CENTRO UNIVERSITÁRIO DO DISTRITO FEDERAL
Curso de Graduação em Sistemas de Informação

IVENS DE ALVARENGA NASCIMENTO

MÉTODOS DE ORDENAÇÃO

BRASÍLIA
2019

IVENS DE ALVARENGA NASCIMENTO

MÉTODOS DE ORDENAÇÃO

Métodos (ou algoritmos) de ordenação são algoritmos que organizam os elementos de uma determinada sequência em uma certa ordem.

Tendo implementado os métodos Bubble Sort, Selection Sort, Insertion Sort e Quick Sort buscamos analisar o desempenho destes para a disciplina Estrutura de Dados.

Professor: Antonio Pereira Lima

BRASÍLIA
2019

RESUMO

A análise dos métodos de ordenação em diversos cenários comparando diferentes atributos vem para mostrar em que situações os diversos algoritmos possuem desempenho elevado ou frágil, mostrando assim seus pontos fortes e fracos.

SUMÁRIO

1 INTRODUÇÃO

Na área de Tecnologia da Informação é muito comum ouvirmos que não existe melhor ou pior: depende da situação. Tendo como base este pensamento foram analisados os métodos Bubble Sort, Selection Sort, Insertion Sort e Quick Sort para determinar as especialidades e fragilidades de cada um destes algoritmos de ordenação, aplicabilidade e desempenho.

Palavras-chave: sort, métodos, algoritmos, ordenação.

2 PROCEDIMENTOS E ANÁLISE

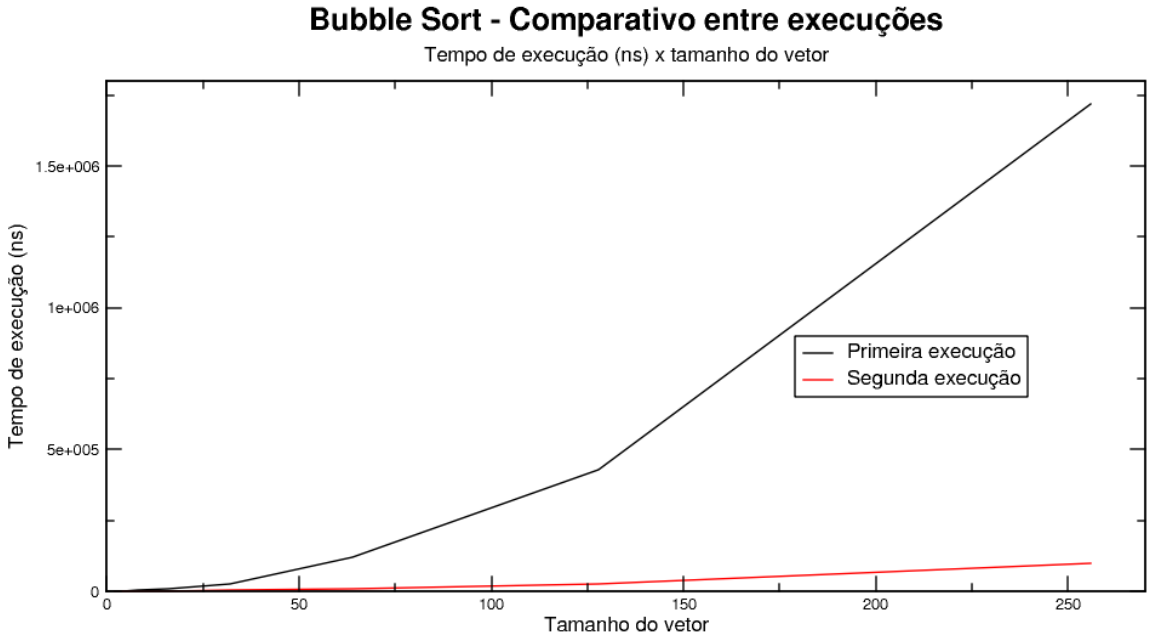
A análise dos métodos de ordenação proposta pelo professor Antonio Pereira Lima desenvolvida aqui consiste na comparação entre os valores médios de tempo de execução x tamanho do vetor, trocas x tamanho do vetor, comparações x tamanho do vetor e trocas x tempo de execução dos algoritmos de ordenação Bubble Sort, Selection Sort, Insertion Sort e Quick Sort.

Os tamanhos de vetor utilizados foram: 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192, 16384, 32768, 65536, 131072 e 262144. Foi utilizado o padrão da linguagem ("type[] vetor = new type[n]") para declaração dos vetores.

Os algoritmos ordenam os números de forma crescente.

Para obter os valores médios o algoritmo foi rodado 11 vezes, e o tempo de execução dos algoritmos foram obtidos em nanosegundos (ns) para maior precisão.

A primeira execução dos algoritmos consumiu um tempo consideravelmente maior, e no caso vetores maiores existiram discrepâncias sutis no tempo de execução de cada um. Por conta disso os dados de tempo de execução da primeira execução (ou execução 0, no arquivo .txt) foram desconsiderados. Os fenômenos podem ser conferidos abaixo, utilizando como exemplo o tempo de execução do algoritmo Bubble Sort até o vetor de tamanho 256.



A cada execução é gerado um vetor multidimensional (matriz) de 4 linhas (1 pra cada algoritmo de ordenação) e número de colunas ditado pelo tamanho do vetor. Cada algoritmo de ordenação ordena uma dessas linhas, sendo que cada uma destas possuem os mesmos dados nas mesmas posições.

Todo o código utilizado e a evolução deste entre as etapas propostas no roteiro, o arquivo gerado pelo algoritmo contendo os dados das 10 execuções e os dados utilizados nos gráficos podem ser encontrados no GitHub (https://github.com/ialvarenga/udf-22019-ed).

Todos os números que necessitassem de arredondamento foram arredondados para cima, sem casas decimais.

Durante a execução do algoritmo não foi alterada a prioridade do processo do mesmo.

Caso tenha detectado algum problema, inconsistência ou bug no algoritmo, favor abrir um *issue* no mesmo GitHub.

O algoritmo foi rodado num computador com as seguintes especificações:

Intel(R) Pentium(R) CPU G2030 @ 3.00 GHz

9 GB RAM (2GB DDR3-1333 + 4GB DDR3-1333 + 1GB DDR3-1333 + 2GB DDR3-1333)

Seagate Desktop HDD ST2000DM001

Windows Server 2019 Standard 1809 (compilação 17763.107)

A análise está dividida em 4 partes:

- Tamanho do vetor vs tempo de execução
- Tamanho do vetor vs trocas
- Tamanho do vetor vs comparações
- Tempo de execução vs trocas

Cada análise envolvendo tamanho de vetor está dividida da seguinte forma:

- 2^1-64
- 128^2-512
- 1024-4096
- 2^1-262144^2

Os períodos foram divididos desta forma por apresentam grande diferença entre seus resultados, logo desafiando o objetivo da análise destes em um gráfico de linhas em alguns casos.

