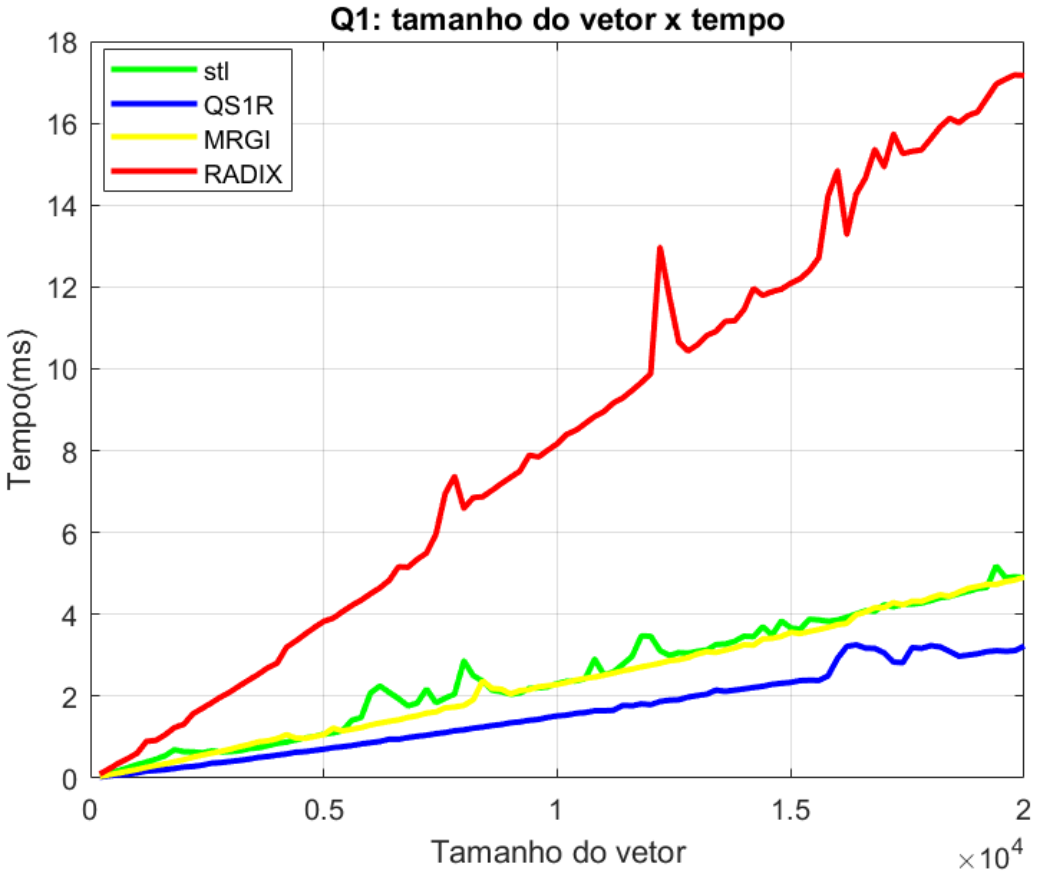
Aluno: Caue Marçal Guimarães - Lab 3 CTC12

