UNIVERSIDADE DA REGIÃO DE JOINVILLE

BACHARELADO EM ENGENHARIA DE SOFTWARE

Tempo de Ordenação

LUCAS FREDERICO ROEDER DE MELLO

PROFESSOR LEANDERSON ANDRE

Estrutura de Dados

Joinville – SC

2023

1. Introdução

Inúmeras aplicações dependem da classificação de dados, desde a organização de itens em listas de reprodução de música até a classificação de entradas em bancos de dados. Todo algoritmo de classificação possui um conjunto de parâmetros que podem melhorar sua adequação em situações específicas. O melhor método é determinado por vários fatores, incluindo limitações de memória, dados pré-ordenados e tamanho do pool de dados. Consequentemente, para fazer julgamentos de ordenação sábios, é preciso estar ciente dos vários algoritmos de classificação e de suas propriedades.

Examinaremos uma variedade de algoritmos de classificação neste artigo, desde os mais básicos e compreensíveis até os mais sofisticados e eficazes. Antes de passar para algoritmos de classificação mais complexos, como classificação rápida, classificação por mesclagem e classificação por seleção, primeiro examinaremos uma visão geral dos algoritmos de classificação fundamentais, como classificação por bolha e classificação por inserção. Para cada algoritmo, os recursos, o desempenho previsto e o uso sugerido serão enfatizados em cada seção.

2. Algoritmos de Ordenação

2.1.1 Merge Sort

Realizando a execução de acordo com o meu código em Java, uma lista de 100.000 elementos únicos e 10 vezes seguidas, temos aqui a saída no terminal do Visual Studio Code:

----------COMEÇOU----------
Tempo de execução com Array crescente: 0,0310 segundos
Tempo de execução com Array decrescente: 0,0209 segundos
Tempo de execução com Array aleatório: 0,0396 segundos
----------ACABOU----------

----------COMEÇOU----------
Tempo de execução com Array crescente: 0,0145 segundos
Tempo de execução com Array decrescente: 0,0110 segundos
Tempo de execução com Array aleatório: 0,0186 segundos
----------ACABOU----------

----------COMEÇOU----------
Tempo de execução com Array crescente: 0,0159 segundos
Tempo de execução com Array decrescente: 0,0100 segundos
Tempo de execução com Array aleatório: 0,0163 segundos
----------ACABOU----------

----------COMEÇOU----------
Tempo de execução com Array crescente: 0,0113 segundos
Tempo de execução com Array decrescente: 0,0137 segundos

Tempo de execução com Array aleatório: 0,0245 segundos
----------ACABOU----------

----------COMEÇOU----------
Tempo de execução com Array crescente: 0,0137 segundos
Tempo de execução com Array decrescente: 0,0134 segundos
Tempo de execução com Array aleatório: 0,0305 segundos
----------ACABOU----------

----------COMEÇOU----------
Tempo de execução com Array crescente: 0,0142 segundos
Tempo de execução com Array decrescente: 0,0147 segundos
Tempo de execução com Array aleatório: 0,0282 segundos
----------ACABOU----------

----------COMEÇOU----------
Tempo de execução com Array crescente: 0,0077 segundos
Tempo de execução com Array decrescente: 0,0083 segundos
Tempo de execução com Array aleatório: 0,0374 segundos
----------ACABOU----------

----------COMEÇOU----------
Tempo de execução com Array crescente: 0,0649 segundos
Tempo de execução com Array decrescente: 0,0196 segundos
Tempo de execução com Array aleatório: 0,0312 segundos
----------ACABOU----------

----------COMEÇOU----------
Tempo de execução com Array crescente: 0,0098 segundos
Tempo de execução com Array decrescente: 0,0180 segundos
Tempo de execução com Array aleatório: 0,0496 segundos
----------ACABOU----------

----------COMEÇOU----------
Tempo de execução com Array crescente: 0,0140 segundos
Tempo de execução com Array decrescente: 0,0762 segundos
Tempo de execução com Array aleatório: 0,0397 segundos
----------ACABOU----------

Média de tempo de execução com Array crescente: 0.0197113 segundos
Média de tempo de execução com Array decrescente: 0.02057395 segundos
Média de tempo de execução com Array aleatório: 0.031548 segundos
Desvio padrão do tempo de execução com Array crescente: 0.016181894340342236 segundos
Desvio padrão do tempo de execução com Array decrescente: 0.018949678233376416 segundos
Desvio padrão do tempo de execução com Array aleatório: 0.00976931498888228 segundos

2.1.1.1 Resumo

Considerando, é claro, que seu melhor desempenho está em arrays que já estariam pré-encomendados, este está inquestionavelmente entre os que apresentam melhor desempenho em comparação aos demais que virão, independentemente das circunstâncias.