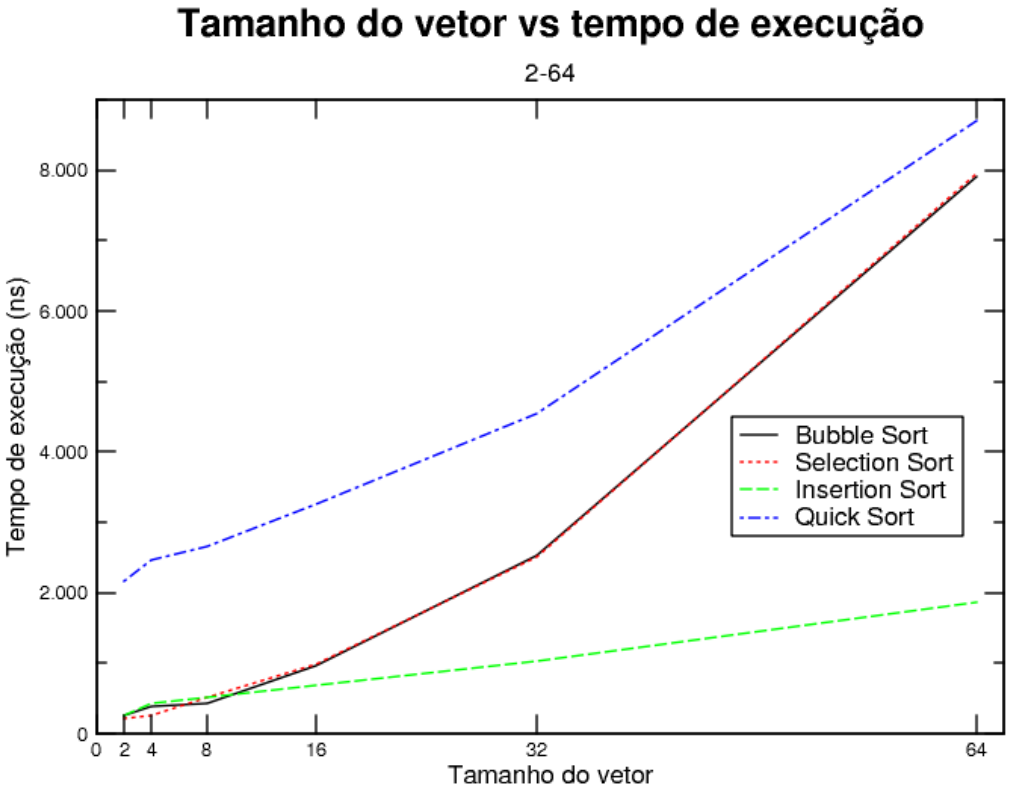
[1] A 8ª execução do algoritmo Quick Sort no vetor de tamanho 2 levou 102500ns. Considerando que a média aritmética arredondada pra cima sem este foi de 2167 a diferença é de 100333ns. O valor foi desconsiderado.

[2] A 9ª execução do algoritmo Selection Sort no vetor de tamanho 128 levou 256300ns. Considerando que a média aritmética arredondada pra cima sem este foi de 4112 a diferença é de 252188ns. O valor foi desconsiderado.

3 TAMANHO DO VETOR vs TEMPO DE EXECUÇÃO

A análise tamanho do vetor vs tempo de execução demonstra quanto o tamanho do vetor pode influenciar na operação (e, consequentemente, a complexidade) e tempo necessário para ordenação utilizando cada um dos algoritmos, mostrando os pontos fortes e fracos de cada um destes.

3.1 VETORES DE TAMANHO 2-64

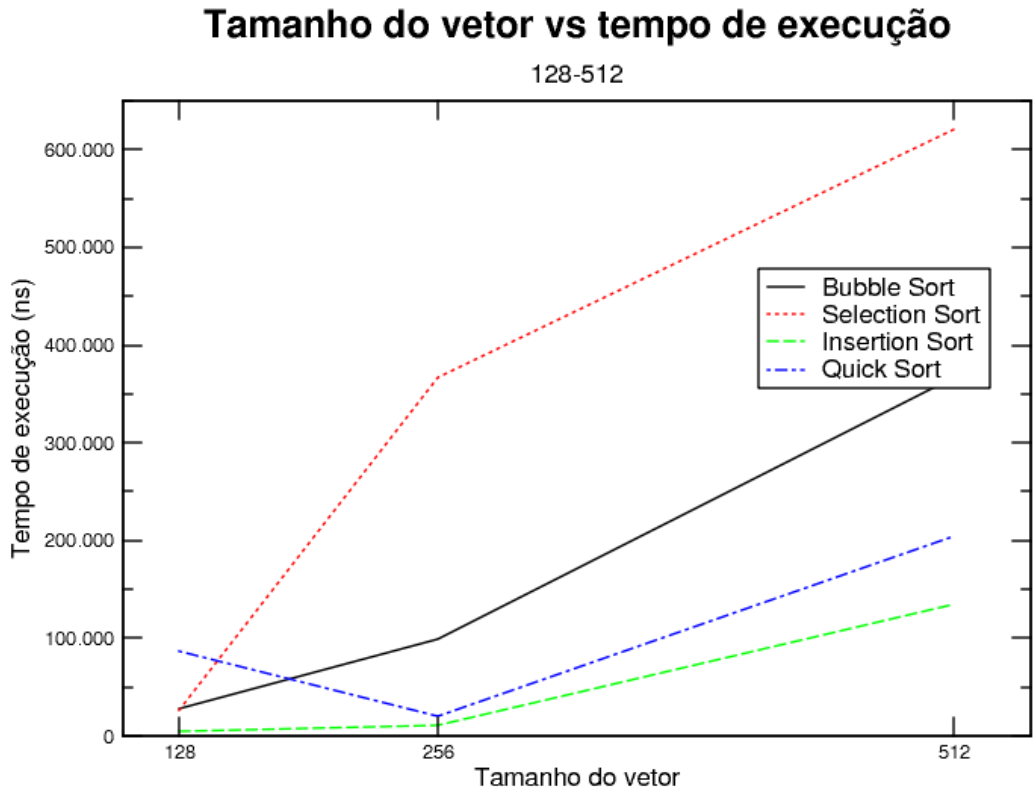


Através dos gráficos é possível observar que os algoritmos Bubble Sort, Selection Sort e Insertion Sort possuem tempos de execução próximos nos vetores de menor tamanho (2, 4 e 8). Selection Sort leva vantagem nos vetores de tamanho 2 e 4, e Bubble Sort parece bem aplicado no vetor de 8 posições.

Com vetores maiores que 8 posições até 64 o método Insertion Sort possui um desempenho superior aos outros métodos de ordenação. Já Bubble Sort e Selection Sort possuem desempenho p em comparação ao Insertion Sort.

Nos vetores de 2 posições até 64 posições Quick Sort possui um desempenho ruim, com um tempo de execução muito maior que os outros métodos testados.

