÇÃO - BSI

Avaliação de Desempenho de Sistemas
Prática 3 – Intervalo de confiança

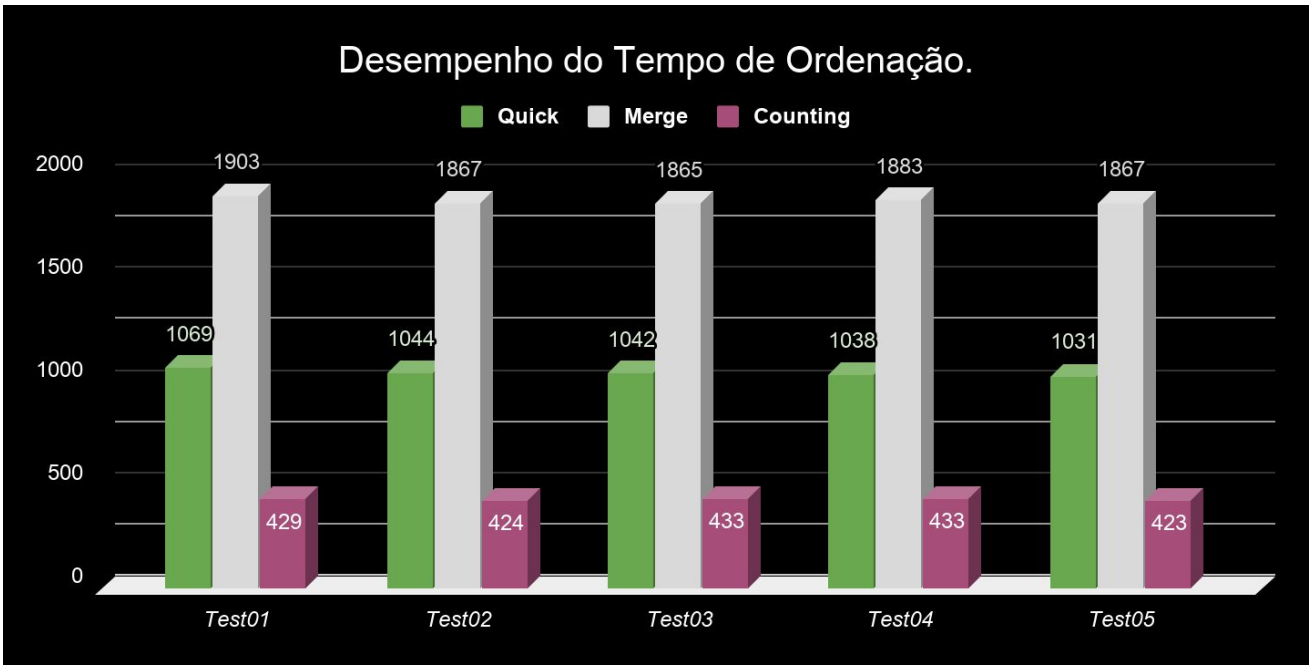
José Carlos Jr.; jose.carlos@dcx.ufpb.br
Mat. 2016046363

Rio Tinto - PB
Fevereiro - 2020

Perguntas

1. *Queremos comparar o desempenho de cada algoritmo de ordenação, medindo o tempo para ordenar arrays de inteiros de certos tamanhos e com diferentes faixas de possíveis valores inteiros.*

> ***a.*** *Realize 5 medições para cada algoritmo (quick, merge, counting) e com base nos dados obtidos calcule, para cada um, quantas vezes cada um deve ser executado (tamanho da amostra) para que se tenha um intervalo de confiança com uma margem de erro menor ou igual a 1%, para um nível de confiança de 95%. Use como parâmetros de entrada um tamanho de entrada de 10000000 (10 milhões) e o valor máximo da entrada 1000000 (1 milhão).*



As execuções dos testes foram realizados realizados na minha máquina enquanto alguns programas estavam em execução em segundo plano, porém em baixo consumo de desempenho (descanso). Os testes foram executados Visual Studio Code (editor de código-fonte), e não diretamente no terminal Linux. Os resultados do Counting ficaram em uma margem de time muitos próximos um do outro, já o algoritmo Merge ficou em um nível intermediário, mesmo assim suas margens extremas não ficaram tão distantes uma da outra e já Quick e último algoritmo testado o seu primeiro teste obteve um resultado onde sua margem de distanciamento de uma extremidade maior até a menor teve um time mais prolongado.

