# Relatório: Insertion, Bubble e Selection sort

Os algoritmos originais do Bubble, Insertion e Selection sort, foram modificados para realização deste relatório. Em todos os 3 algoritmos, foram adicionadas linhas de código adequadas para contabilizar e mostrar os dados de: Número de trocas realizadas, número de comparações e o tempo de execução de cada código.

Abaixo trago uma das saídas que coletei durante os testes.

```
Para array de tamanho 10

Insertion Sort:
Tempo de execução: 1700 nanossegundos
Número de trocas: 23
Número de comparações: 28
Array Ordenado: [20, 98, 310, 433, 443, 547, 581, 640, 713, 858]

Bubble Sort:
Tempo de execução: 2300 nanossegundos
Número de trocas: 13
Número de comparações: 45
Array Ordenado: [52, 180, 196, 321, 417, 484, 637, 846, 861, 988]

Selection Sort:
Tempo de execução: 2400 nanossegundos
Número de trocas: 9
Número de comparações: 45
Array Ordenado: [44, 72, 344, 418, 483, 578, 601, 664, 682, 847]

================================================================================

Para array de tamanho 100

Insertion Sort:
Tempo de execução: 52400 nanossegundos
Número de trocas: 2813
Número de comparações: 2908
Array Ordenado: [5, 10, 12, 36, 47, 88, 94, 101, 101, 111, 111, 113, 115, 123,
132, 140, 167, 183, 185, 218, 223, 227, 231, 232, 237, 245, 245, 254, 256, 283,
297, 299, 304, 309, 311, 316, 319, 329, 331, 358, 361, 371, 385, 395, 396, 396,
419, 460, 467, 487, 497, 499, 508, 508, 530, 531, 532, 564, 574, 579, 594, 606,
614, 621, 623, 656, 672, 673, 680, 682, 687, 698, 707, 708, 720, 723, 726, 732,
736, 743, 767, 818, 835, 849, 861, 867, 870, 878, 880, 880, 885, 888, 901, 903,
931, 949, 952, 962, 978, 990]

Bubble Sort:
Tempo de execução: 229600 nanossegundos
Número de trocas: 2387
Número de comparações: 4950
Array Ordenado: [1, 13, 45, 48, 66, 77, 87, 127, 136, 144, 147, 149, 156, 185,
190, 196, 199, 201, 212, 215, 244, 252, 253, 254, 266, 269, 285, 287, 312, 313,
314, 326, 337, 367, 373, 384, 387, 395, 398, 403, 406, 428, 442, 470, 479, 491,
```

```
504, 508, 513, 517, 517, 517, 533, 558, 618, 629, 634, 638, 643, 651, 661, 662,
663, 667, 669, 676, 687, 714, 719, 722, 724, 726, 730, 743, 746, 748, 764, 777,
786, 789, 805, 854, 865, 867, 888, 890, 902, 905, 906, 910, 922, 932, 933, 952,
971, 981, 985, 990, 994, 995]

Selection Sort:
Tempo de execução: 84700 nanossegundos
Número de trocas: 108
Número de comparações: 4995
Array Ordenado: [0, 22, 25, 27, 31, 41, 70, 71, 74, 86, 89, 117, 131, 132, 155,
162, 169, 182, 189, 193, 216, 240, 246, 251, 257, 258, 264, 287, 301, 310, 311,
311, 326, 328, 341, 349, 354, 358, 372, 389, 389, 390, 391, 406, 410, 413, 425,
434, 441, 453, 502, 514, 520, 524, 530, 538, 550, 558, 558, 568, 575, 586, 591,
592, 596, 604, 605, 646, 651, 651, 680, 685, 687, 688, 694, 698, 711, 720, 738,
745, 780, 800, 809, 825, 832, 855, 867, 867, 877, 888, 893, 908, 920, 920, 939,
947, 953, 960, 978, 997]
```

Baseado nos dados recebidos após a execução do código, extraímos estes insights:

Bubbe sort:

```
Tempo de execução: 2300 nanossegundos
Número de trocas: 13
Número de comparações: 45

E

Tempo de execução: 229600 nanossegundos
Número de trocas: 2387
Número de comparações: 4950
```

```
Insertion sort:

Tempo de execução: 1700 nanossegundos
Número de trocas: 23
Número de comparações: 28

E

Tempo de execução: 52400 nanossegundos

Número de trocas: 2813
Número de comparações: 2908
```

```
Selection sort:

Tempo de execução: 2400 nanossegundos
Número de trocas: 9
Número de comparações: 45

E

Tempo de execução: 84700 nanossegundos
Número de trocas: 108
Número de comparações: 4995
```

De suas informações, uma das mais importantes que podemos reparar, está presente no Selection sort, que foi o algoritmo que fez o menor número de trocas, tanto para o array com 10 elementos, como para o array com 100 elementos. Nos demais quesitos, obteve resultados medianos se comparado aos demais algoritmos.