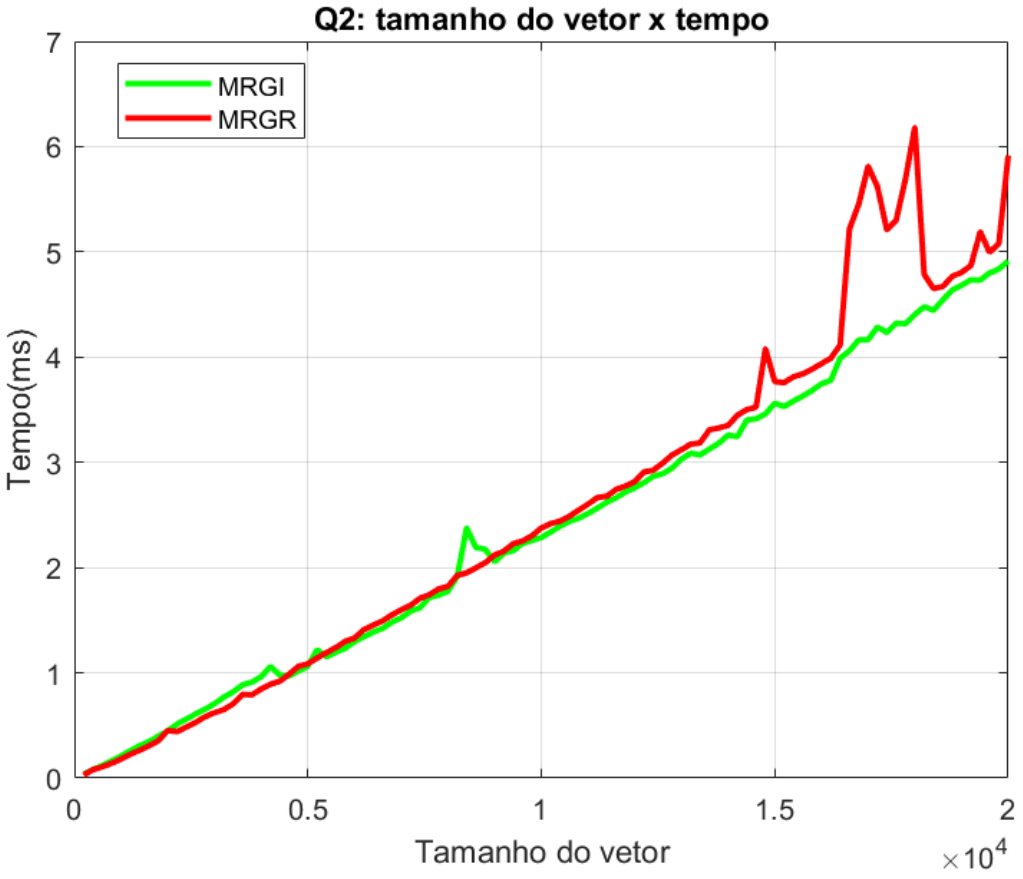
Q1 QuickSort X MergeSort X RadixSort x std::sort

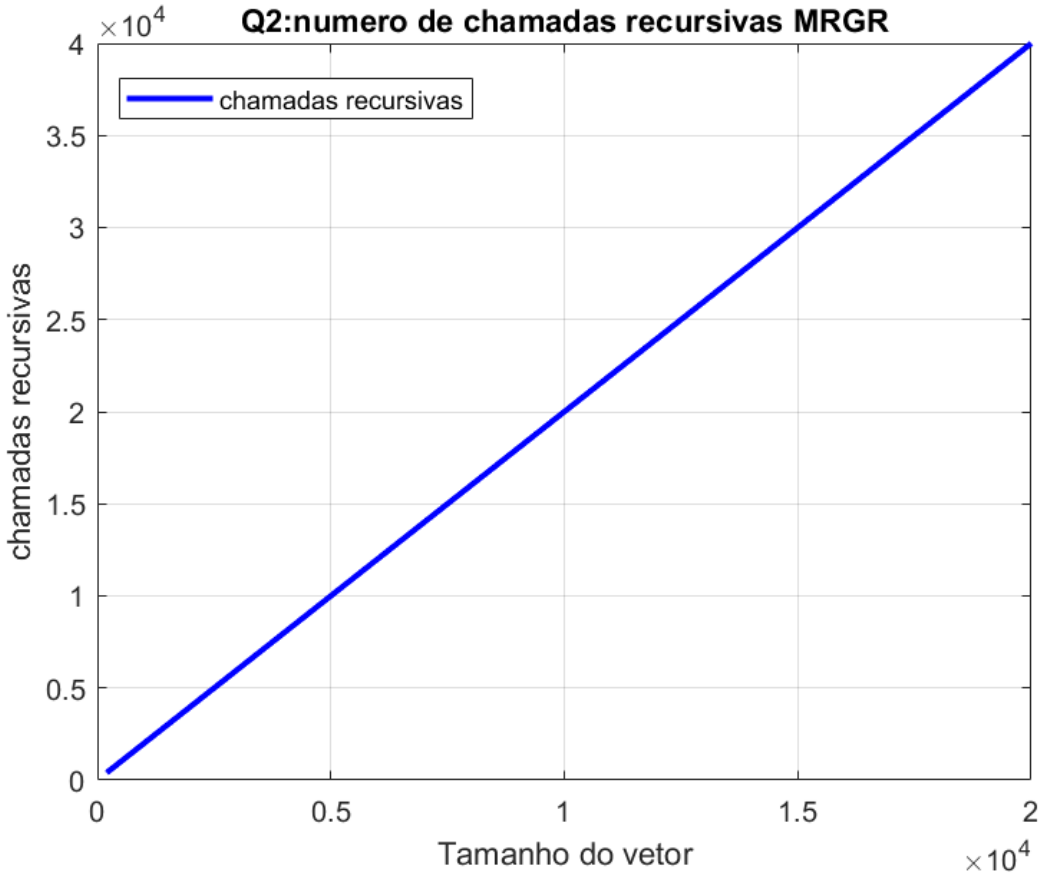
Aqui comparamos os tempos para vetores random usando stl, QuickSort com uma recursão e mediana de 3 (QS1R), MergeSort Iterativo (MRGI) e o RadixSort (RADIX). Como previsto em teoria, vemos um melhor desempenho do QuickSort em relação aos demais. Este, apesar de, como o MergeSort, no caso médio, ser O(n.logn), por utilizar apenas operações rápidas apresenta um melhor resultado que o MergeSort, que já tem operações que exigem maior tempo de processamento (manipulação de subvetores). Já comparando com o Radix, apesar dele ser O(n), suas altas constantes fazem com que ele saia perdendo. Existe apenas o adendo acerca da pilha de recursão gerada pelo QuickSort que pode vir a ser um problema para a memória dependendo do caso, apesar de ele já ser um Quick melhorado com apenas uma chamada recursiva essa desvantagem precisa ser levada em conta.

