UNIVERSIDADE FEDERAL DE PERNAMBUCO

**SISTEMAS DE INFORMAÇÃO**

**ALGORITMOS E ESTRUTURA DE DADOS**

**DISCENTE: JOÃO RAFAEL SANTOS CAMELO**

Centro de Informática UFPE

# PROJETO 3

## QUESTÃO 1
## CONTA SOMAS

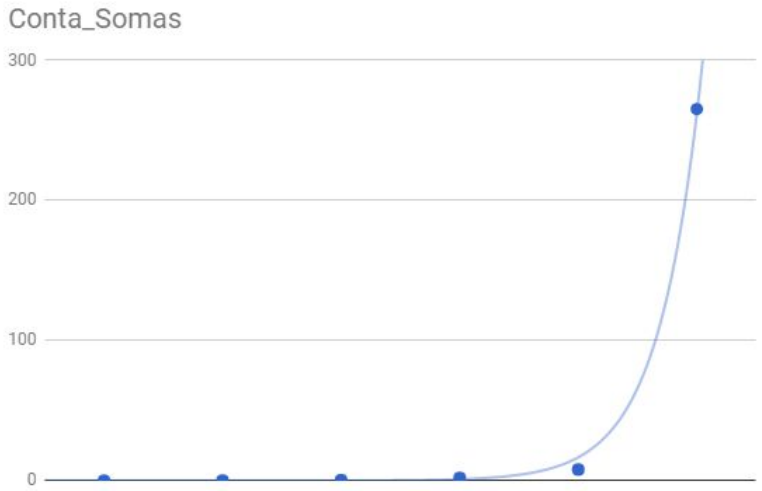
Foram utilizados um Merge Sort então dois loops encadeados para contar as triplas que somadas resultam em 0, com isso uma complexidade de tempo de $O(n^2 \log n)$ foi alcançada.

RESULTADOS DO CONTA_SOMAS ANTERIOR COM $O(n^3)$

Tempo gasto com 50 elementos foi 0.0865416532464798124 segundos
Tempo gasto com 100 elementos foi 0.063846546954635461 segundos
Tempo gasto com 250 elementos foi 0.092205151654696524 segundos
Tempo gasto com 500 elementos foi 7.623654359801042678 segundos
Tempo gasto com 1000 elementos foi 61.2028403287910223 segundos
Tempo gasto com 1500 elementos foi 204.746549015570180 segundos

RESULTADOS DO CONTA_SOMAS COM $O(n^2 \log n)$

Tempo gasto com 50 elementos foi 0.002767453324128619 segundos
Tempo gasto com 100 elementos foi 0.01294074214718516 segundos
Tempo gasto com 250 elementos foi 0.08511639574220304 segundos
Tempo gasto com 500 elementos foi 0.40159798695541798 segundos
Tempo gasto com 1000 elementos foi 1.9821585592580047 segundos
Tempo gasto com 1500 elementos foi 7.9846549846549012 segundos
Tempo gasto com 10000 elementos foi 265.1264235344617 segundos

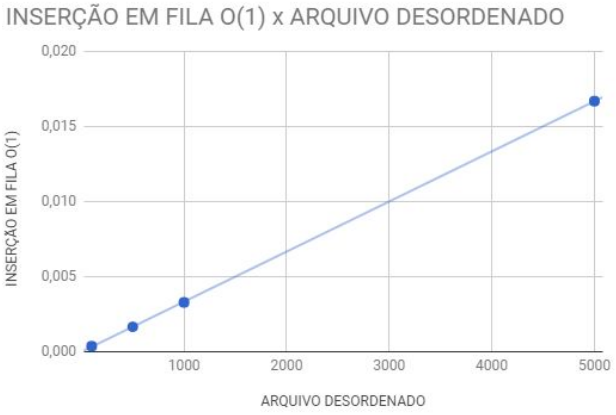


QUESTÃO 2
LISTA DUPLAMENTE ENCADEADA

Uma lista duplamente encadeada foi implementada, com inserção em fila, ordenada crescente, ordenada decrescente, busca por placa , remoção e outras funções que funcionam diretamente com o Projeto 2.