2.1.2 Quick Sort

Realizando a execução de acordo com o meu código em Java, uma lista de 100.000 elementos únicos e 10 vezes seguidas, temos aqui a saída no terminal do Visual Studio Code:

```
----------COMEÇOU----------
Tempo de execução com Array crescente: 1,0978 segundos
Tempo de execução com Array decrescente: 3,5892 segundos
Tempo de execução com Array aleatório: 0,0074 segundos
----------ACABOU----------

----------COMEÇOU----------
Tempo de execução com Array crescente: 2,2808 segundos
Tempo de execução com Array decrescente: 4,0093 segundos
Tempo de execução com Array aleatório: 0,0074 segundos
----------ACABOU----------

----------COMEÇOU----------
Tempo de execução com Array crescente: 2,4030 segundos
Tempo de execução com Array decrescente: 3,7665 segundos
Tempo de execução com Array aleatório: 0,0075 segundos
----------ACABOU----------

----------COMEÇOU----------
Tempo de execução com Array crescente: 2,8543 segundos
Tempo de execução com Array decrescente: 3,7437 segundos
Tempo de execução com Array aleatório: 0,0076 segundos
----------ACABOU----------

----------COMEÇOU----------
Tempo de execução com Array crescente: 2,4174 segundos
Tempo de execução com Array decrescente: 3,4934 segundos
Tempo de execução com Array aleatório: 0,0090 segundos
----------ACABOU----------

----------COMEÇOU----------
Tempo de execução com Array crescente: 2,4250 segundos
Tempo de execução com Array decrescente: 3,6907 segundos
Tempo de execução com Array aleatório: 0,0076 segundos
----------ACABOU----------
```

----------COMEÇOU----------
Tempo de execução com Array crescente: 2,3754 segundos
Tempo de execução com Array decrescente: 3,7917 segundos
Tempo de execução com Array aleatório: 0,0077 segundos
----------ACABOU----------

----------COMEÇOU----------
Tempo de execução com Array crescente: 2,3988 segundos
Tempo de execução com Array decrescente: 3,7594 segundos
Tempo de execução com Array aleatório: 0,0072 segundos
----------ACABOU----------

----------COMEÇOU----------
Tempo de execução com Array crescente: 2,2629 segundos
Tempo de execução com Array decrescente: 3,3939 segundos
Tempo de execução com Array aleatório: 0,0075 segundos
----------ACABOU----------

----------COMEÇOU----------
Tempo de execução com Array crescente: 2,4502 segundos
Tempo de execução com Array decrescente: 3,4503 segundos
Tempo de execução com Array aleatório: 0,0076 segundos
----------ACABOU----------

Média de tempo de execução com Array crescente: 2.29656389 segundos
Média de tempo de execução com Array decrescente: 3.66882733 segundos
Média de tempo de execução com Array aleatório: 0.007634410000000001 segundos
Desvio padrão do tempo de execução com Array crescente: 0.42802254929798833 segundos
Desvio padrão do tempo de execução com Array decrescente: 0.17756046407007414 segundos
Desvio padrão do tempo de execução com Array aleatório: 4.74780687159872E-4 segundos

2.1.2.1 Resumo

De acordo com as descobertas, esse algoritmo foi o melhor para classificar em quase todos os aspectos. Semelhante ao Merge Sort, ele tem um desempenho melhor em todos os cenários e é superior aos outros; No entanto, diferentemente do Merge Sort, seu melhor desempenho foi ao ordenar decisões decrescentes.

2.1.3 Bubble Sort

Realizando a execução de acordo com o meu código em Java, uma lista de 100.000 elementos únicos e 10 vezes seguidas, temos aqui a saída no terminal do Visual Studio Code:

----------COMEÇOU----------
Tempo de execução com Array crescente: 0,0016 segundos
Tempo de execução com Array decrescente: 10,2852 segundos
Tempo de execução com Array aleatório: 25,7752 segundos
----------ACABOU----------

----------COMEÇOU----------
Tempo de execução com Array crescente: 0,0001 segundos
Tempo de execução com Array decrescente: 14,0254 segundos
Tempo de execução com Array aleatório: 22,1753 segundos
----------ACABOU----------

----------COMEÇOU----------
Tempo de execução com Array crescente: 0,0014 segundos
Tempo de execução com Array decrescente: 13,2582 segundos
Tempo de execução com Array aleatório: 23,1748 segundos
----------ACABOU----------

----------COMEÇOU----------
Tempo de execução com Array crescente: 0,0001 segundos
Tempo de execução com Array decrescente: 13,6881 segundos
Tempo de execução com Array aleatório: 22,3902 segundos
----------ACABOU----------

