Relatório Projeto 4.1 AED

Filipe David Amado Mendes
Nº Estudante: 2020218797
Login* no *Mooshak: 2020218797
Turma: PL3
Docente Responsável: Prof. Doutor Ivo Gonçalves

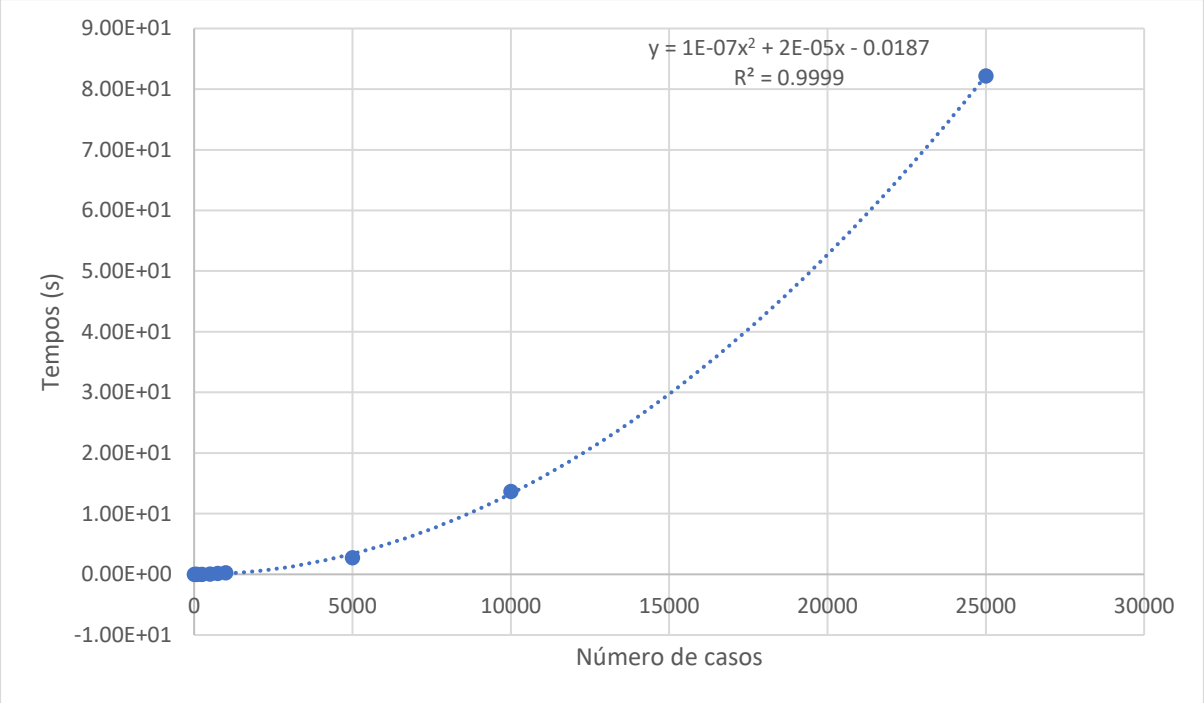
Solução 1: Sem algoritmos de ordenamento

Tabela

Num de casos	Tempos (s)
10	6.51E-05
50	0.000686884
100	0.002980709
250	0.013731956
500	0.045454979
750	0.124615431
1000	0.263025761
5000	2.756973743
10000	13.66617322
25000	82.18584204

Gráfico

Equação: $y = 1E\text{-}07x^2 + 2E\text{-}05x - 0.0187$
$R^2 = 0.9999$



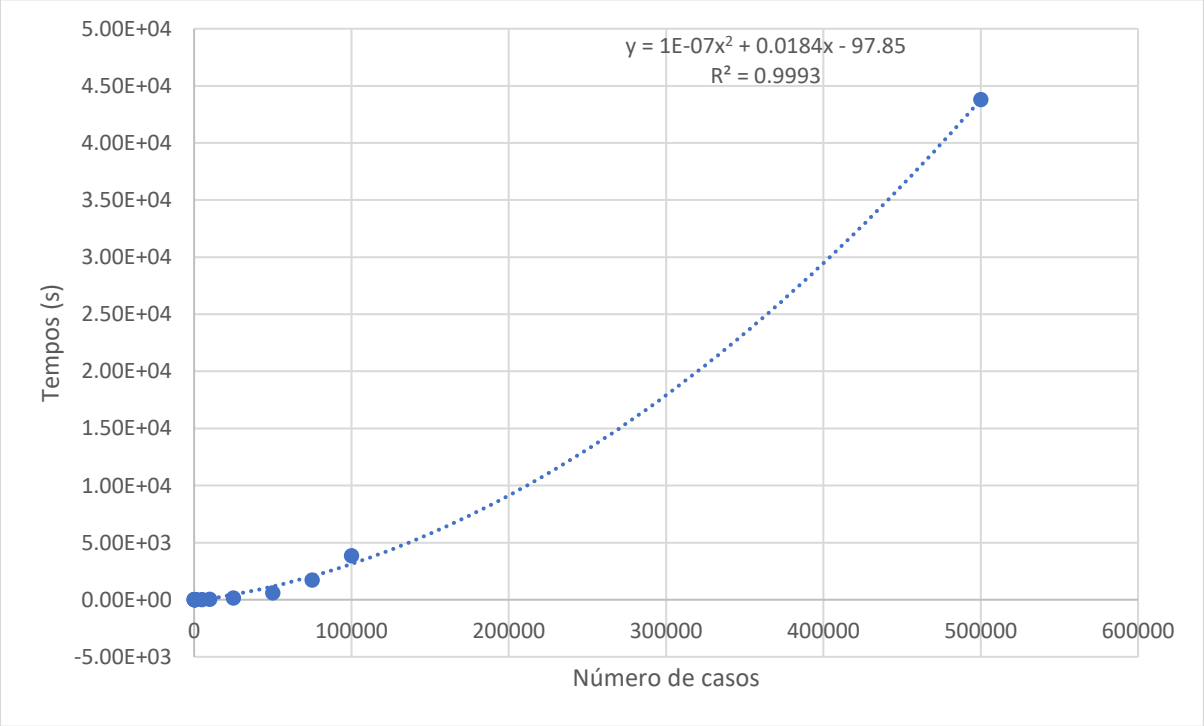
Solução 2: Algoritmo de ordenamento elementar