Os mesmos testes foram realizados pela segunda vez, e os resultados foram bastantes semelhantes e em alguns casos até iguais, onde a diferença

entre a primeira realização dos testes para com a segunda, foram bem menores. Ou seja, de acordo com tais testes, a probabilidade em que uma terceira realização dos mesmo os seus valores não iriam sofrer tantas alterações e seus valores também seriam próximos um do outro.

Tamanho de amostra para os maiores e menores valores obtidos com os teste com o Quick, Merge e Counting com o nível de confiabilidade de 95% e com um erro de margem de 1%. Os cálculos foram realizados em um site de soluções estatísticas onde tem uma calculadora online de amostragem e de margem de erro.

Algoritmo	Teste de Maior Valor	Tamanho De Amostra	Teste de Maior Valor	Tamanho De Amostra
Quick	1069	963	1031	932
Merge	1903	1589	1865	1562
Counting	433	415	423	406

https://www.solvis.com.br/calculos-de-amostragem/ *(Site)*

b. Use o maior valor dos três tamanhos de amostra (n) obtidos na questão anterior, execute as ordenações n vezes para cada algoritmo, com a mesma entrada anterior. Qual a média e o intervalo de confiança, com nível de confiança de 95%, do tempo de ordenação obtidos para cada algoritmo?

Média	**433,5**	**Média**	**2111**	**Média**	**1119,5**
Desvio Padrão	**10,60660172**	**Desvio Padrão**	**356,3818177**	**Desvio Padrão**	**126,5721138**
Execuções	**22,99780893**	**Execuções**	**1094,880582**	**Execuções**	**491,0661666**
Intervalo de conf	**2,939945979**	**Intervalo de conf**	**98,7821849**	**Intervalo de conf**	**35,08335535**
Counting		**Merge**		**Quick**	

c. *Com base no intervalo de confiança da resposta anterior, podemos afirmar estatisticamente que desses três algoritmos existem uns que são melhores que os outros? Qual seria o melhor e o pior algoritmo para esta entrada? JustIfique.*

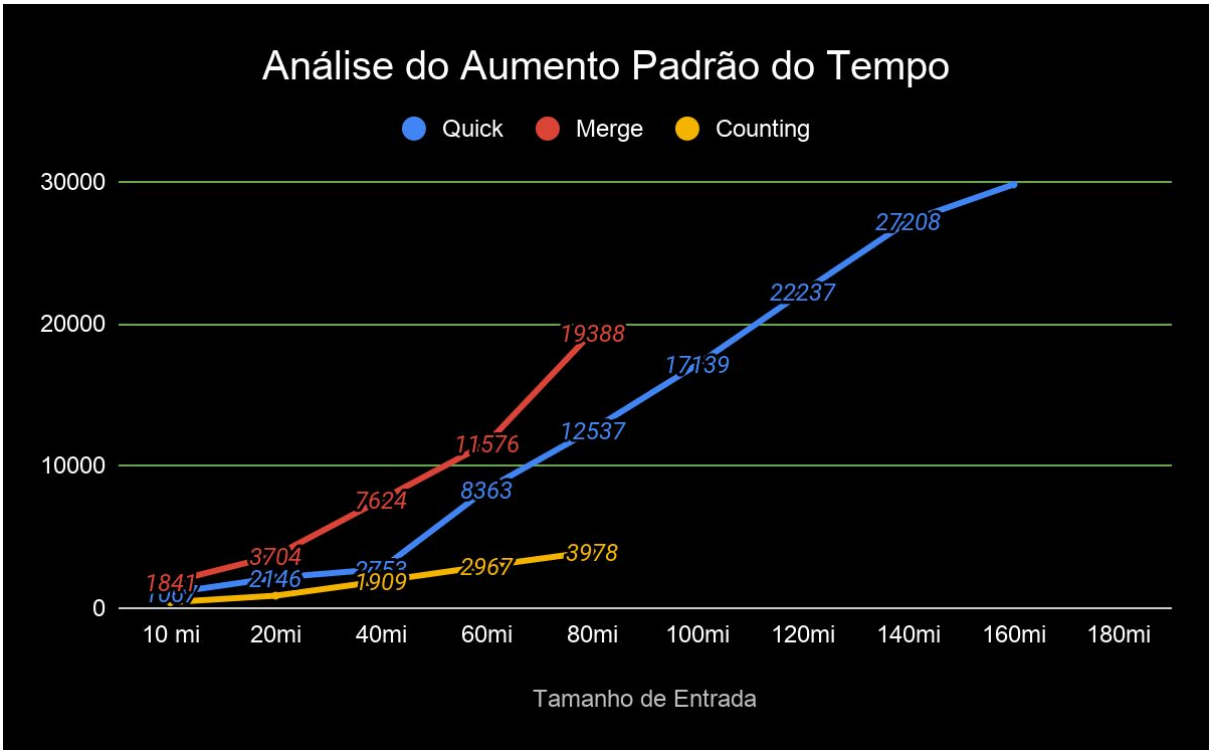
Sim. O melhor seria o Counting sort, ou ordenação por contagem. Onde ele é um algoritmo de ordenação estável, ou seja, tem complexidade O(n). E o pior de acordo com os testes seria o Merge sort, ou ordenação por mistura, que é um exemplo de algoritmo de ordenação por comparação do

