Home

You can type something here...

Algoritmos de Intercalação e Troca

Bhruno R. Leifheit; Rafael C. Ribeiro; Gustavo D. Silva; Thiago C. M. Araujo; Rafael Torres


Twitter


Facebook


Instagram

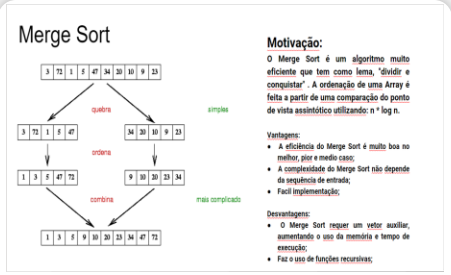

SlidesMania

2021

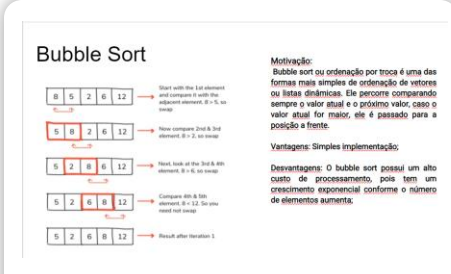
Home | Recent Tabs

You can type something here...

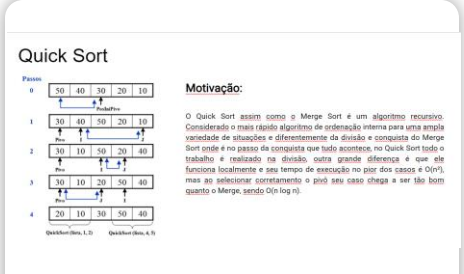
Busca...



Merge Sort

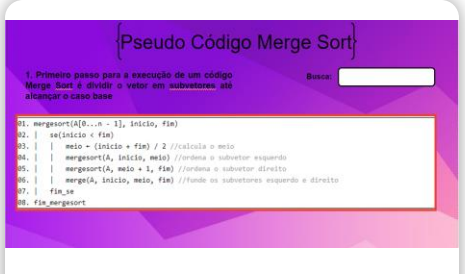


Bubble Sort

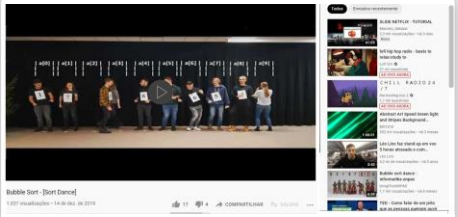


Quick Sort

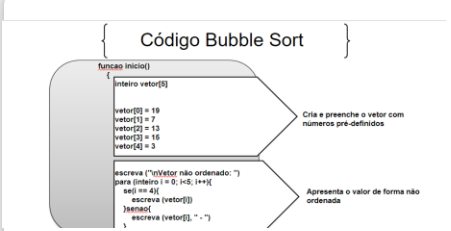
Funcionamento Merge Sort



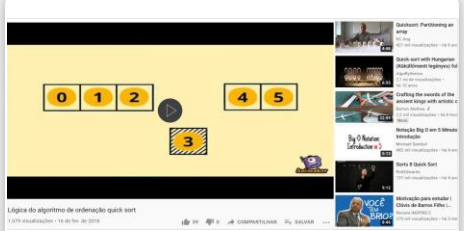
Pseudo Código Merge Sort



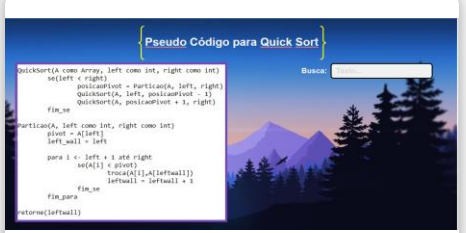
Funcionamento Bubble Sort



Pseudo Código Bubble Sort



Funcionamento Quick Sort

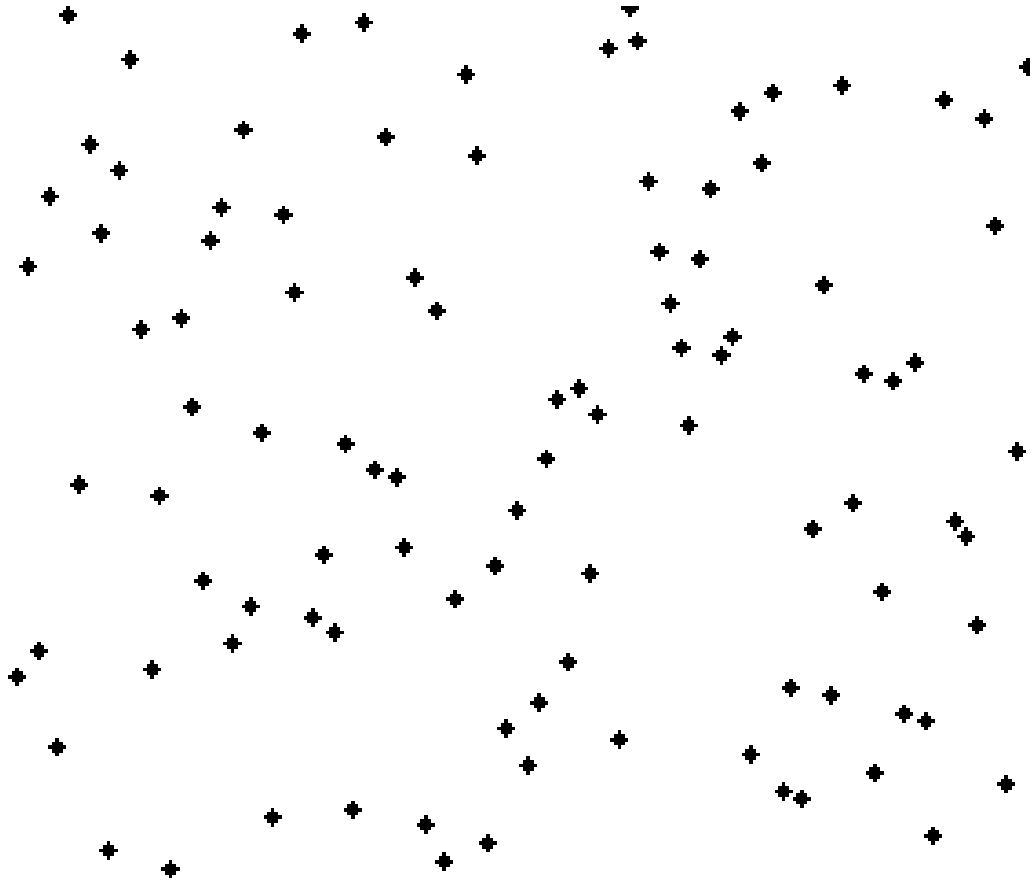


Pseudo Código Quick Sort

Análises

Bubble Sort

Ordenção por flutuação



Motivação:

Bubble sort ou ordenação por flutuação é uma das formas mais simples de ordenação de vetores ou listas dinâmicas. Com ele, Comparam-se dois elementos e trocam-se suas posições se o segundo elemento é menor do que o primeiro.