----------COMEÇOU----------
Tempo de execução com Array crescente: 0,0001 segundos
Tempo de execução com Array decrescente: 13,5301 segundos
Tempo de execução com Array aleatório: 22,2974 segundos
----------ACABOU----------

----------COMEÇOU----------
Tempo de execução com Array crescente: 0,0001 segundos
Tempo de execução com Array decrescente: 13,5035 segundos
Tempo de execução com Array aleatório: 22,7040 segundos
----------ACABOU----------

----------COMEÇOU----------
Tempo de execução com Array crescente: 0,0001 segundos
Tempo de execução com Array decrescente: 13,5171 segundos
Tempo de execução com Array aleatório: 21,1890 segundos
----------ACABOU----------

----------COMEÇOU----------
Tempo de execução com Array crescente: 0,0001 segundos
Tempo de execução com Array decrescente: 13,7700 segundos
Tempo de execução com Array aleatório: 26,4443 segundos
----------ACABOU----------

----------COMEÇOU----------
Tempo de execução com Array crescente: 0,0001 segundos
Tempo de execução com Array decrescente: 13,1728 segundos
Tempo de execução com Array aleatório: 21,9375 segundos
----------ACABOU----------

----------COMEÇOU----------
Tempo de execução com Array crescente: 0,0001 segundos
Tempo de execução com Array decrescente: 13,8541 segundos
Tempo de execução com Array aleatório: 22,4032 segundos
----------ACABOU----------

Média de tempo de execução com Array crescente: 4.035299999999999E-4 segundos
Média de tempo de execução com Array decrescente: 13.260460089999999 segundos
Média de tempo de execução com Array aleatório: 23.04908345 segundos
Desvio padrão do tempo de execução com Array crescente: 5.670379794158412E-4 segundos
Desvio padrão do tempo de execução com Array decrescente: 1.021632081176184 segundos
Desvio padrão do tempo de execução com Array aleatório: 1.6115109985306515 segundos

2.1.3.1 Resumo

Embora seja um dos mais rápidos para a situação que se está a tornar mais organizada, é de longe o pior de todos quando se examinam os cenários que se estão a tornar menos ordenados e imprevisíveis. É a velha história de ser excelente em fazer o que já foi feito, mas péssimo em realizar o que precisa ser feito.

2.1.4 Insert Sort

Realizando a execução de acordo com o meu código em Java, uma lista de 100.000 elementos únicos e 10 vezes seguidas, temos aqui a saída no terminal do Visual Studio Code:

----------COMEÇOU----------
Tempo de execução com Array crescente: 0,0026 segundos
Tempo de execução com Array decrescente: 6,0168 segundos
Tempo de execução com Array aleatório: 1,7341 segundos
----------ACABOU----------

----------COMEÇOU----------
Tempo de execução com Array crescente: 0,0001 segundos
Tempo de execução com Array decrescente: 3,8937 segundos
Tempo de execução com Array aleatório: 1,7501 segundos
----------ACABOU----------

----------COMEÇOU----------
Tempo de execução com Array crescente: 0,0002 segundos
Tempo de execução com Array decrescente: 4,1079 segundos
Tempo de execução com Array aleatório: 1,5041 segundos
----------ACABOU----------

----------COMEÇOU----------
Tempo de execução com Array crescente: 0,0016 segundos
Tempo de execução com Array decrescente: 3,5218 segundos
Tempo de execução com Array aleatório: 1,5187 segundos
----------ACABOU----------

----------COMEÇOU----------
Tempo de execução com Array crescente: 0,0002 segundos
Tempo de execução com Array decrescente: 4,0755 segundos
Tempo de execução com Array aleatório: 2,3863 segundos
----------ACABOU----------

----------COMEÇOU----------
Tempo de execução com Array crescente: 0,0003 segundos
Tempo de execução com Array decrescente: 4,7964 segundos
Tempo de execução com Array aleatório: 1,9149 segundos
----------ACABOU----------

----------COMEÇOU----------
Tempo de execução com Array crescente: 0,0002 segundos
Tempo de execução com Array decrescente: 3,2934 segundos
Tempo de execução com Array aleatório: 1,6799 segundos
----------ACABOU----------

----------COMEÇOU----------
Tempo de execução com Array crescente: 0,0002 segundos
Tempo de execução com Array decrescente: 3,0685 segundos
Tempo de execução com Array aleatório: 1,8323 segundos
----------ACABOU----------

----------COMEÇOU----------
Tempo de execução com Array crescente: 0,0001 segundos
Tempo de execução com Array decrescente: 3,1163 segundos
Tempo de execução com Array aleatório: 1,6400 segundos
----------ACABOU----------