tipo dividir-para-conquistar. Sua ideia básica consiste em Dividir e Conquistar. E por último ficou o Quick sort como um algoritmo intermediário, onde ele também usa divisão e conquista é um pouco diferente de como o merge sort faz. Porém todo o trabalho acontece na etapa da divisão.

2. *Queremos analisar agora como o tempo de ordenação de cada algoritmo varia ao mudarmos a entrada.*

> ***a.*** *Verifique como o tempo de ordenação aumenta ao aumentarmos o tamanho da entrada, para cada algoritmo. Por exemplo, mantenha fixo o valor máximo em 1000000 (1,0 milhão) e varie o tamanho da entrada em vários valores entre 1000000 (10 milhões) e 2000000 (200 milhões) e analise o padrão de aumento do tempo.*

Valores resultantes dos testes realizados:



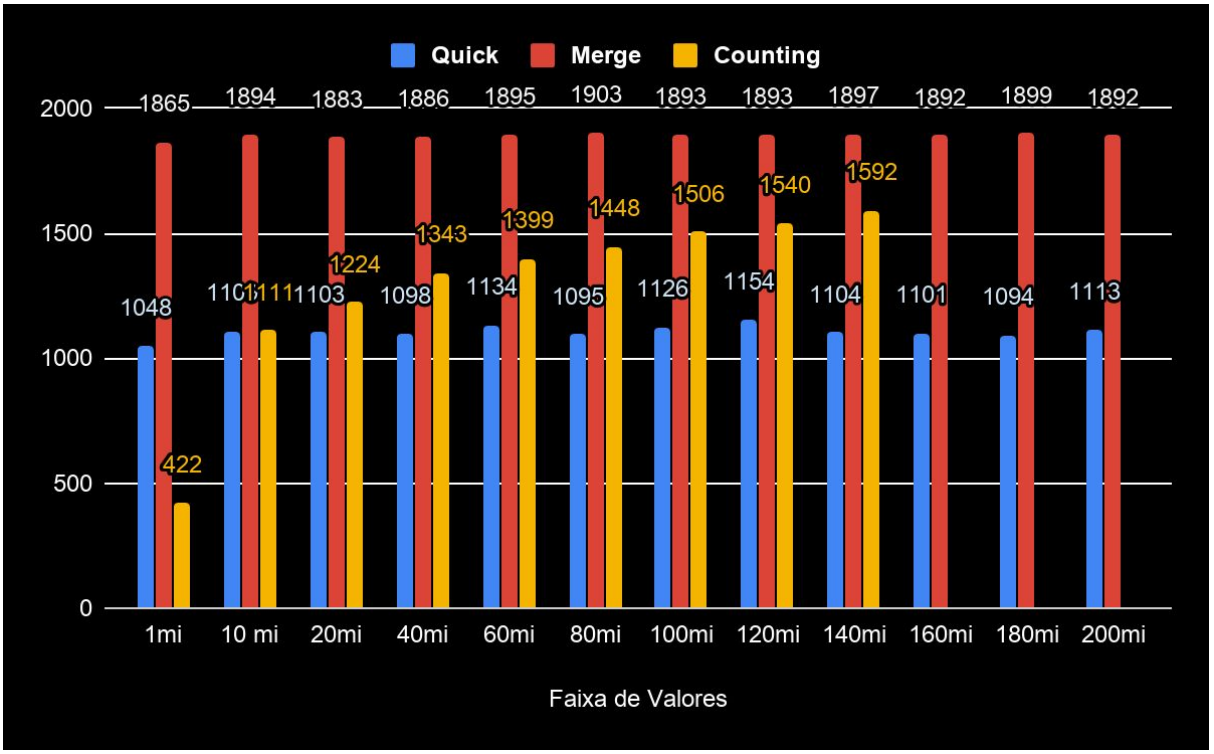
(Nota): Onde se tem “Erro”, foi porque houve algum tipo de transbordamento e/ou incapacitação de memória de dados ou seja, ocorreu um estouro de buffer. Onde tal anomalia ultrapassou os limites do buffer e segurança, sobrescrevendo a memória adjacente.