Leitura do arquivo - Inserção em fila - O(1)

| | |
|---|---|
| 1 elemento | 0.0000218142669155204 |
| 100 elementos | 0.000345499492176698 |
| 500 elementos | 0.00163959879713565 |
| 1000 elementos | 0.00326540327842767 |
| 5000 elementos | 0.0166770070569153 |
| 20000 elementos | 0.084927753072764 |




Leitura do arquivo desordenado - Inserção ordenada crescente - O(n)

| | |
|---|---|
| 1 elemento | 3.4325170359394974e-05 |
| 100 elementos | 0.005051574370455114 |
| 500 elementos | 0.0967649008130902 |
| 1000 elementos | 0.38944921254209675 |
| 5000 elementos | 10.705686686260886 |
| 20000 elementos | 200.7585660547804 |

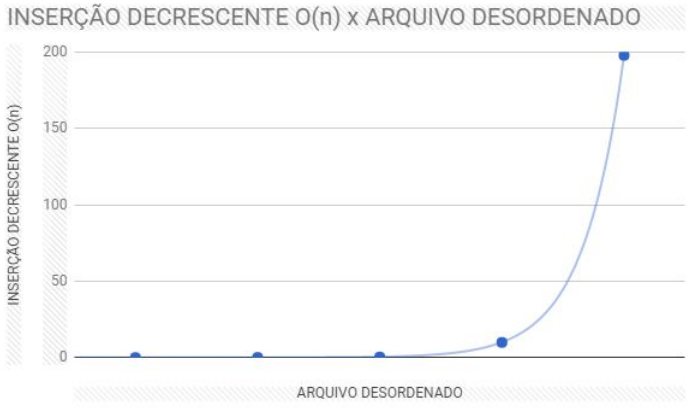
Leitura de um arquivo crescente - Inserção ordenada crescente - O(n)


| | |
|---|---|
| 1 elemento | 2.63054395157746e-05 |
| 100 elementos | 0.008487353820351812 |
| 500 elementos | 0.17830596819094924 |
| 1000 elementos | 0.7241454421334866 |
| 5000 elementos | 19.283300280441647 |
| 20000 elementos | 355.9306979987976 |

Busca por placa

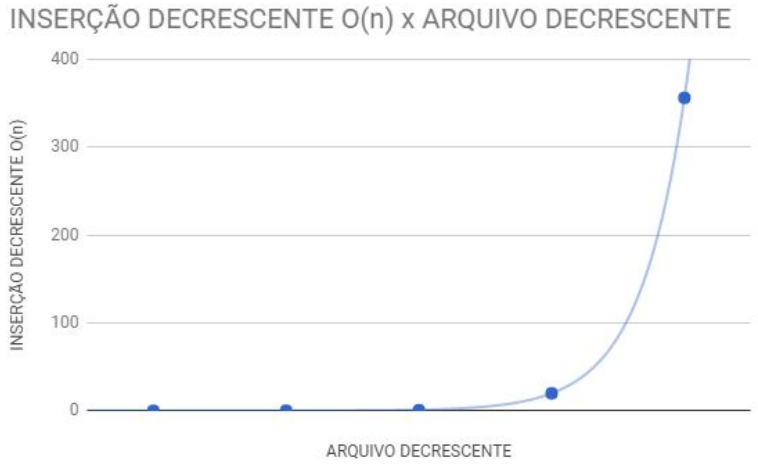
| | |
|---|---|
| Último item do arquivo desordenado | |
| Tempo de Operação: | 0.018794915736009443 |
| Último item do arquivo crescente | |
| Tempo de Operação: | 0.04887261943792964 |
| Último item do arquivo decrescente | |
| Tempo de Operação: | 3.240060232201358e-05 |
| | |
| 500 placas aleatórias | |
| Tempo de Operação: | 11.864606223610394 |