----------COMEÇOU----------
Tempo de execução com Array crescente: 0,0001 segundos
Tempo de execução com Array decrescente: 3,0619 segundos
Tempo de execução com Array aleatório: 1,6626 segundos
----------ACABOU----------

Média de tempo de execução com Array crescente: 5.5568E-4 segundos
Média de tempo de execução com Array decrescente: 3.8952181399999994 segundos
Média de tempo de execução com Array aleatório: 1.7623169400000003 segundos
Desvio padrão do tempo de execução com Array crescente: 7.976414053946797E-4 segundos
Desvio padrão do tempo de execução com Array decrescente: 0.8875341084416736 segundos
Desvio padrão do tempo de execução com Array aleatório: 0.24004375854634583 segundos

2.1.4.1 Resumo

Geralmente é bastante semelhante ao que foi descrito anteriormente sobre Bubble Sort em termos de desempenho em todos os seus cenários, ou seja, ele tem um bom desempenho com matrizes pré-classificadas e um desempenho ruim com matrizes decrescentes e aleatórias. Com base nos resultados, ainda pode ser um pouco superior ao Bubble Sort.

2.1.5 Selection Sort

Realizando a execução de acordo com o meu código em Java, uma lista de 100.000 elementos únicos e 10 vezes seguidas, temos aqui a saída no terminal do Visual Studio Code:

----------COMEÇOU----------
Tempo de execução com Array crescente: 1,3522 segundos
Tempo de execução com Array decrescente: 2,8380 segundos
Tempo de execução com Array aleatório: 2,5476 segundos
----------ACABOU----------

----------COMEÇOU----------
Tempo de execução com Array crescente: 2,4032 segundos
Tempo de execução com Array decrescente: 2,7186 segundos
Tempo de execução com Array aleatório: 2,4027 segundos
----------ACABOU----------

----------COMEÇOU----------
Tempo de execução com Array crescente: 2,5105 segundos
Tempo de execução com Array decrescente: 2,5296 segundos
Tempo de execução com Array aleatório: 2,5622 segundos
----------ACABOU----------

----------COMEÇOU----------
Tempo de execução com Array crescente: 2,5660 segundos
Tempo de execução com Array decrescente: 2,6732 segundos
Tempo de execução com Array aleatório: 2,4278 segundos

----------ACABOU----------

----------COMEÇOU----------
Tempo de execução com Array crescente: 3,2650 segundos
Tempo de execução com Array decrescente: 2,6702 segundos
Tempo de execução com Array aleatório: 2,5377 segundos
----------ACABOU----------

----------COMEÇOU----------
Tempo de execução com Array crescente: 2,5317 segundos
Tempo de execução com Array decrescente: 3,0692 segundos
Tempo de execução com Array aleatório: 2,5597 segundos
----------ACABOU----------

----------COMEÇOU----------
Tempo de execução com Array crescente: 2,5821 segundos
Tempo de execução com Array decrescente: 2,7959 segundos
Tempo de execução com Array aleatório: 3,2578 segundos
----------ACABOU----------

----------COMEÇOU----------
Tempo de execução com Array crescente: 3,5978 segundos
Tempo de execução com Array decrescente: 2,7094 segundos
Tempo de execução com Array aleatório: 2,4310 segundos
----------ACABOU----------

----------COMEÇOU----------
Tempo de execução com Array crescente: 2,5985 segundos
Tempo de execução com Array decrescente: 2,6245 segundos
Tempo de execução com Array aleatório: 2,5719 segundos
----------ACABOU----------

----------COMEÇOU----------
Tempo de execução com Array crescente: 2,5503 segundos
Tempo de execução com Array decrescente: 3,2310 segundos
Tempo de execução com Array aleatório: 2,7137 segundos
----------ACABOU----------

Média de tempo de execução com Array crescente: 2.59572234 segundos
Média de tempo de execução com Array decrescente: 2.78594856 segundos
Média de tempo de execução com Array aleatório: 2.6012097499999998 segundos
Desvio padrão do tempo de execução com Array crescente: 0.5522744616784868 segundos
Desvio padrão do tempo de execução com Array decrescente: 0.20234390959415208 segundos
Desvio padrão do tempo de execução com Array aleatório: 0.23519568056865012 segundos

2.1.5.1 Resumo

Dado o tempo médio coletado para teste, esse algoritmo pode ser superior à classificação do Bobble Sort e Insertion Sort no cenário com matrizes decrescentes, mas perde para ambos no cenário crescente. Foi o único a apresentar certa igualdade entre todos os cenários testados, mas isso nem sempre implica bons resultados. No entanto, comparado ao Merge Sort e ao Quick Sort, ainda é muito pior.