Vantagens: Simples implementação, sendo poucos trechos de código e fácil de entender, e estabilidade;

Desvantagens:

- **O bubble sort possui um alto custo de processamento, pois tem um crescimento exponencial conforme o número de elementos aumenta;**
- **São feitas várias passagens pela tabela.**

Indicações:

- **Tabelas muito pequenas;**
- **Quando sabe-se que a tabela está quase toda ordenada;**
- **Facilidade para fins didáticos.**

Complexidade:

$O(n^2)$

Home | Recent Tabs | Funcionamento Bub...

https://www.youtube.com/watch?v=i6PRSM3kbts

Bubble Sort - [Sort Dance]

1.037 visualizações • 14 de dez. de 2019

17 4 COMPARTILHAR SALVAR ...

Todos | Enviados recentemente

SLIDE NETFLIX - TUTORIAL
Marcelo_Setubal
2,3 mil visualizações • há 3 dias
Novo

lofi hip hop radio - beats to relax/study to
Lofi Girl
31 mil assistindo
AO VIVO AGORA

CHILL RADIO 24/7
the bootleg boy 2
1,7 mil assistindo
AO VIVO AGORA

Abstract Art Speed Green light and Stripes Background...
MG1010
552 mil visualizações • há 3 meses

Léo Lins faz stand-up em voo 5 horas atrasado e com...
Léo Lins
4,2 mi de visualizações • há 5 anos

Bubble sort dance - informatika unpas
bringITonUNPAS
1,7 mil visualizações • há 8 meses

TED - Como falar de um jeito que as pessoas queiram ouvir

Código Bubble Sort

```
funcao inicio()
  {
    inteiro vetor[5]


    vetor[0] = 19
    vetor[1] = 7
    vetor[2] = 13
    vetor[3] = 15
    vetor[4] = 3

    escreva ("\nVetor não ordenado: ")
    para (inteiro i = 0; i<5; i++){
       se(i == 4){
          escreva (vetor[i])
       }senao{
          escreva (vetor[i], " - ")
       }
```

Cria e preenche o vetor com números pré-definidos

Apresenta o valor de forma não ordenada

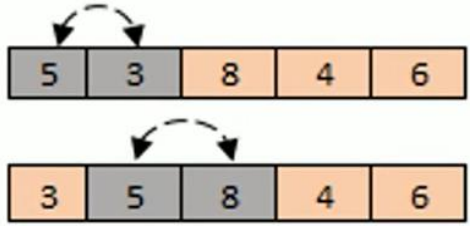
Código Bubble Sort

```
para(inteiro y=4; y>=0; y--){
    para (inteiro x=0; x<y; x++){
        se(vetor[x] > vetor[x+1]){
            inteiro aux = vetor[x]
            vetor[x]   = vetor[x+1]
            vetor[x+1] = aux
        }
    }
}

escreva ("\nVetor ordenado: ")
para (inteiro i = 0; i<5; i++){
    se(i == 4){
        escreva (vetor[i])
    }senao{
        escreva (vetor[i], " - ")
    }
}
```

1º Para: decrementa o tamanho total do vetor

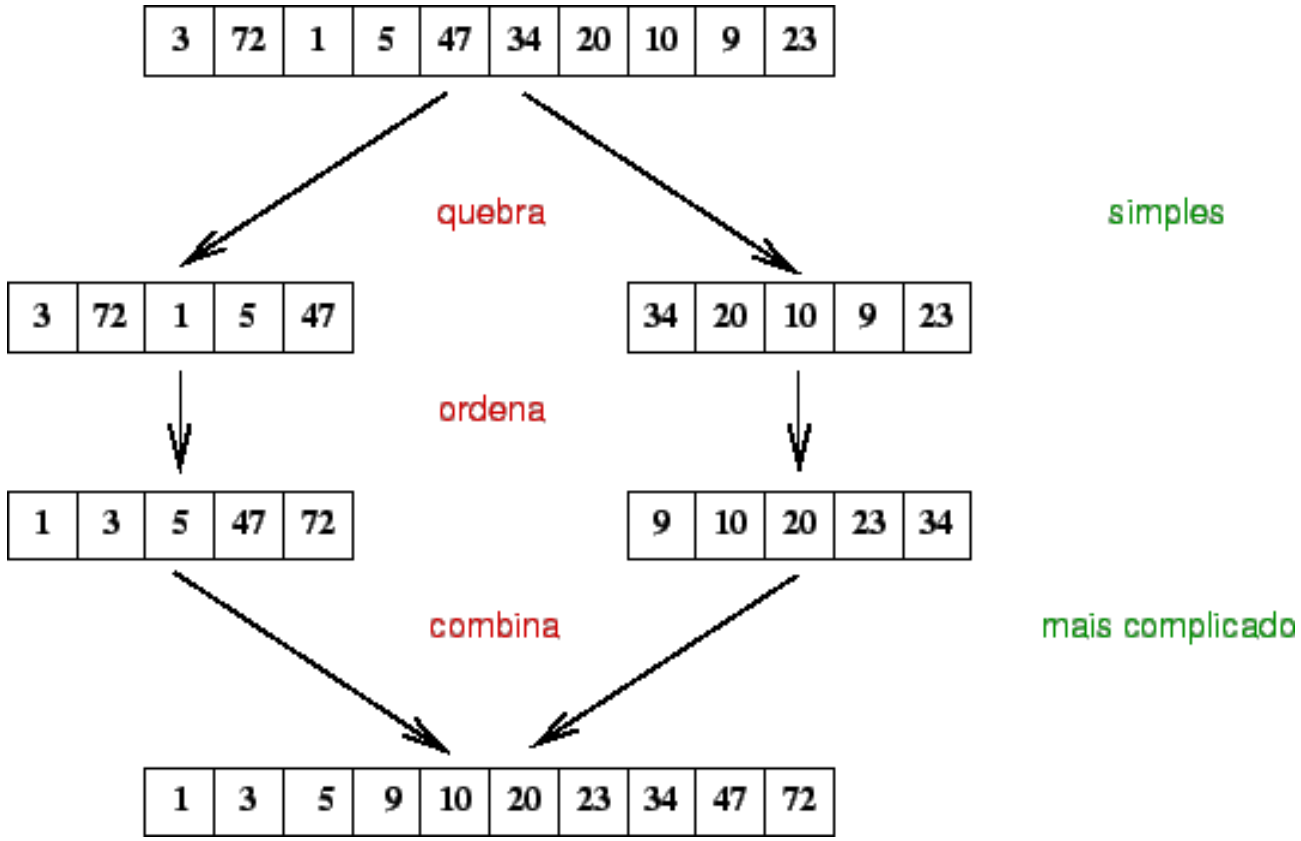
2º Para: Joga o valor mais alto do vetor para o final

Verifica se o valor atual do vetor é maior que o valor do proximo. Se maior, ele troca o valor do atual pelo próximo, se não, incrementa o índice do vetor



Apresenta novamente o vetor, porém desta vez já ordenado.

