# Análise de complexidade de algoritmos de ordenação

**1. Analise as complexidades de tempo (melhor e pior caso) dos algoritmos**

**(i) bubblesort, (ii) mergesort e (iii) quicksort. Cite as complexidades de**

**tempo de cada um dos algoritmos e explique como cada complexidade**

**foi calculada por você (2,0)**

O melhor caso do Quicksort ocorre quando as partições são balanceadas, ou seja, cada partição possui n/2 elementos. A equação de recorrência neste caso é igual a Mergesort , T(n)=2T(n/2)+Θ(n)(melhor caso) ,Θ(n) é a complexidade de tempo do método de partição. A solução desta recorrência é T(n)=Θ(n log n) ou, equivalentemente, T(n)=O(log n). O pior caso do Quicksort ocorre quando chamadas recursivas produzem partições com 0 e n−1 elementos. A partição de tamanho zero está à direita ou à esquerda do pivô (dependente do vetor). A equação de recorrência deste caso T(n)=T(n−1)+Θ(n)(pior caso) A solução desta recorrência é T(n)=O(n2).

O mergesort Independente do caso (melhor, pior ou médio) ele sempre será n∗log⁡ n. Isso ocorre porque a divisão do problema sempre gera dois sub-problemas com a metade do tamanho do problema original (2∗T(n/2)).

O melhor caso do bubblesort tem O (n), tem como seu pior caso quando a lista está em ordem reversa, nesse caso, no loop inicial o item que estava no posição 0 terminará na posição n-1 então temos complexidade T(n) = O (n2).

**3. Faça um experimento em que cada um dos algoritmos é executado**

**considerando 3 cenários, em que cada um ordena um vetor de números**

**inteiros aleatórios contendo: (i) 50000, (ii) 100000 e (iii) 200000**

**elementos. Execute cada algoritmo 30 vezes em cada cenário e extraia a**

**média do tempo de execução, em segundos, para cada cenário. Os**

**números aleatórios devem ser gerados dentro de um intervalo entre 0 e**

**10000 (5,0)**

**a) Mostre a média de cada algoritmo em cada cenário na sua resposta.**

**(1,0)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | MÉDIA | BubbleSort | MergeSort | QuickSort |  |
|  | 50000 | 4,404 s | 0,011 s | 0,0072 s |  |
|  | 100000 | 1731,06456 s | 0,0125 s | 0,0176 s |  |
|  | 200000 | 69,52556 s | 0,0402 s | 0,030433 s |  |
|  |  |  |  |  |  |

**b) Explique a relação existente entre o aumento do tamanho do vetor e o**

**aumento do tempo de execução (1,5)**

o bubblesort e por causa dessa forma de execução, o vetor terá que ser percorrido quantas vezes for necessário, tornando o algoritmo ineficiente para listas muito grandes.

A função mergesort requer um vetor auxiliar, o que aumenta o consumo de memória e o tempo de execução.

o quicksort tem sua complexidade no melhor caso ́e Θ(n log2 n). Semelhante ao MergeSort, mas necessitada apenas de uma pequena pilha como memória auxiliar o que se torna melhor e mais rápido mesmo com um vetor grande .

**c) Qual a análise que pode ser feita entre as respostas ao item 1 e ao**

**item 3.b? (1,5)**

Que o algoritmo quicksort e todas as situações entre vetores menores e maiores ele se sair o mais eficaz mesmo em um grande quantidade de números para ordenar .

**d) No caso de algoritmos de mesma complexidade, o que diferencia o**

**tempo de execução dos dois no seu experimento? (1,0)**

Por que no bubblesort ele tem que percorrer o vector diversas vezes , já no quicksort ele não precisa estar percorrendo o vetor levando isso ao quicksort se mais eficiente .