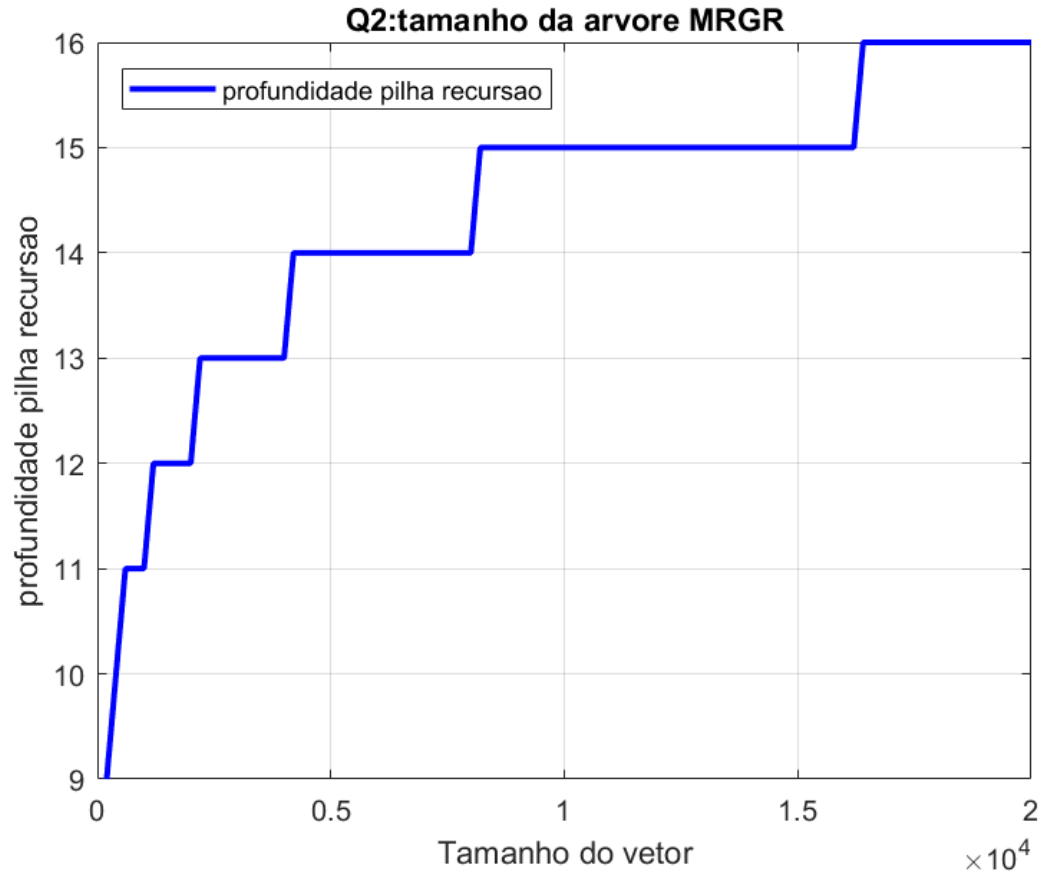


Q2 MergeSort: Recursivo x Iterativo

Aqui foi feita a análise acerca do tempo de ordenação do MergeSort recursivo (MRGR) e do MergeSort iterativo (MRGI), comparados para vetores aleatórios**.** Aqui vemos que o MRGI é levemente superior em relação ao tempo, e superior, no geral, por não possuir as chamadas recursivas do MRGR que ocupam espaço na árvore de recursão.