Leitura do arquivo desordenado - Inserção ordenada decrescente - O(n)



| | |
|---|---|
| 1 elemento | 0,0000279094297142 |
| 100 elementos | 0.004102365372261829 |
| 500 elementos | 0.09446572048352664 |
| 1000 elementos | 0.36613610947682673 |
| 5000 elementos | 10.242450820056035 |
| 20000 elementos | 198.95242532944349 |

Leitura de um arquivo decrescente - Inserção ordenada decrescente - O(n)




| | |
|---|---|
| 1 elemento | 2.8230227826497867e-05 |
| 100 elementos | 0.007906709362714537 |
| 500 elementos | 0.1857202525578714 |
| 1000 elementos | 0.767908069546479 |
| 5000 elementos | 20.102604046033207 |
| 20000 elementos | 367.69668031769277 |

Busca por placa

| | |
|---|---|
| Último item do arquivo desordenado | |
| Tempo de Operação: | 0.028871823858708012 |
| Último item do arquivo crescente | |
| Tempo de Operação: | 1.9247882505624148e-05 |
| Último item do arquivo decrescente | |
| Tempo de Operação: | 0.046802830465253464 |
| | |
| 500 placas aleatórias | |
| Tempo de Operação: | 11.804158248391445 |

QUESTÃO 3
ÁRVORE BINÁRIA

Uma árvore binária foi implementada, com todas as funções da anterior e foi implementada no Projeto 2, com uma inserção binária, busca e remoção.

Leitura do arquivo
Inserção binária - O(log n)


| | |
|---|---|
| 1 elemento | 3.4004483459064376e-05 |
| 100 elementos | 0.0026427257993940786 |
| 500 elementos | 0.01667759515160225 |
| 1000 elementos | 0.03742514201683799 |
| 5000 elementos | 0.22431410390486958 |
| 20000 elementos | 1.0870370417895854 |

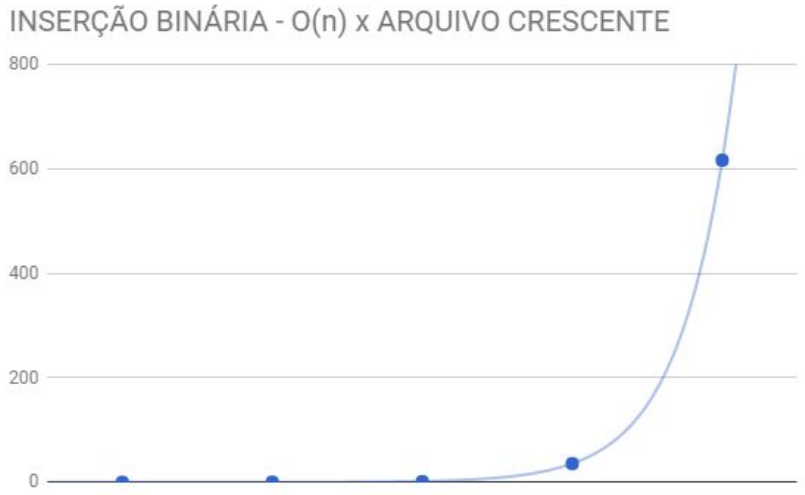
Busca por placa
Último item do arquivo desordenado

| | |
|---|---|
| Tempo de Operação: | 5.613947740878977e-05 |
| Último item do arquivo crescente | |
| Tempo de Operação: | 6.351780872559942e-05 |
| Último item do arquivo decrescente | |
| Tempo de Operação: | 3.464607748671078e-05 |

| | |
|---|---|
| 500 placas aleatórias | |
| Tempo de Operação: | 0.013287091513122462 |

Teste dos piores casos da Árvore Binária

Leitura de um arquivo crescente
Inserção binária - O(n)