Merge Sort



Motivação:

O Merge Sort é um algoritmo muito eficiente que tem como lema, "dividir e conquistar". A ordenação de uma Array é feita a partir de uma comparação do ponto de vista assintótico utilizando: n * log n (Brunet, 2019).

Vantagens:

- **A eficiência do Merge Sort é muito boa no melhor, pior e medio caso;**
- **A complexidade do Merge Sort não depende da sequência de entrada;**
- **Facil implementação;**

Desvantagens:

- **O Merge Sort requer um vetor auxiliar, aumentando o uso da memória e tempo de execução;**
- **Faz o uso de funções recursivas;**

Home | Recent Tabs | Funcionamento Mer...

https://www.youtube.com/watch?v=4VqmGXwpLqc

Merge sort in 3 minutes

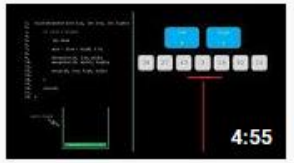
415.374 visualizações • 30 de jul. de 2016

6,2 MIL 401 COMPARTILHAR SALVAR

Todos | Da sua pesquisa | Algoritmos de orde

Heap sort in 4 minutes
Michael Sambol
433 mil visualizações • há 5 anos

Notação Big O em 5 Minutos - Introdução
Michael Sambol
482 mil visualizações • há 4 anos



mergeSort(): A Graphical, Recursive, C++ Explanation
Dylan Sallee
54 mil visualizações • há 4 anos

15 Sorting Algorithms in 6 Minutes
Timo Bingmann
15 mi de visualizações • há 8 anos

Copa do Brasil - Quartas de final - Rainbow Six Siege
Rainbow Six Esports Brasil
300 assistindo
AO VIVO AGORA

a novela carrossel era o shitpost da tv brasileira
Maicon Küster
779 mil visualizações • há 5 dias
Novo

LUAN SANTANA - Flow Podcast #432
Flow Podcast

Pseudo Código Merge Sort

1. Primeiro passo para a execução de um código Merge Sort é dividir o vetor em subvetores até alcançar o caso base

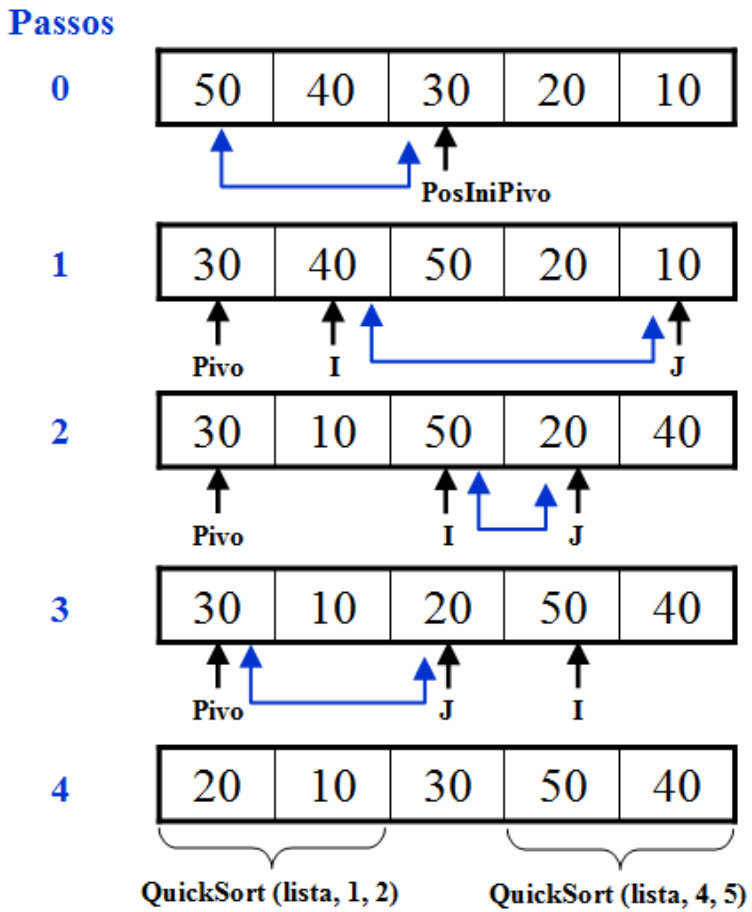
Busca:

```
01. mergesort(A[0...n - 1], inicio, fim)
02. |   se(inicio < fim)
03. |   |   meio ← (inicio + fim) / 2 //calcula o meio
04. |   |   mergesort(A, inicio, meio) //ordena o subvetor esquerdo
05. |   |   mergesort(A, meio + 1, fim) //ordena o subvetor direito
06. |   |   merge(A, inicio, meio, fim) //funde os subvetores esquerdo e direito
07. |   fim_se
08. fim_mergesort
```

```
01. merge(A[0...n - 1], inicio, meio, fim)
02. |   tamEsq ← meio - inicio + 1 //tamanho do subvetor esquerdo
03. |   tamDir ← fim - meio //tamanho do subvetor direito
04. |   inicializar vetor Esq[0...tamEsq - 1]
05. |   inicializar vetor Dir[0...tamDir - 1]
06. |   para i ← 0 até tamEsq - 1
07. |   |   Esq[i] ← A[inicio + i] //elementos do subvetor esquerdo
08. |   fim_para
09. |   para j ← 0 até tamDir - 1
10. |   |   Dir[j] ← A[meio + 1 + j] //elementos do subvetor direito
11. |   fim_para
12. |   idxEsq ← 0 //índice do subvetor auxiliar esquerdo
13. |   idxDir ← 0 //índice do subvetor auxiliar direito
```

```
14. |   para k ← inicio até fim
15. |   |   se(idxEsq < tamEsq)
16. |   |   |   se(idxDir < tamDir)
17. |   |   |   |   se(Esq[idxEsq] < Dir[idxDir])
18. |   |   |   |   |   A[k] ← Esq[idxEsq]
19. |   |   |   |   |   idxEsq ← idxEsq + 1
20. |   |   |   |   senão
21. |   |   |   |   |   A[k] ← Dir[idxDir]
22. |   |   |   |   |   idxDir ← idxDir + 1
23. |   |   |   |   fim_se
24. |   |   |   senão
25. |   |   |   |   A[k] ← Esq[idxEsq]
26. |   |   |   |   idxEsq ← idxEsq + 1
27. |   |   |   fim_se
28. |   |   senão
29. |   |   |   A[k] ← Dir[idxDir]
30. |   |   |   idxDir ← idxDir + 1
31. |   |   fim_se
32. |   fim_para
33. fim_merge
```