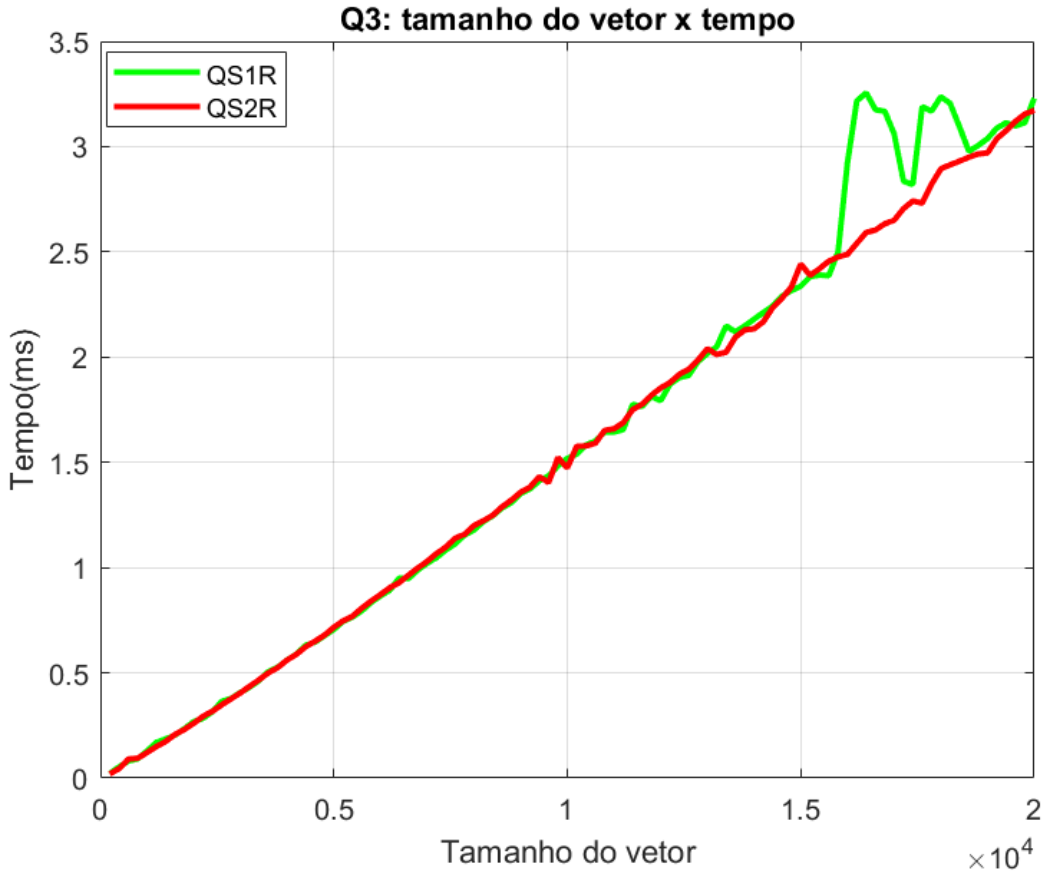


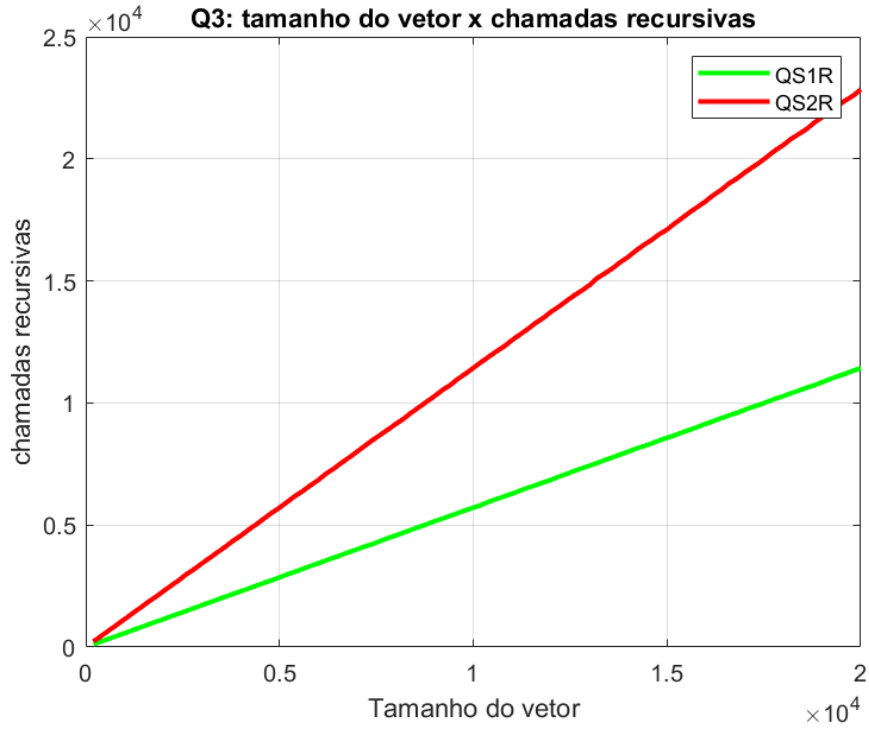

Q2:numero de chamadas recursivas MRGR
×10^4
chamadas recursivas
chamadas recursivas
Tamanho do vetor
×10^4
0
0.5
1