Tabela

Num de casos	Tempos (s)
10	2.98E-05
50	0.000412703
100	0.001761198
250	0.010929108
500	0.052087307
750	0.162031174
1000	0.359715939
5000	4.215124846
10000	21.78264141
25000	128.078934
50000	577.449321
75000	1699.922995
100000	3850.667918
500000	43780.49742

Gráfico

Equação: $y = 1E\text{-}07x^2 + 0.0184x - 97.85$
$R^2 = 0.9993$



Introdução

Objetivo deste projeto 4 de AED é ver a diferença de eficiência de um programa implementando diferentes algoritmos de ordenamento de dados. Para isso foi criado um programa com quatro implementações diferentes.
Neste subprojeto 1 serão comparadas duas implementações diferentes. Uma recorrendo à força bruta, sem recurso a qualquer algoritmo de ordenamento e a segunda com a implementação de um algoritmo elementar de ordenamento, tendo sido escolhido o insertion sort.

Solução 1

Na solução 1, o programa foi implementado sem nenhum tipo de ordenamento, recorrendo apenas a iteratividade para fazer as comparações entre cada um dos números da matriz. Visto que este método obriga o programa a comparar cada elemento da matriz sempre que se acede à mesma, torna-se extremamente ineficiente para um grande número de casos. Por essa razão apenas se fizeram testes até 25000 números.
Além da solução de força bruta, para encontrar a mediana, visto que o array não está ordenado, foi implementado um algoritmo para achar a mediana de qualquer vetor desordenado.
Os resultados obtidos estão de acordo com o previsto. Os tempos de execução aumentam exponencialmente com o aumento do número de casos. Como podemos ver através do gráfico da regressão linear, o algoritmo tem complexidade quadrática, $O(n^2)$, com um erro muito pequeno, $R^2 = 0.9999$. Tiramos dos resultados que esta solução é bastante ineficiente e difícil de implementar, tornando o processo de encontrar a mediana bastante complicado.

Solução 2

Na solução 2, por outro lado, foi implementado um algoritmo de insertion sort. Um algoritmo de ordenação com complexidade $O(n^2)$, que compara um elemento do array com os adjacentes, e assim sucessivamente, até inserir o elemento na posição certa do vetor, repetindo este processo até que todo o vetor esteja ordenado.
Nesta solução, foram realizados os mesmos testes que na primeira solução e mais quatro testes nas dezenas e centenas de milhares de casos.
Os resultados obtidos estão dentro das espectativas teóricas. Para um número pequeno de casos na matriz o algoritmo implementado mostrou-se bastante eficiente, porém perde eficiência exponencialmente à medida que o número de elementos da matriz aumenta, sendo inclusive menos eficiente que a primeira solução a partir dos milhares de casos.
O gráfico comprova a complexidade quadrática teórica ($O(n^2)$) do algoritmo de ordenamento por inserção, com um erro pequeno, $R^2 = 0.9993$ e mostra a sua perda de eficiência exponencial.

Conclusão

Podemos concluir através deste subprojeto que algoritmos de ordenamento elementares são bastante eficientes para uma lista pequena de elementos, mas que à medida que o número de elementos da matriz aumenta a sua eficiência baixa bastante. Vemos também na primeira solução, que apesar de usar apenas força bruta e de ser evidentemente mais lenta que a segunda solução para um pequeno número de casos, que o algoritmo de procura da mediana é bastante eficiente, mostrando-se superior ao insertion sort a partir dos milhares de casos.
Assim, concluímos que nenhum dos algoritmos é particularmente eficiente, sendo apenas aconselhada a sua implementação para um pequeno número de casos e percebemos a importância de um bom algoritmo de ordenamento de dados.