Quick Sort

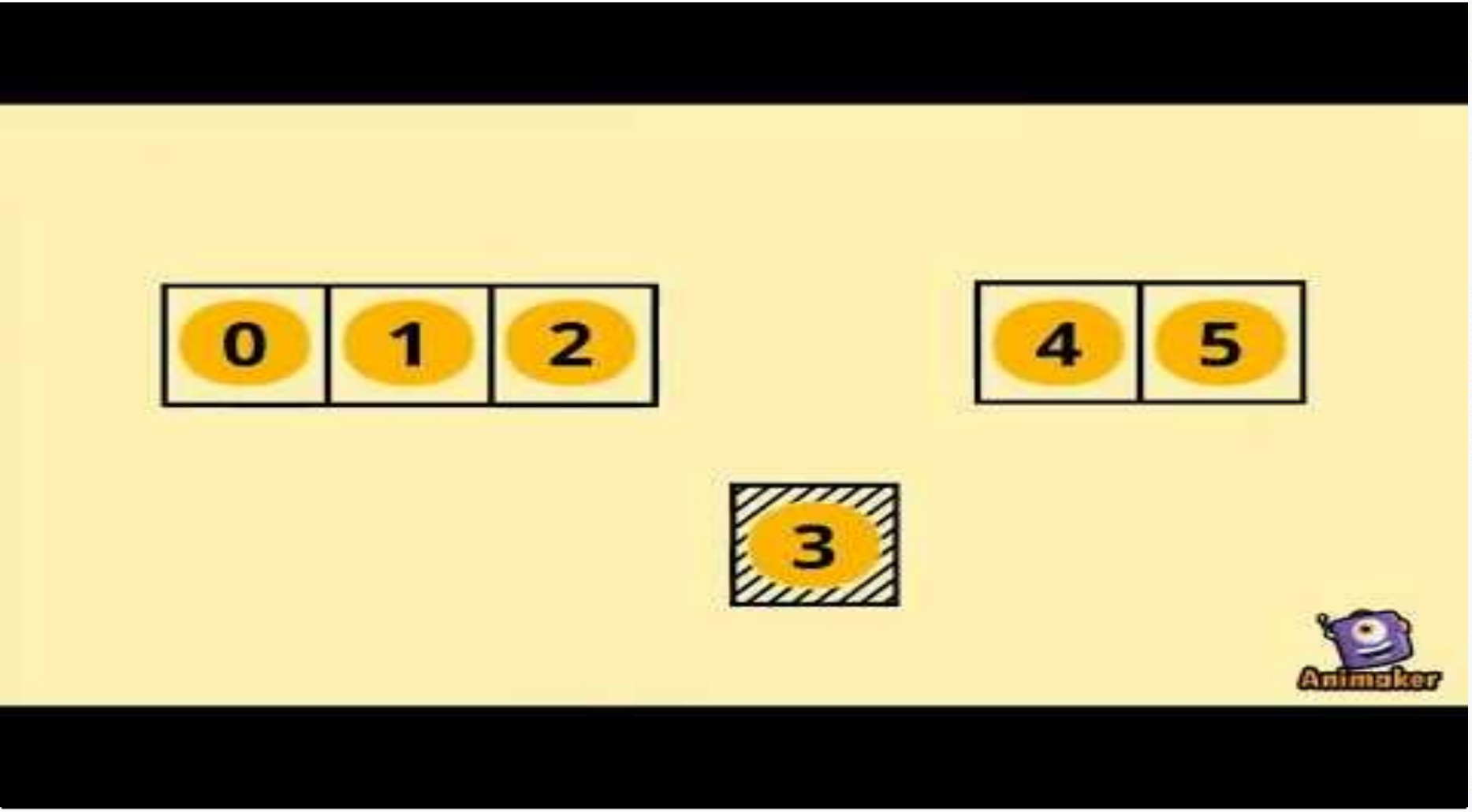


Motivação:

O Quick Sort assim como o Merge Sort é um algoritmo recursivo. Considerado o mais rápido algoritmo de ordenação interna para uma ampla variedade de situações e diferentemente da divisão e conquista do Merge Sort onde é no passo da conquista que tudo acontece, no Quick Sort todo o trabalho é realizado na divisão, outra grande diferença é que ele funciona localmente e seu tempo de execução no pior dos casos é $O(n^2)$, mas ao selecionar corretamento o pivô seu caso chega a ser tão bom quanto o Merge, sendo $O(n * \log n)$.

Home | Recent Tabs | Funcionamento Quick...

https://www.youtube.com/watch?v=WP7KDljG6IM



Lógica do algoritmo de ordenação quick sort

1.079 visualizações • 16 de fev. de 2018

39 0 COMPARTILHAR SALVAR ...

Quicksort: Partitioning an array
KC Ang
421 mil visualizações • há 6 an

Quick-sort with Hungarian (Küküllőmenti legényes) fo
AlgoRythmics
2,1 mi de visualizações •
há 10 anos

Crafting the swords of the ancient kings with artistic
Barton Alethea
7,2 mil visualizações • há 8 hor
Novo

Notação Big O em 5 Minut Introdução
Michael Sambol
482 mil visualizações • há 4 an

Sorts 8 Quick Sort
RobEdwards
137 mil visualizações • há 4 an

Motivação para estudar | Clóvis de Barros Filho |...
Revista INSPIRE-C
375 mil visualizações • há 3 me