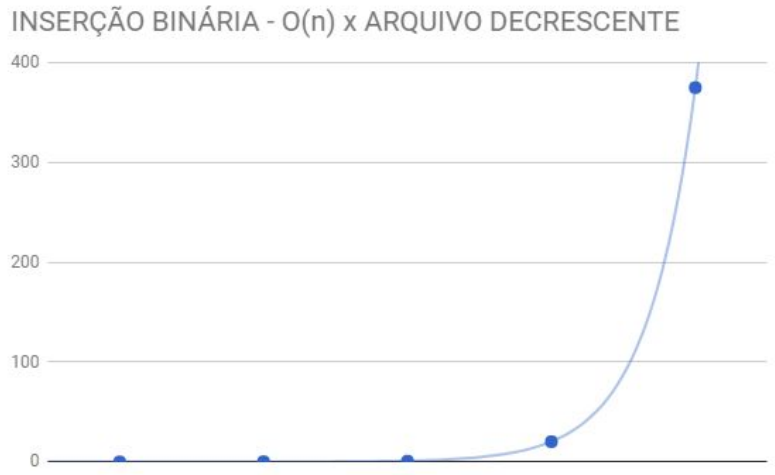

| | |
|---|---|
| 1 elemento | 3.6891656582938026e-05 |
| 100 elementos | 0.014384858894225472 |
| 500 elementos | 0.35033247402506573 |
| 1000 elementos | 1.3761274411048763 |
| 5000 elementos | 35.72158700849086 |
| 20000 elementos | 616.0965242926746 |

Busca por placa
Último item do arquivo desordenado
Tempo de Operação: 0.015243953297158441
Último item do arquivo crescente
Tempo de Operação: 0.03900667129471458
Último item do arquivo decrescente
Tempo de Operação: 1.5398256664411747e-05

500 placas aleatórias
Tempo de Operação: 9.953073171232404

Pop
Item anterior - O(n)
Tempo de Operação: 0.03978588724112342

Leitura de um arquivo decrescente
Inserção binária - O(n)




| | |
|---|---|
| 1 elemento | 3.721245366250514e-05 |
| 100 elementos | 0.00849278014243282 |
| 500 elementos | 0.18733390734610111 |
| 1000 elementos | 0.7665707697460675 |
| 5000 elementos | 20.232600290770392 |
| 20000 elementos | 374.7265519838729 |

Busca por placa
Último item do arquivo desordenado
Tempo de Operação: 0.01477270248392415
Último item do arquivo crescente
Tempo de Operação: 1.0586301414150512e-05
Último item do arquivo decrescente
Tempo de Operação: 0.02430871451508665

500 placas aleatórias
Tempo de Operação: 5.6700426275072005

QUESTÃO 4
QUICKSORT

Dois tipos de QuickSort, um utilizando o método de Dividir e Conquistar por meio da recursão e outro utilizando o Particionamento e uma solução iterativa, ambos testados com 500 elementos de 3 listas, uma desordenada, outra ordenada crescente e por último uma ordenada decrescentemente.
Na lista desordenada, o quicksort tem o seu melhor caso em listas aleatórias, ficando com uma complexidade de O(n log n), porém, no caso da lista já estar ordenada, resulta no seu pior caso, que é $O(n^2)$

QUICKSORT DIVIDIR E CONQUISTAR

QuickSort na lista desordenada - O(n log n)
Tempo de Operação: 0.007954482753310945

QuickSort na lista crescente - $O(n^2)$
Tempo de Operação: 0.16906034705104137

QuickSort na lista decrescente - $O(n^2)$
Tempo de Operação: 0.16705729049709422

QUICKSORT PARTICIONADO

QuickSort na lista desordenada - O(n log n)
Tempo de Operação: 0.00923029247705337

QuickSort na lista crescente - $O(n^2)$
Tempo de Operação: 0.19499903119301842

QuickSort na lista decrescente - $O(n^2)$
Tempo de Operação: 0.1872607656269849