3.2 VETORES DE TAMANHO 128-512

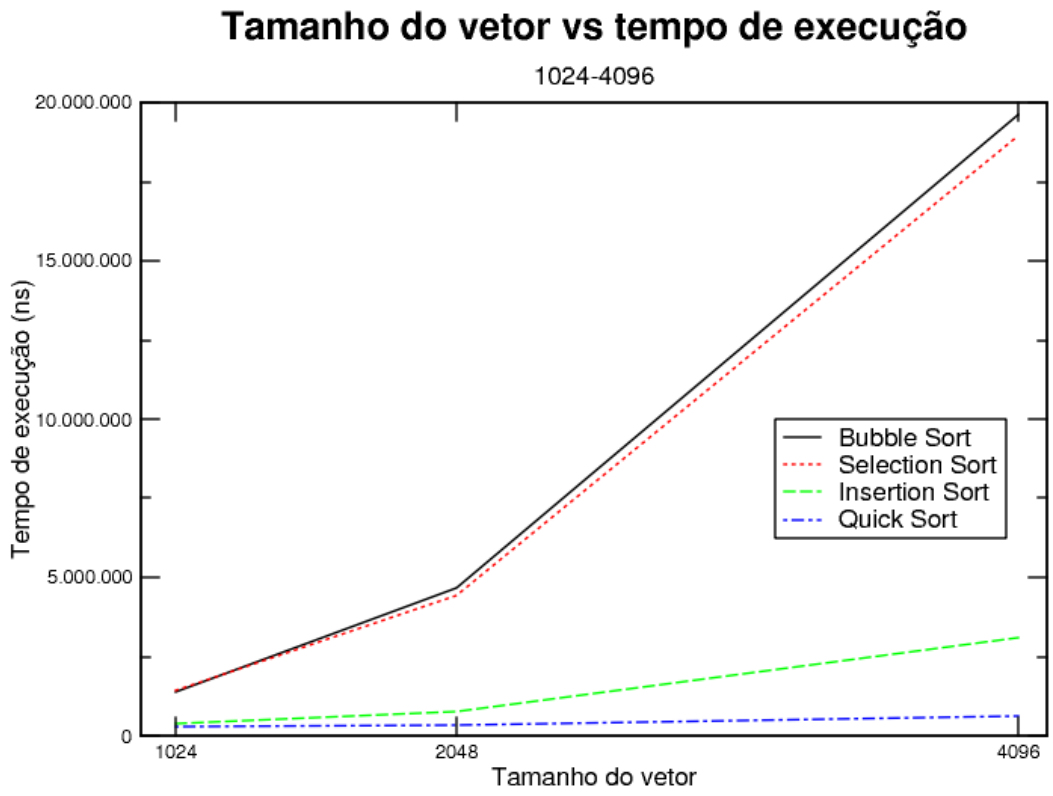


Os gráficos mostram que o Insertion Sort tem o melhor desempenho entre os 4 métodos testados nos vetores de 128 a 512 posições.

Por se tratarem de algoritmos mais simples, Selection Sort e Bubble Sort possuem desempenhos parecidos até vetores de 128 posições. A partir daí, o Selection Sort começa a mostrar suas deficiências, enquanto que o Bubble Sort também não se mostra uma boa alternativa em relação aos outros métodos de ordenação.

Com seu algoritmo de recursão Quick Sort começa a mostrar bom desempenho à medida que o vetor vai crescendo, tendo desempenho próximo – porém não melhor – do que Insertion Sort.

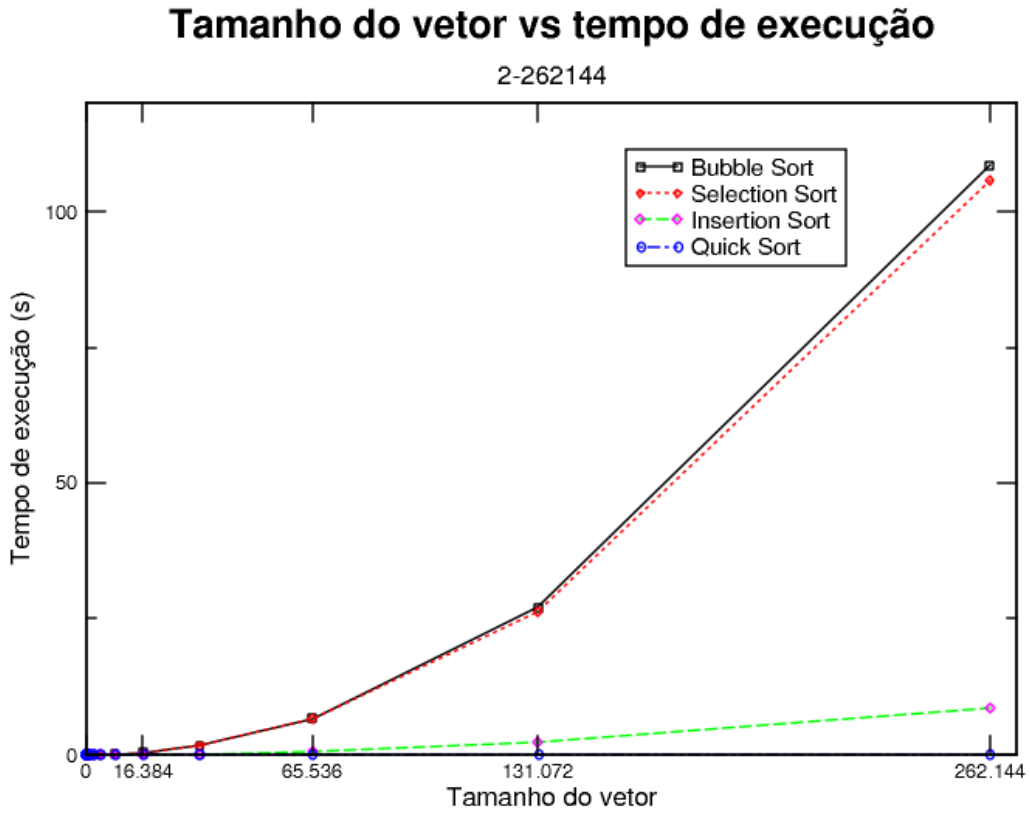
3.3 VETORES DE TAMANHO 1024-4096



É possível verificar através dos gráficos que Insertion Sort tem o melhor desempenho entre os 4 métodos testados nos vetores de 128 a 512 posições. Selection Sort e Bubble Sort possuem desempenhos parecidos até vetores de 128 posições. A partir daí, o Selection Sort começa a mostrar suas deficiências, enquanto que Bubble Sort também não se mostra uma boa alternativa em relação aos outros métodos de ordenação.

Quick Sort começa a mostrar bom desempenho à medida que o vetor vai crescendo, tendo desempenho próximo ao Insertion Sort até o vetor de 2.048 posições.

3.4 VETORES DE TAMANHO 2-262144



A vantagem do Quick Sort em grandes vetores se torna ainda mais expressiva ao se comparar todos os tamanhos de vetor testados a ponto da linha representando o tempo de execução deste não estar visível no gráfico acima.