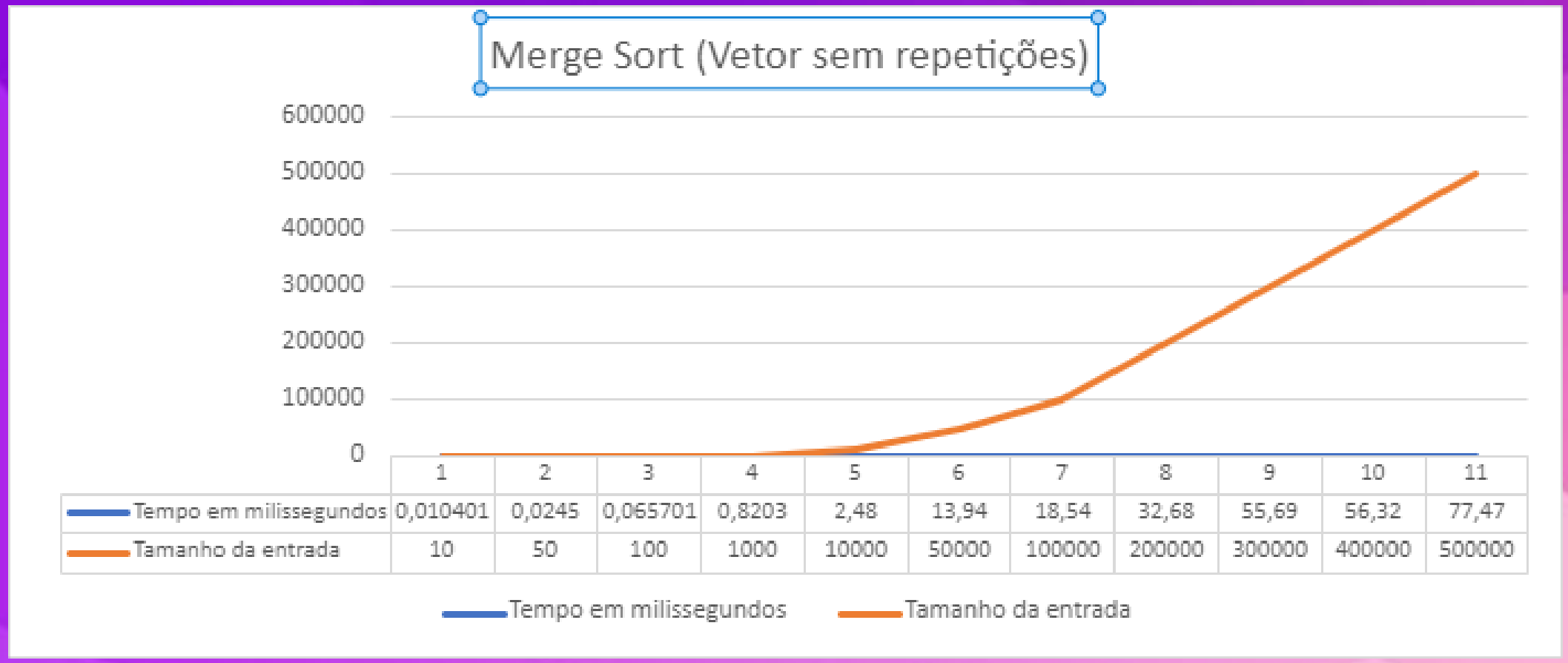
Pseudo Código para Quick Sort

```
QuickSort(A como Array, left como int, right como int)
        se(left < right)
                posicaoPivot = Particao(A, left, right)
                QuickSort(A, left, posicaoPivot - 1)
                QuickSort(A, posicaoPivot + 1, right)
        fim_se

Particao(A, left como int, right como int)
        pivot = A[left]
        left_wall = left

        para i <- left + 1 até right
                se(A[i] < pivot)
                        troca(A[i],A[leftwall])
                        leftwall = leftwall + 1
                fim_se
        fim_para

retorne(leftwall)
```

Busca: Texto...

Implementação prática do Merge Sort

Testes empíricos realizados no Merge Sort

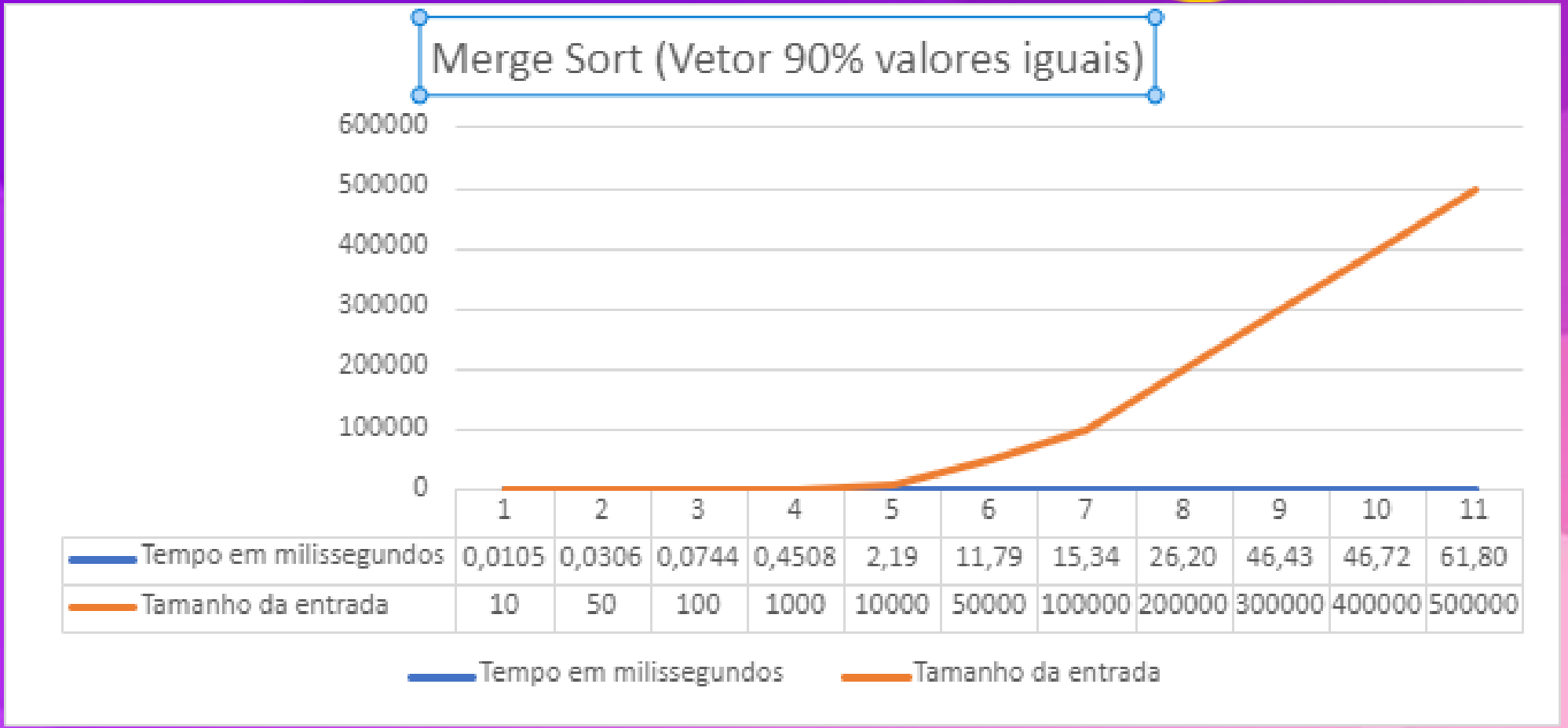


	1	2	3	4	5	6	7	8	9	10	11
Tempo em milissegundos	0,010401	0,0245	0,065701	0,8203	2,48	13,94	18,54	32,68	55,69	56,32	77,47
Tamanho da entrada	10	50	100	1000	10000	50000	100000	200000	300000	400000	500000

Home | Recent Tabs | Testes Merge Sort

Testes empíricos no Merge Sort

Testes empíricos realizados no Merge Sort



	1	2	3	4	5	6	7	8	9	10	11
Tempo em milissegundos	0,0105	0,0306	0,0744	0,4508	2,19	11,79	15,34	26,20	46,43	46,72	61,80
Tamanho da entrada	10	50	100	1000	10000	50000	100000	200000	300000	400000	500000

Testes empíricos realizados no Merge Sort

Análise: A partir dos resultados obtidos podemos ver que o algoritmo Merge Sort é eficiente em sua ordenação através do dividir e conquistar tendo para seu pior, melhor e caso médio uma eficiência de $n*\log n$, garantindo que independente da forma como estiver disposto a array (seja por um vetor aleatório e sem repetições ou para um com 90% de dados repetidos) a sua ordenação será sempre eficiente.