Tam_Entrada	Quick	Merge	Counting
10 mi	1067	1841	423
20mi	2146	3704	881
40mi	2753	7624	1909

60mi	8363	11576	2967
80mi	12537	19388	3978
100mi	17139	Erro!	Erro!
120mi	22237		
140mi	27208		
160mi	29798		
180mi	Erro!		

Mesmo obtendo um feedback de erro referente ao valor do qual está sendo informado, realizei alguns testes de valores intermediários entre o último valor do qual obtive “êxito” com o valor do qual ocorreu o estouro de memória, mesmo assim, o erro persistiu, resultando na incapacidade da máquina da qual foi realizadas tais testes a executar tamanha ação.

b. *Verifique como o tempo de ordenação aumenta ao aumentarmos a faixa de valores de entrada possíveis (ou seja, aumentando valor máximo da entrada). Por exemplo, mantenha o tamanho da entrada fixo em 10000000 (10 milhões) e varie o valor máximo de entrada com valores entre 1000000 (1 milhão) e 200000000 (200 milhões).*



Faixa de Valor	Quick	Merge	Counting
1mi	1048	1865	422
10 mi	1109	1894	1111
20mi	1103	1883	1224
40mi	1098	1886	1343
60mi	1134	1895	1399
80mi	1095	1903	1448
100mi	1126	1893	1506
120mi	1154	1893	1540
140mi	1104	1897	1592
160mi	1101	1892	Erro!
180mi	1094	1899	
200mi	1113	1892	

Os resultados mostram que o algoritmo Merge nesta ocasião se manteve mais instável em relação aos demais e se manteve em um padrão entre 1903 a 1865, mesmo assim seu desempenho foi o pior entre os demais. E o algoritmo Counting ao ser testado com uma entrada com o tamanho de 160mi, novamente ocorreu um estouro de buffer. E novamente o Quick se saiu melhor que os demais.

3. *Com base nas questões anteriores e em outras análises que você pode fazer, quais algoritmos são os mais adequados para diferentes situações de tamanho da entrada e do valor máximo? Justifique.*

Se for para um sistema do qual tem pequenas entradas o mais indicado pelo seu desempenho é o algoritmo Counting Sort, pois por ele ser é um algoritmo de ordenação estável, porém para grandes sistemas ele não é indicado pois ele precisa de bastante memória alocada.

No entanto, se a esquematização sistemática for complexa, robusta e de grandes requisições/entradas, o melhor e mais indica será o algoritmo Quicksort, pois ele apresentou uma maior flexibilização para sistemas cujo requer um suporte para muitos processamentos e entradas, e foi o único que suportou o maior número de entradas e obteve um melhor desempenho a longo prazo para um sistema consideravelmente grande. Pois tal algoritmo é

um método de ordenação muito rápido e eficiente, ele adota uma estratégia de dividir e conquistar rearranjar as chaves de modo que as chaves "menores" precedem as chaves "maiores". Em seguida o quick sort ordena as duas sublistas de chaves menores e maiores recursivamente até que a lista completa se encontre ordenada.

Configurações da máquina na qual foi executada as análises de desempenho dos testes:
Processador: Intel ® Core(TM) i5-8250U; CPU 1.60GHz x 8.
Memória: 3.74GB
Disco: 931.51GB