1.5
2
2.5
3
3.5
4

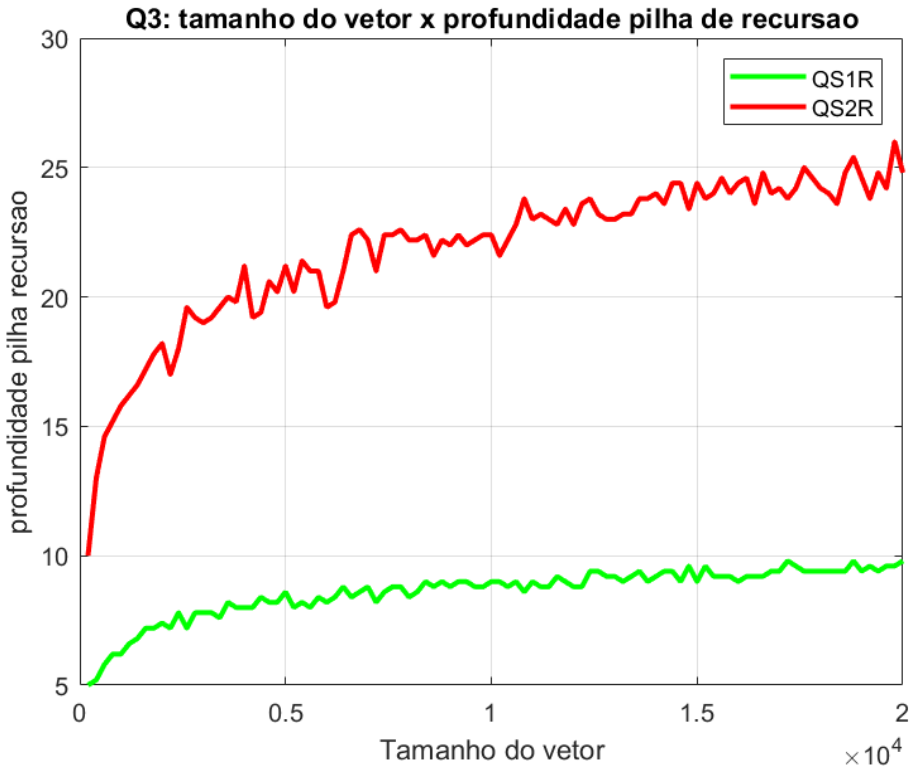

Q2:tamanho da arvore MRGR
profundidade pilha recursao
profundidade pilha recursao
Tamanho do vetor
×10^4
9
10
11
12
13
14
15
16
0
0.5
1
1.5
2