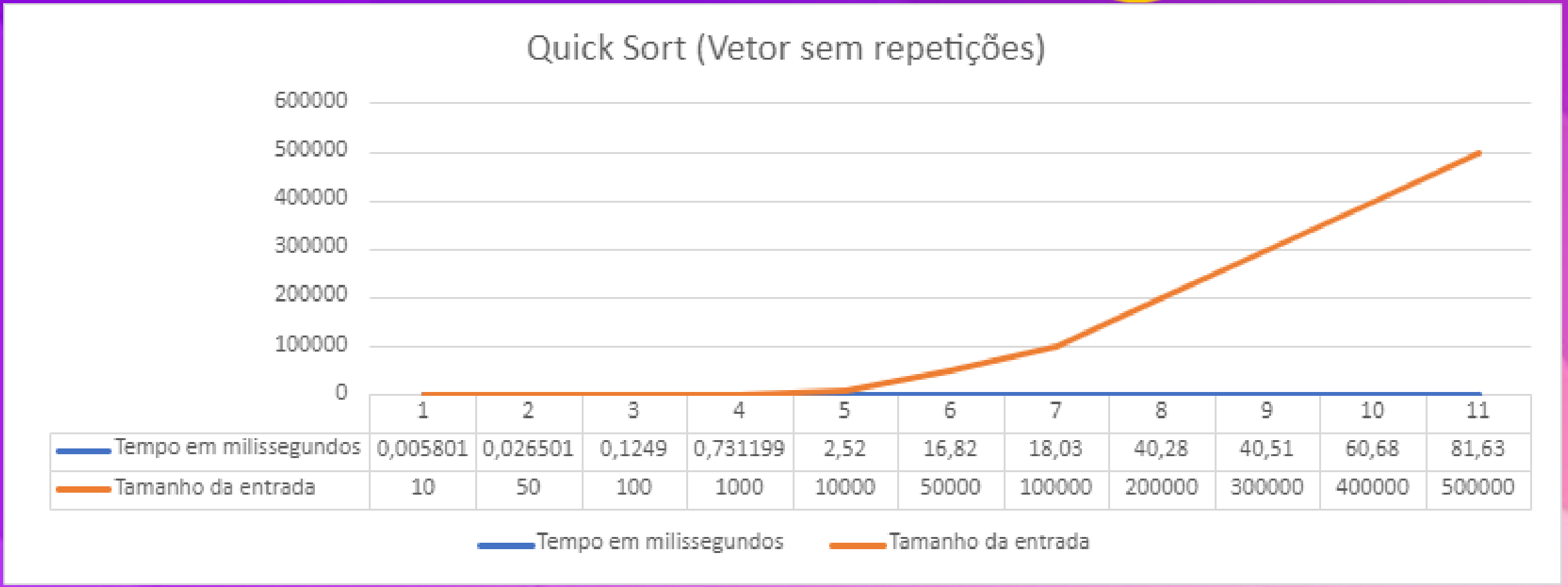
Mas claro que por conta de 90% do vetor ser valores iguais, facilita a velocidade de ordenação e portanto, seu tempo em relação a um vetor sem repetições e com números aleatórios é menor.