Os algoritmos de ordenação Bubble Sort e Selection Sort voltam a mostrar suas dificuldades com vetores maiores que 16 posições, ultrapassando a marca de 100s de tempo de execução durante o teste no vetor de 262 mil posições.

Entre os 4 métodos testados o Insertion Sort é o que fica mais próximo do Quick Sort, porém já começa a mostrar complicações ao lidar com vetores a partir de 131.072 posições, em que seu tempo de execução cresce significativamente.

Note-se que o gráfico possui tempo de execução expressado em segundos, ao contrário dos outros gráficos mostrados neste capítulo, com tempo de execução expressado em nano segundos.

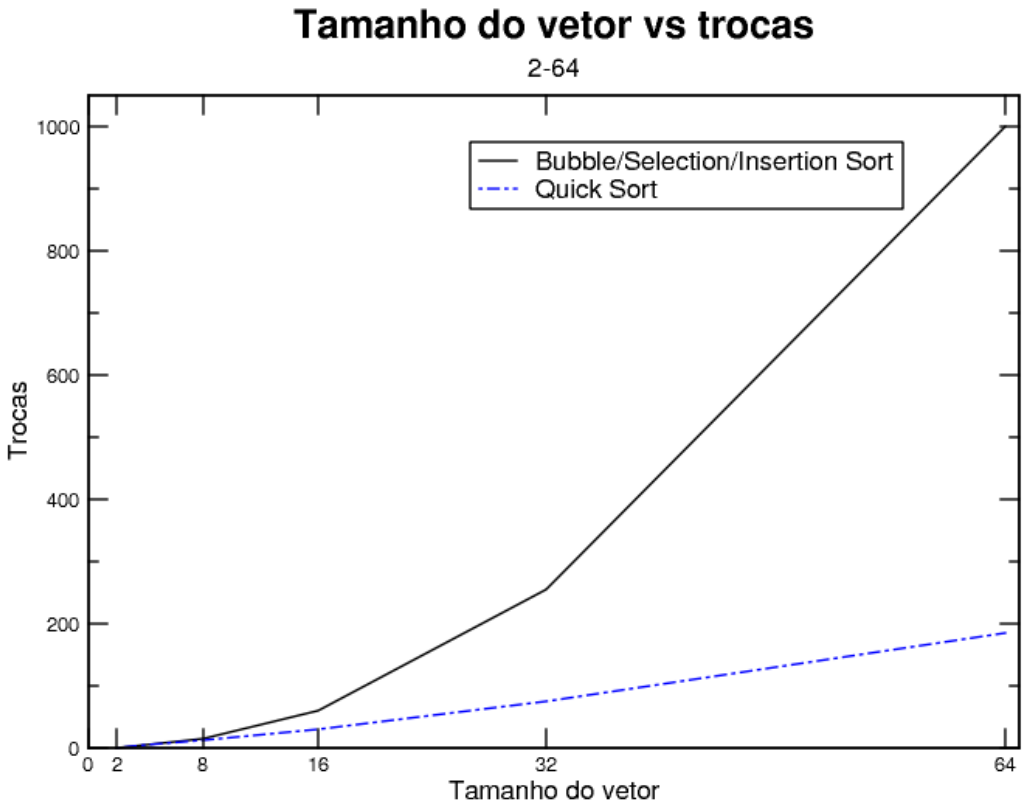
4 TAMANHO DO VETOR vs TROCAS

A análise tamanho do vetor vs trocas mostra quais algoritmos utilizam menos recursos para realizar a operação de alteração da posição de um número dentro de um *array* quando necessário por se tratar de um número fora da ordem, logo alcançando o objetivo do algoritmo.

Por se tratar de uma grandeza que sofre alterações de acordo com quão desorganizado o vetor está foi considerada a média destas entre as 11 execuções citadas na introdução.

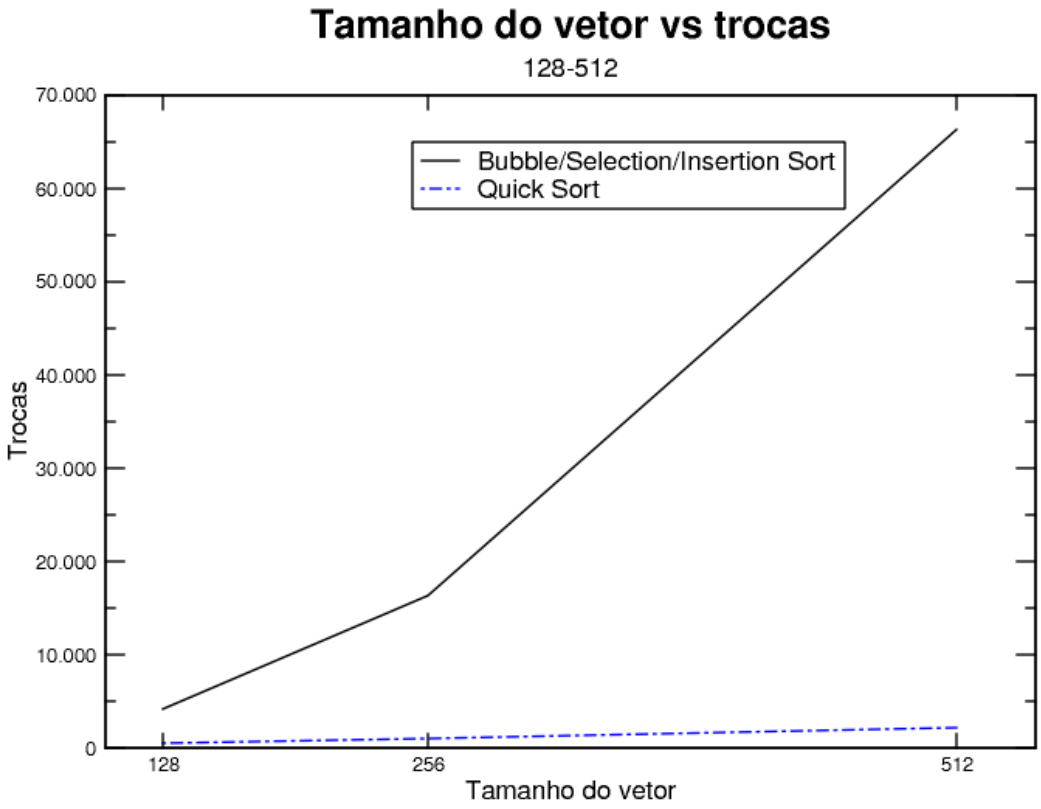
Os métodos Bubble Sort, Selection Sort e Insertion Sort possuem os mesmos valores de trocas por se tratarem de algoritmos com implementação parecida.

4.1 VETORES DE TAMANHO 2-64



Por mais que o Quick Sort tenha um tempo de execução maior quando comparado à Bubble Sort, Selection Sort ou Insertion Sort nos vetores de menor tamanho este realiza uma quantidade notadamente menor de trocas à medida que a capacidade do vetor cresce.