Q3 QuickSort com 1 recursão x QuickSort com 2 recursões

Aqui foi feita a comparação entre o QuickSort com 1 recursão (QS1R) e o QuickSort com 2 recursões (QS2R), ambos usando mediana de 3, e com as comparações sendo feitas para vetores randômicos. Em relação ao tempo de execução vemos que ambos são muito similares, porém, nota-se uma melhor alocação de memória da pilha de execução do QS1R quando comparamos tanto o número de chamadas recursivas quanto o tamanho da árvore de ambos, a árvore do QS1R chega a ser quase três vezes menor que a do QS2R

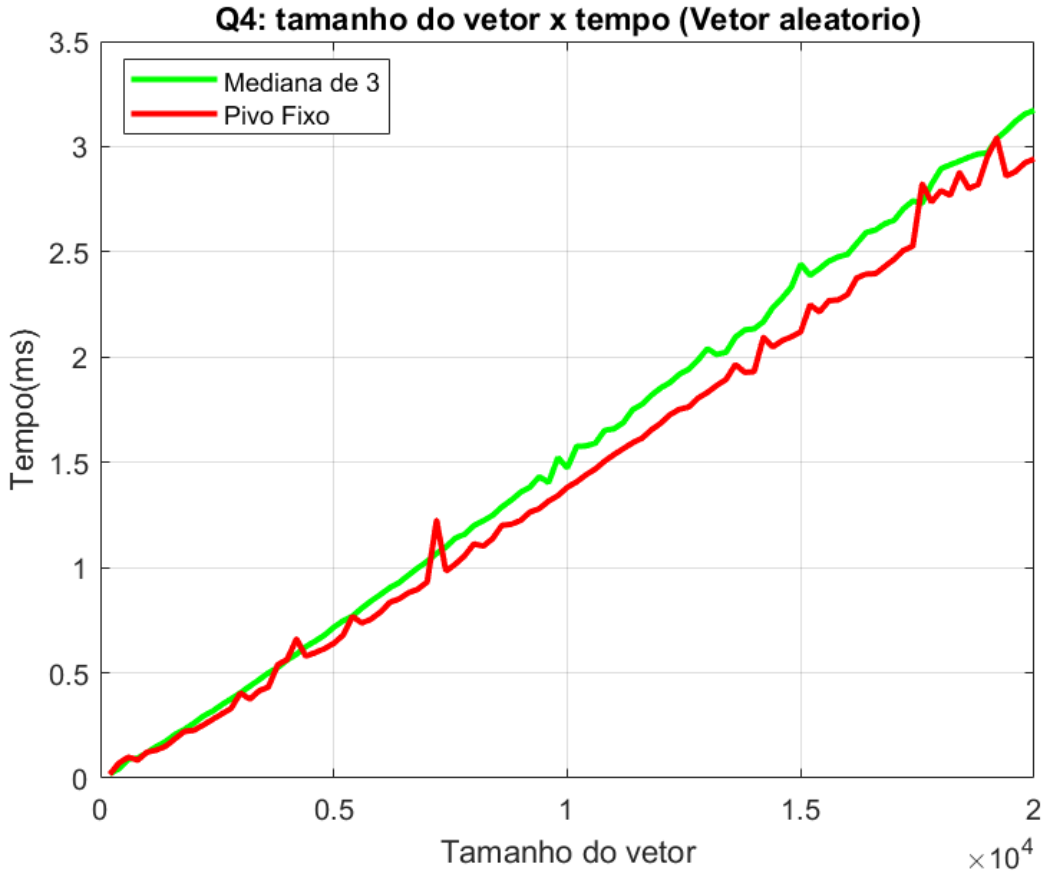


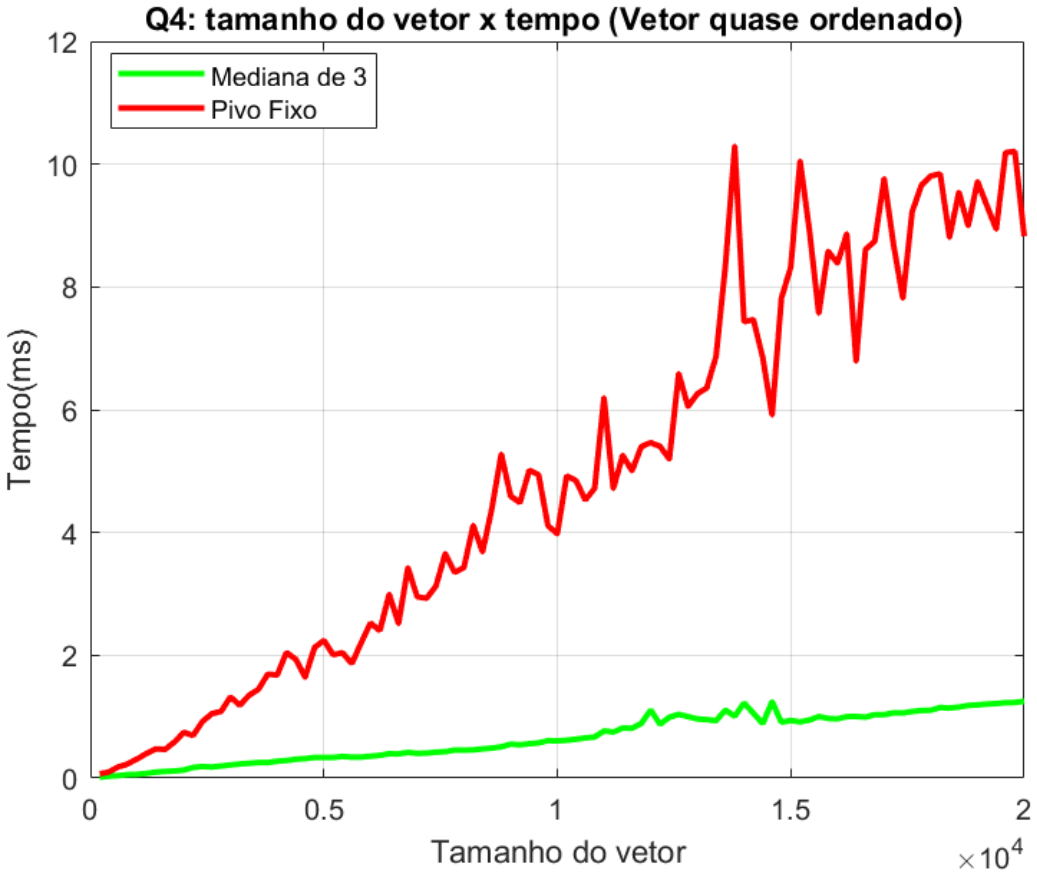

Q3: tamanho do vetor x chamadas recursivas
×10^4
2.5
2
1.5
1
0.5
0
chamadas recursivas
QS1R
QS2R
0
0.5
1
1.5
2
Tamanho do vetor
×10^4


Q3: tamanho do vetor x profundidade pilha de recursao
30
25
20
15
10
5
profundidade pilha recursao
QS1R
QS2R
0
0.5
1
1.5
2
Tamanho do vetor
×10^4

Q4 QuickSort com mediana de 3 x Quicksort com pivô fixo para vetores

Aqui comparamos o QuickSort com mediana de 3 com o QuickSort de pivo fixo, ambos com duas chamadas recursivas. Primeiro foi feita a análise para vetores aleatórios onde já se nota um leve melhor desempenho da mediana de 3 em relação ao pivô fixo, o que é esperado uma vez que na mediana de 3 são raros os casos em que o QuickSort