Implementação prática do Quick Sort

Home | Recent Tabs | Testes Quick Sort

Testes empíricos no Quick Sort

Testes empíricos realizados no Quick Sort

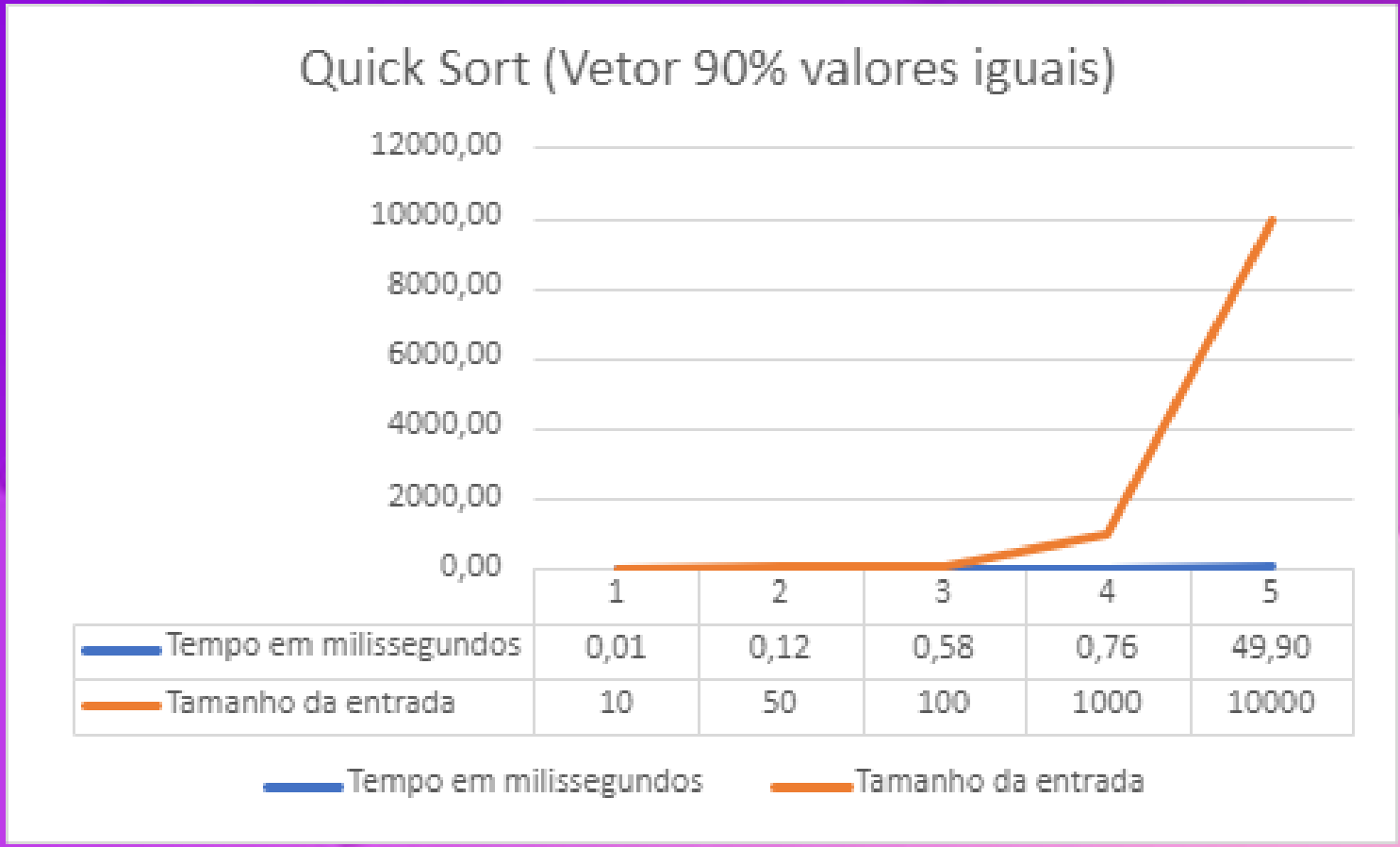


	1	2	3	4	5	6	7	8	9	10	11
Tempo em milissegundos	0,005801	0,026501	0,1249	0,731199	2,52	16,82	18,03	40,28	40,51	60,68	81,63
Tamanho da entrada	10	50	100	1000	10000	50000	100000	200000	300000	400000	500000

Home | Recent Tabs | Testes Quick Sort

Testes empíricos no Quick Sort

Testes empíricos realizados no Quick Sort



	1	2	3	4	5
Tempo em milissegundos	0,01	0,12	0,58	0,76	49,90
Tamanho da entrada	10	50	100	1000	10000

Testes empíricos realizados no Quick Sort

Análise: Levando em consideração ao que foi gerado a partir dos testes, podemos notar que ele possui uma boa eficiência tanto com vetores de tamanhos menores, como maiores, porém em relação ao Merge Sort possui um tempo de execução melhor em casos de vetores de tamanho pequeno, perdendo para o Merge Sort no momento que vai crescendo o tamanho da array. Possui em seu pior caso um O(n^2), enquanto que o Merge Sort possui n * log n em todos os seus casos.

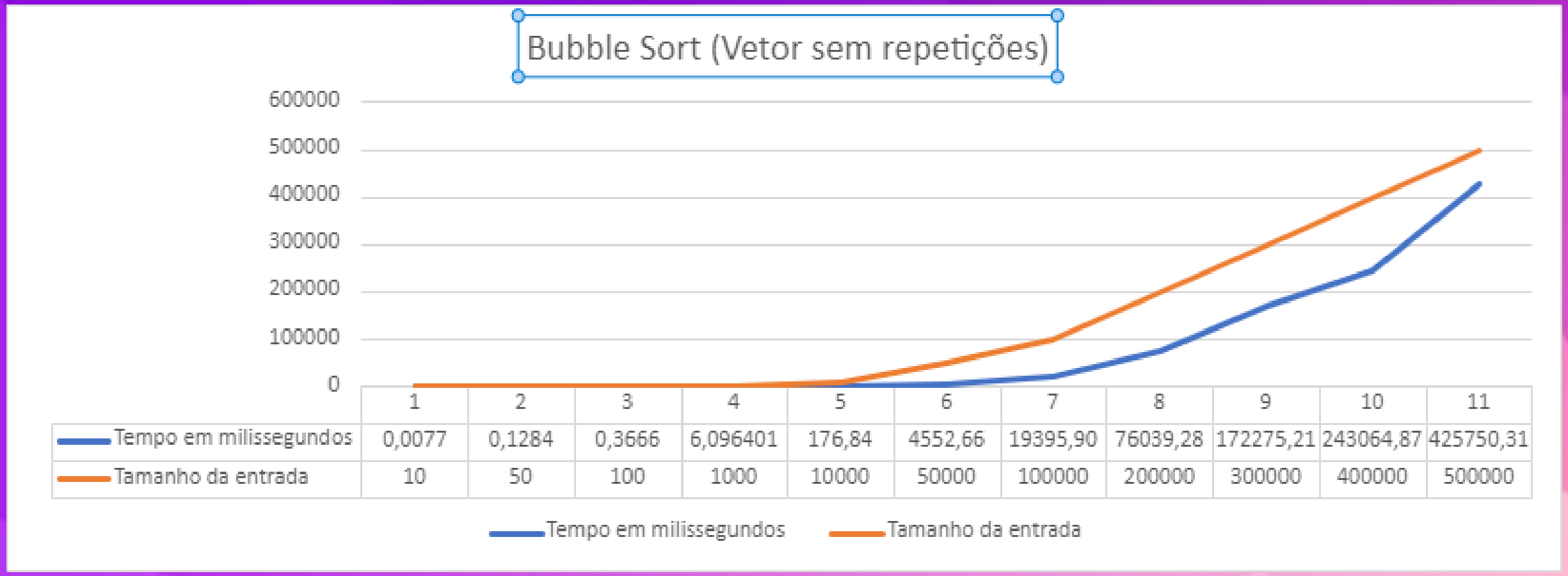
Assim tornando o Quick Sort um poderoso algoritmo de ordenação em casos de vetores com menor tamanho.