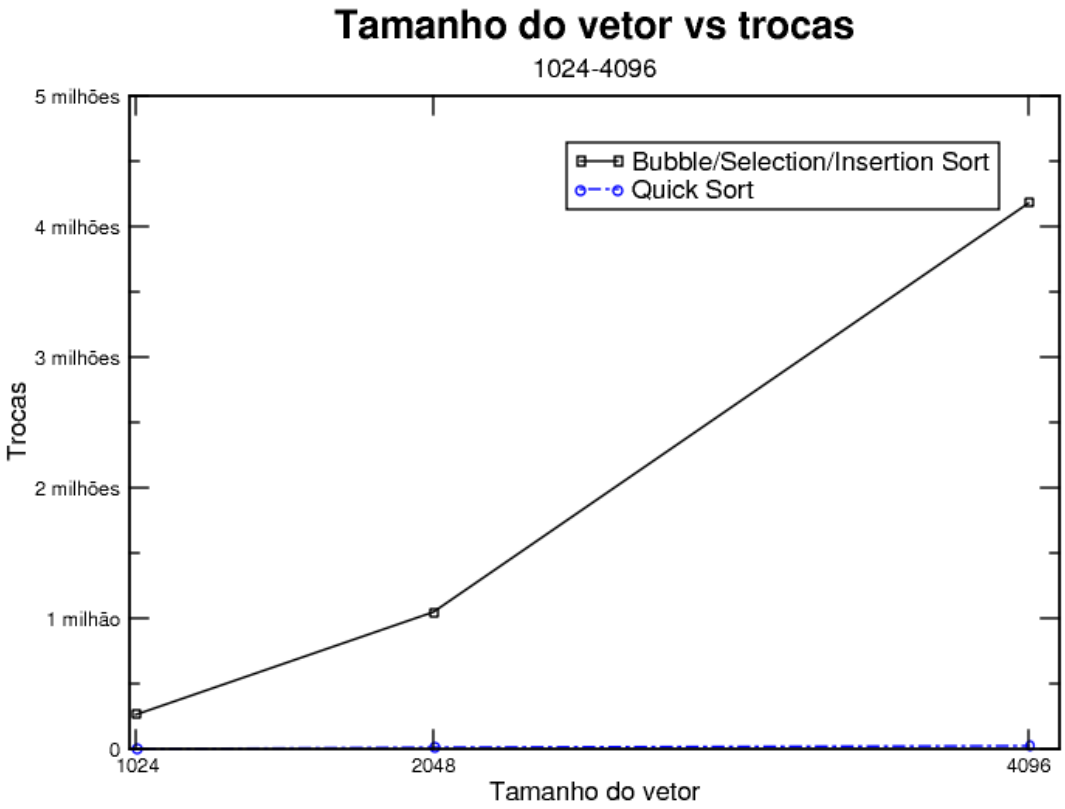
4.2 VETORES DE TAMANHO 128-512



Ao comparar a média de trocas entre os vetores de 128 a 512 posições Quick Sort mostra sua gritante superioridade aos demais métodos de ordenação, realizando um número muito menor de trocas.

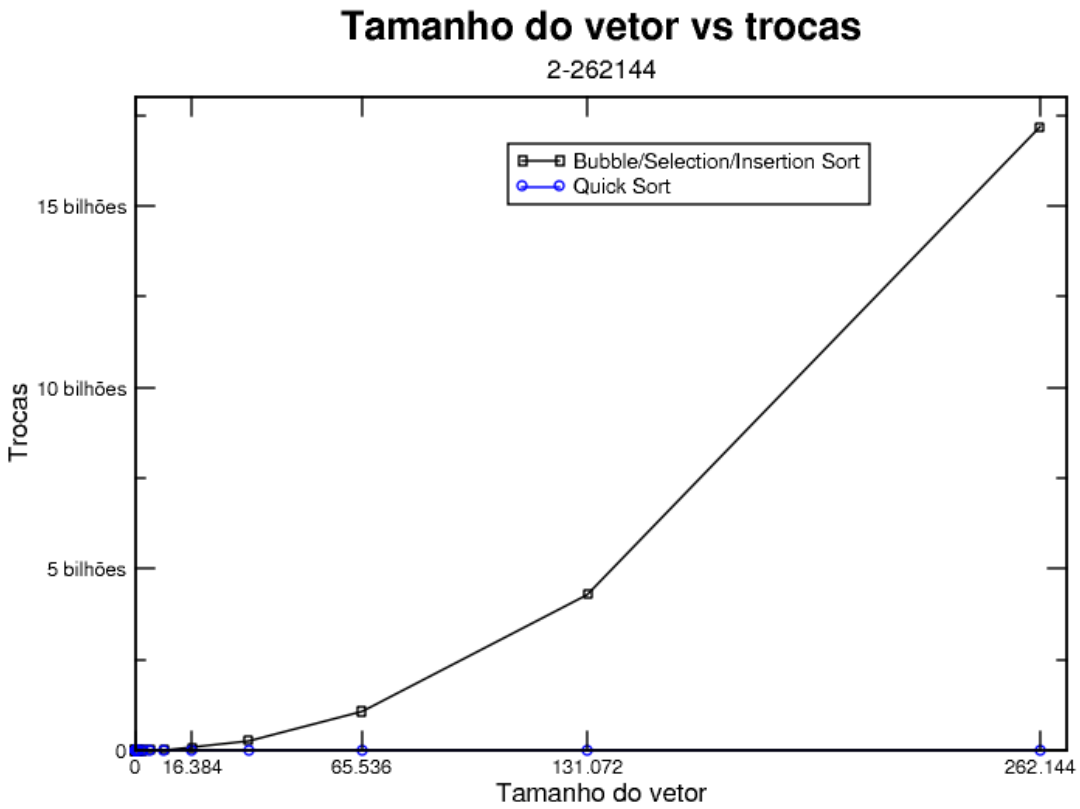
É possível verificar que mesmo em vetores até 128 posições Bubble Sort, Selection Sort e Insertion Sort não possuem uma quantidade tão alta de trocas em comparação ao Quick Sort, ainda que seja uma diferença considerável. Já a partir disto estes algoritmos começam a mostrar seus pontos fracos. A tendência é que o Quick Sort aumente ainda mais a distância, sendo um algoritmo melhor ao lidar com vetores grandes, como visto ao comparar o tamanho do vetor ao tempo de execução.

4.3 VETORES DE TAMANHO 1024-4096



Assim como nas outras comparações entre tamanho do vetor e trocas em vetores de outros tamanhos, o gráfico é parecido, e reforça a disparidade do algoritmo Quick Sort: ao dobrar o tamanho do vetor o número de trocas quase que quadruplica ao utilizar os algoritmos Bubble Sort, Selection Sort ou Insertion Sort.