gasta tempo quadrático. Agora a análise fica ainda mais nítida ao se comparar ambos para vetores quase ordenados, que é justamente o pior caso para o pivô fixo (quadrático), que acaba realizando muitas trocas desnecessárias, sendo extremamente mais lento que a mediana de 3 que faz um bom trabalho evitando tais trocas.




Q4: tamanho do vetor x tempo (Vetor quase ordenado)
Mediana de 3
Pivo Fixo
Tempo(ms)
Tamanho do vetor
×10^4
0
2
4
6
8
10
12
0
0.5
1
1.5
2

Relatório: Q5 Questão sobre caso real:

Resposta: Não faz sentido o QuickSort estar implementado como pivô fixo, dado que a implementação por mediana de 3 é uma correção extremamente simples e já torna raríssimo os casos de ordem quadrática. No entanto, se já se trata de uma implementação eficiente como a mediana de 3, e são realmente casos muito específicos que acabam atingindo a ordem quadrática, não vejo problema na implementação, pois, nos casos reais os vetores quase ordenados são extremamente raros, e, se por algum motivo, eles se tornam parte do projeto, cabe a equipe do projeto solucionar tal problema específico. É melhor solucionar um problema específico de forma específica do que trocar uma solução muito boa para o caso geral e com ótimos custo-benefício por uma extremamente complicada, com péssimo custo-benefício, que atenda a exceção.