Implementação prática do Bubble Sort

Home | Recent Tabs | Testes Bubble Sort

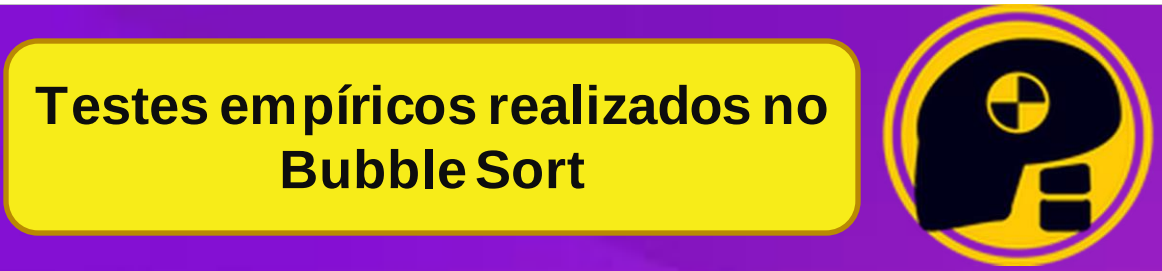
Testes empíricos no Bubble Sort

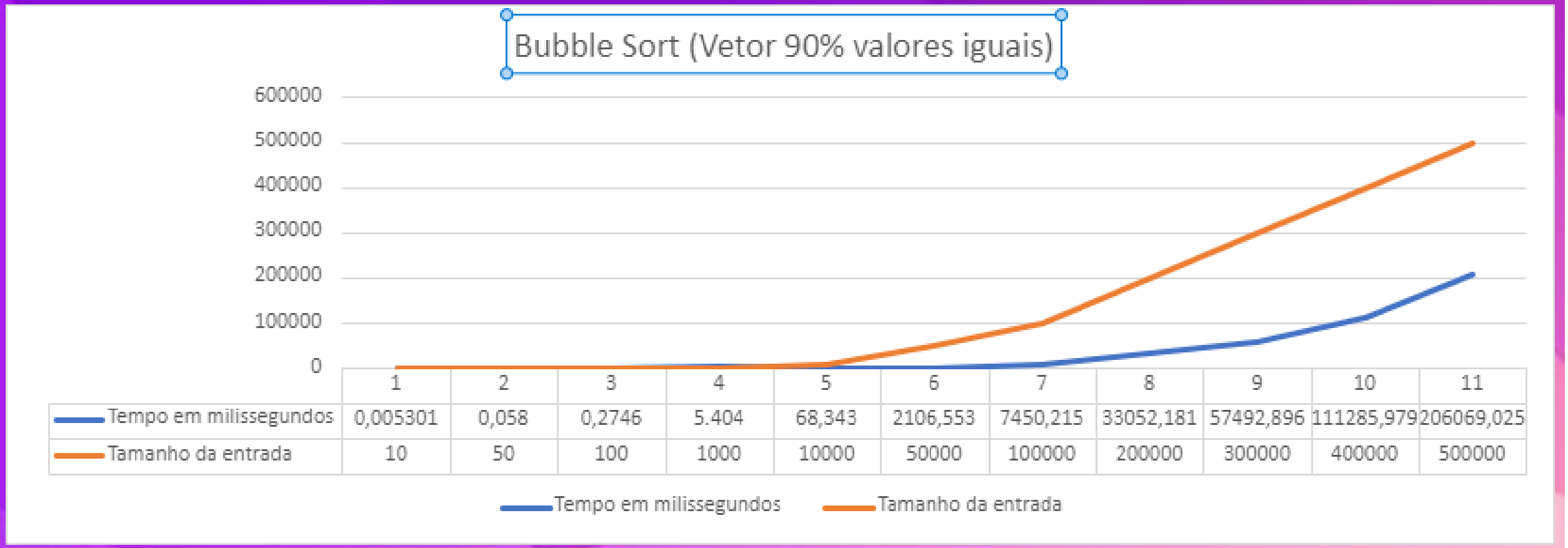
Testes empíricos realizados no Bubble Sort



	1	2	3	4	5	6	7	8	9	10	11
Tempo em milissegundos	0,0077	0,1284	0,3666	6,096401	176,84	4552,66	19395,90	76039,28	172275,21	243064,87	425750,31
Tamanho da entrada	10	50	100	1000	10000	50000	100000	200000	300000	400000	500000

Testes empíricos realizados no Bubble Sort





	1	2	3	4	5	6	7	8	9	10	11
Tempo em milissegundos	0,005301	0,058	0,2746	5.404	68,343	2106,553	7450,215	33052,181	57492,896	111285,979	206069,025
Tamanho da entrada	10	50	100	1000	10000	50000	100000	200000	300000	400000	500000

Testes empíricos realizados no Bubble Sort

Análise: Com os testes realizados podemos notar que em relação aos outros algoritmos de ordenação o Bubble Sort possui um tempo de execução muito mais alto independente de ser com um vetor com números aleatório e sem repetições ou com um vetor de 90% de números repetidos.

Esse tempo de execução alto acontece por conta de em seu pior caso e caso médio serem O(n^2) (no Quick Sort o pior caso é O(n^2)) e melhor é O(n), além de sua complexidade ser de ordem quadrática, ou seja, apresenta um crescimento quadrático na hora de ordenar um vetor.

CONCLUSÃO

- Com todos estes testes podemos portanto, gerar uma classificação dos três algoritmos de ordenação, em que o Merge Sort fica em primeiro, por possuir um tempo de execução menor em praticamente todos os testes, além de ter uma eficiência boa independente do caso.
- Em segundo lugar temos o Quick Sort que fica atrás do Merge Sort por conta de seus tempos serem menores somente quando o tamanho do vetor era menor, além de possuir uma eficiência inferior ao do Merge Sort.
- Por último temos o Bubble Sort, possuindo um tempo de execução ridiculamente maior do que os outros dois algoritmos, envolvendo uma eficiência abaixo dos outros também.

Home | Recent Tabs | Option J

You can type something here...

Referências:

LEANDER, Rick. Advantages & Disadvantages of Bubble Sort. Techwalla. Disponível em: <https://www.techwalla.com/articles/advantages-disadvantages-of-bubble-sort>
Acesso em: 04 de Agosto de 2021.

BRUNET, João. Ordenação por Comparação: Quick Sort. GitHub, 2019. Disponível em: <https://joaoarthurbm.github.io/eda/posts/quick-sort/>
Acesso em: 04 de Agosto de 2021.

BRUNET, João. Ordenação por Comparação: Merge Sort. **Github**, 2019. Disponível em: <https://joaoarthurbm.github.io/eda/posts/merge-sort/>
Acesso em: 04 de Agosto de 2021.

SAMBOL, Michael. Merge sort in 3 minutes. **Youtube**, 30 de jul. de 2016. Disponível em:<https://www.youtube.com/watch?v=4VqmGXwpLqc>
Acesso em: 04 de Agosto de 2021.

NETO, Nelson. Análise de Algoritmos Algoritmos de Ordenação. Unifap, **2016**. Disponível em: <https://www2.unifap.br/furtado/files/2016/11/Aula4.pdf>
Acesso em: 04 de Agosto de 2021.

FELIPE, Henrique. Merge Sort. **Blog Cyberini**, 2018. Disponível em: <https://www.blogcyberini.com/2018/07/merge-sort.html>
Acesso em: 11 de Agosto de 2021.

Referências:

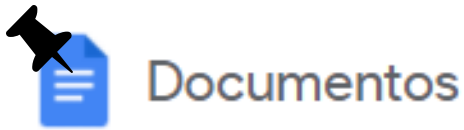
Canal do Código. Como fazer Merge Sort em Java - Canal do Código. **Youtube**, 23 de Setembro de 2016. Disponível em: <https://www.youtube.com/watch?v=yj8igr9DjeY> Acesso em: 11 de Agosto de 2021.

Informatica Nova Cruz. Lógica do algoritmo de ordenação quick sort. **Youtube**, 16 de Fevereiro de 22018. Disponível em:<https://www.youtube.com/watch?v=WP7KDljG6IM> Acesso em: 11 de Agosto de 2021.

https://pt.khanacademy.org/computing/computer-science/algorithms/quick-sort/a/overview-of-quicksort

Home | Recent Tabs | Ata das Atividades

You can type something here...



Ata das Atividades:

https://docs.google.com/document/d/1OCLomTJjT2QqFeon5-we4idYRIR4CjkbM9DLcniTSY8/edit