4.4 VETORES DE TAMANHO 2-262144



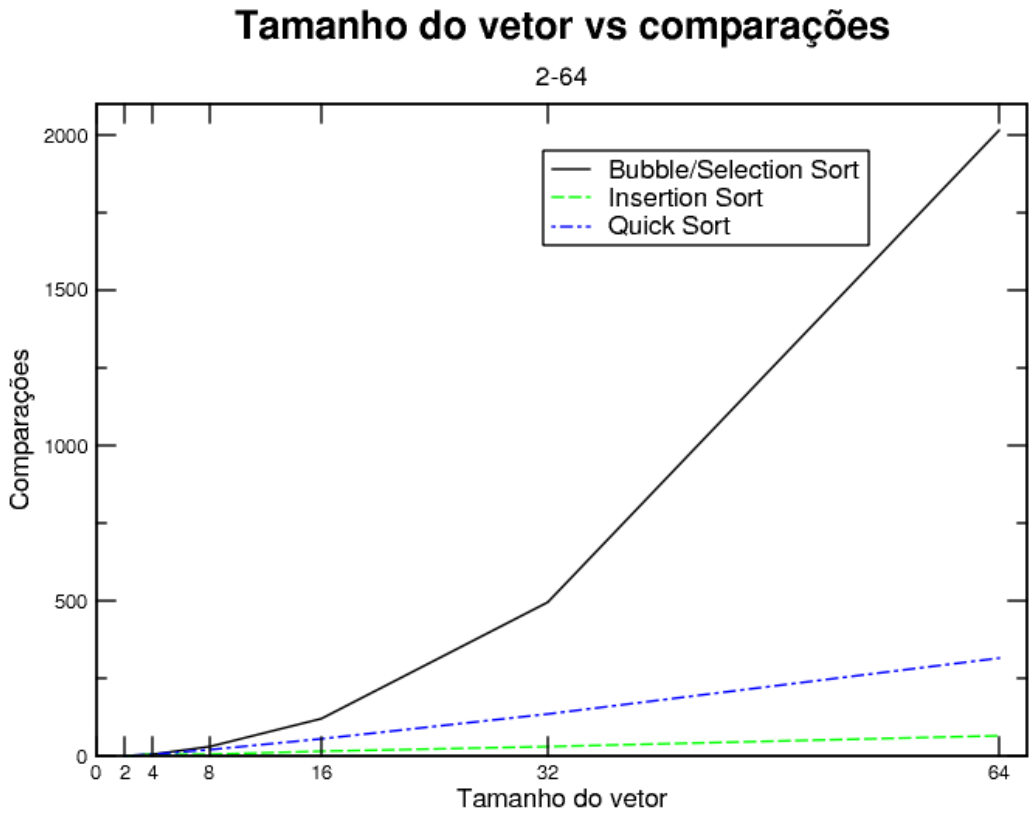
Neste gráfico é possível verificar que não importa o tamanho do vetor: ao dobrar o tamanho do vetor a quantidade de trocas fica próximo do quádruplo quando comparado ao vetor anterior nos algoritmos Bubble Sort, Selection Sort e Insertion Sort. Mais uma vez o contraste entre estes algoritmos e Quick Sort é evidenciado quando o algoritmo Quick Sort fica quase invisível ao considerar todos os tamanhos de vetor analisados aqui.

5 TAMANHO DO VETOR vs COMPARAÇÕES

A análise tamanho do vetor vs comparações exprime quantas comparações foram necessárias para organizar todo o vetor.

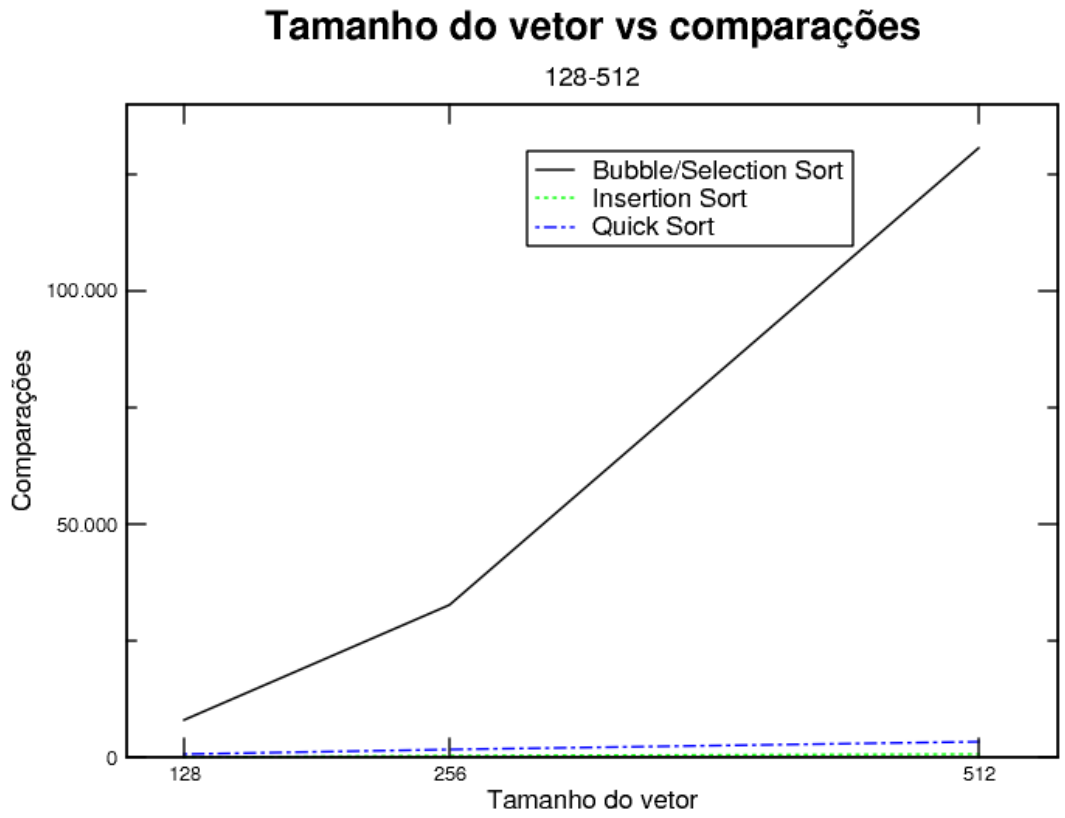
No caso dos algoritmos Bubble Sort, Selection Sort e Insertion Sort trata-se de uma grandeza só se altera caso o tamanho do vetor for modificado. Por exemplo: utilizando o método Bubble Sort ou Selection Sort um vetor de 256 posições vai sempre passar por 130.816 comparações, salvo troca do local onde o incrementador do contador se encontra. Isso não se aplica em Quick Sort, onde foi considerado a média das execuções, já que este valor não é fixo como nos algoritmos anteriormente citados.

5.1 VETORES DE TAMANHO 2-64



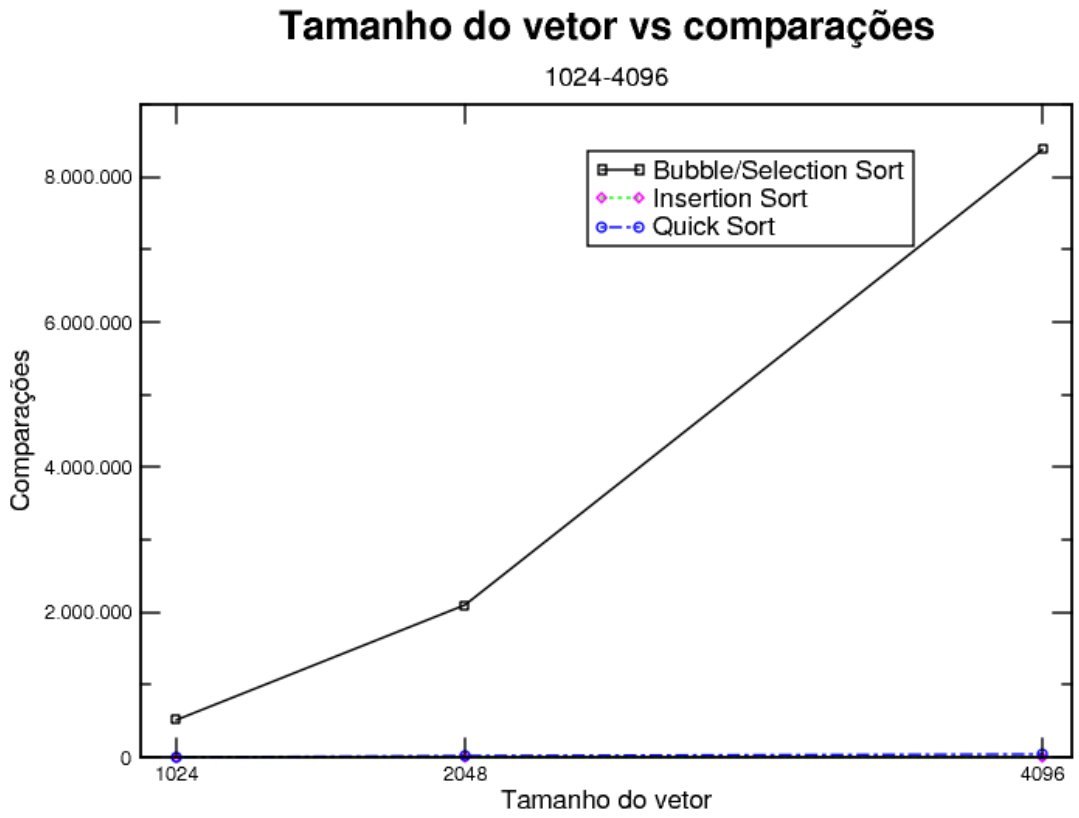
A vantagem do Insertion Sort sobre o Quick Sort é visível neste gráfico. Já no caso do Bubble Sort e Selection Sort a propensão dos valores quadruplicarem ao dobrar o tamanho do vetor continua, realçando assim o desempenho superior dos algoritmos Insertion Sort e Quick Sort.

5.2 VETORES DE TAMANHO 128-512



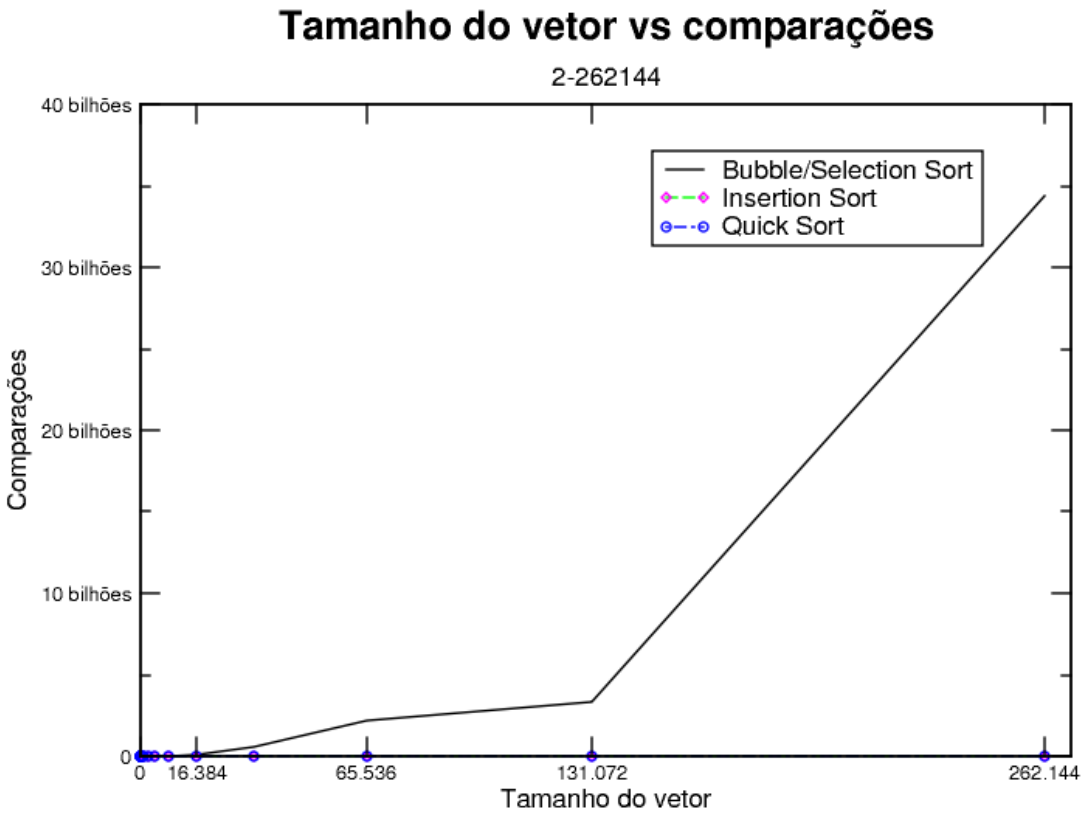
Os métodos Bubble Sort e Selection Sort revelam seu alto números de comparações quando colocado lado a lado com os métodos Insertion Sort e Quick Sort. O método Insertion Sort mostra vantagem sobre o método Quick Sort com um número de comparações formidavelmente menor.

5.3 VETORES DE TAMANHO 1024-4096



A disparidade dos métodos Bubble Sort e Selection Sort no que tange a quantidade de comparações continua a crescer, chegando ao ponto das linhas representando Insertion Sort e Quick Sort respectivamente estarem próximos do eixo X, quase invisíveis.

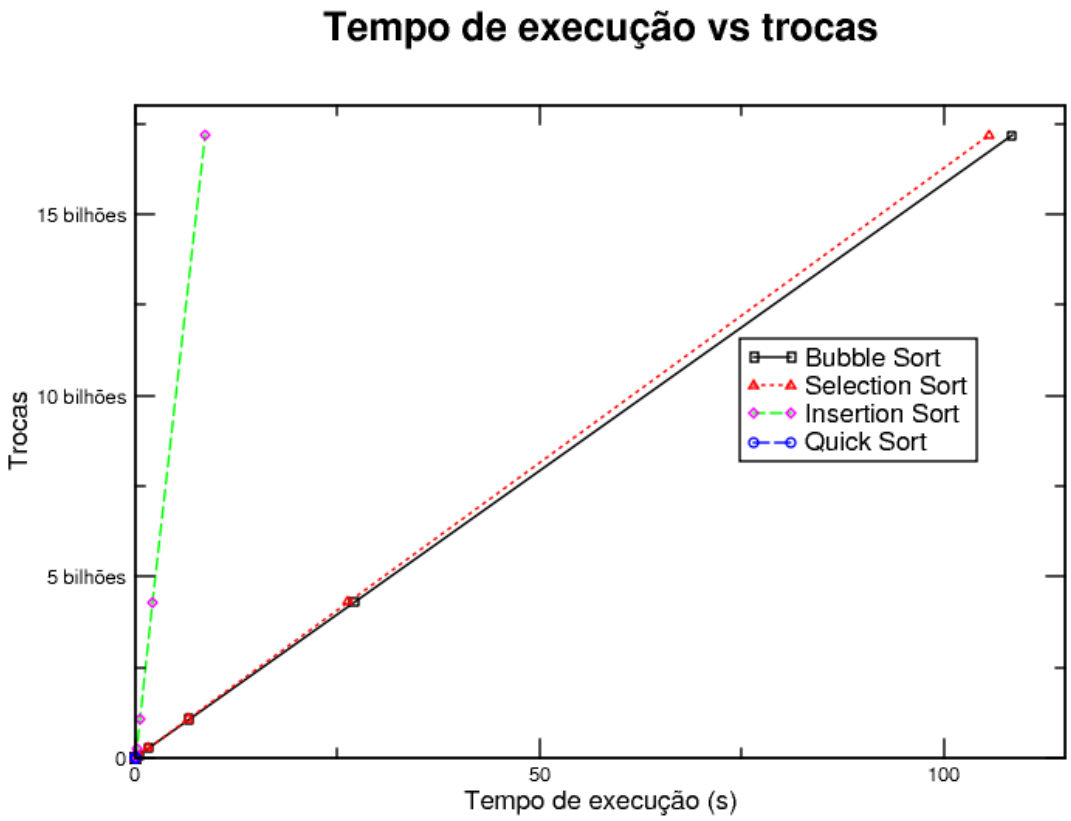
5.4 VETORES DE TAMANHO 2-262144



Os métodos Bubble Sort e Selection Sort não se mostram boas opções em vetores de muitas posições (>128 posições) no que diz respeito a quantidade de comparações. Os métodos Insertion Sort e Quick Sort mostram ser opções de melhor desempenho nestes casos, mantendo um valor extraordinariamente menor que os outros métodos.

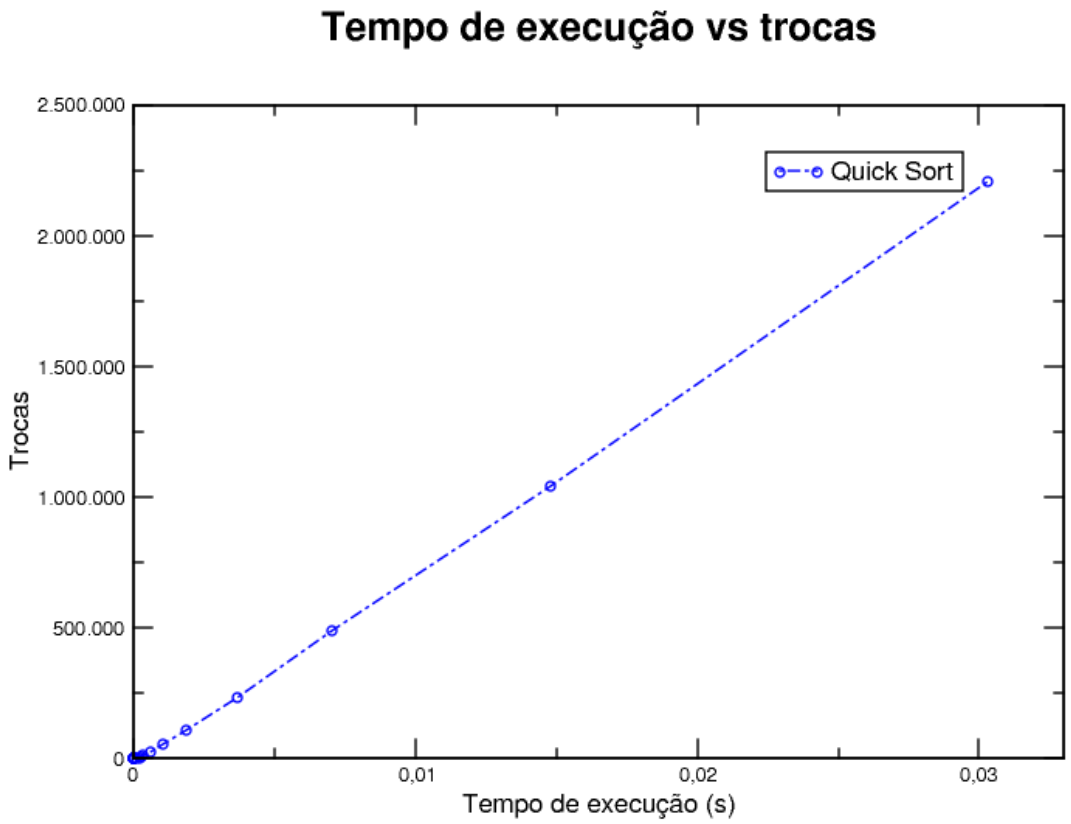
6 TEMPO DE EXECUÇÃO VS TROCAS

A análise de tempo de execução vs trocas permite avaliar quanto as trocas impactam no tempo de execução dos algoritmos, dando um parecer sobre sua performance de modo geral.



Bubble Sort e Selection Sort possuem desempenho parecido: ambos realizaram mais de 15 bilhões de trocas, enquanto Insertion Sort realizou o mesmo número de trocas em um tempo muito menor.

Quick Sort leva uma grande vantagem sobre os algoritmos aqui comparados, ao ponto de que nem mesmo aparece neste gráfico.



Ainda que o Insertion Sort possua um desempenho superior ao Bubble Sort e Selection Sort, Quick Sort ainda possui um tempo de execução muito menor que o Insertion Sort, realizando um número amplamente menor de trocas.

Portanto, o número de trocas pode influenciar no tempo de execução do algoritmo, mas como mostrado pelo Insertion Sort em comparação ao Bubble Sort e Selection Sort, não é a única razão pela qual esta grandeza cresce.

7 CONCLUSÃO

Os métodos de ordenação e sua análise compõem uma parte importante do aprendizado ao considerar as grandes diferenças que Bubble Sort, Selection Sort e Insertion Sort apresentam, esclarecendo assim os pontos fortes e fracos de cada um e seus melhores usos.

8 BIBLIOGRAFIA

DUDE, That JS. **"JS: Sorting Algorithm"**. Disponível em: https://khan4019.github.io/front-end-Interview-Questions/sort.html. Acesso em: 29/8/2019.