UNIVERSIDADE CATÓLICA DE BRASÍLIA
CIÊNCIA DA COMPUTAÇÃO

ALICE FERREIRA DE ANDRADE

ALGORITMOS DE ORDENAÇÃO

Brasília - DF
2024

SUMÁRIO

1. INTRODUÇÃO

Algoritmos de ordenação são algoritmos que foram desenvolvidos para ordenar uma lista de itens de determinada maneira. Eles permitem o funcionamento de outros algoritmos, como os de busca, além de facilitarem a leitura e a organização de dados. Listas de dados são ordenadas principalmente em ordem numérica crescente ou em ordem alfabética.

Uma importante classificação para algoritmos de ordenação, assim como para outros algoritmos, é a complexidade de tempo, comumente medida na notação *big* O, que permite a análise em pior caso do tempo de execução de determinado algoritmo.

O estudo de algoritmos de ordenação é importante, por serem ferramentas de grande utilidade e muitas vezes essenciais para os mais diversos tipos de programas, além de serem conceitos fundamentais da programação.

2. MERGE

O algoritmo Merge Sort baseia-se na mesclagem de duas listas já ordenadas para obter uma lista inteira ordenada. Inicialmente, a lista é repetidamente dividida ao meio até que se obtenham sublistas unitárias, que são naturalmente ordenadas. Em seguida, os itens das sublistas são comparados e inseridos em uma nova lista ordenada. Após a ordenação de cada dupla de sublistas, o processo se repete com duplas maiores, onde os itens da extrema esquerda da primeira sublista da dupla serão comparados com os itens da extrema esquerda da segunda, até que reste apenas uma lista ordenada.

A complexidade do Merge Sort é O(n log n) em todos os casos.

3. QUICK SORT

O Quick Sort é um algoritmo de ordenação no qual é estabelecido arbitrariamente ou aleatoriamente um item pivô na lista, com o qual serão comparados os itens da lista. Então, a lista é rearranjada de maneira que todos os elementos menores que o pivô fiquem à sua esquerda, e todos os maiores fiquem à sua direita. Esse processo é repetido recursivamente nas sublistas criadas, até que elas tenham um ou zero elementos, o que significa que elas já estão ordenadas. Depois disso, as sublistas ordenadas serão combinadas para formar uma lista inteira ordenada.

O Quick Sort tem uma complexidade de aproximadamente O(n log n), mas dependendo da escolha do pivô pode chegar a $O(n^2)$, no pior caso.

4. SHELL SORT

Para explicar esse algoritmo de ordenação, primeiramente teremos uma breve explicação do Insertion Sort:

No Insertion Sort, a lista é dividida em uma parte ordenada e uma parte não ordenada. O algoritmo percorre a lista não ordenada, pegando um elemento por vez e inserindo-o na posição correta na parte ordenada. Essa inserção é feita comparando com os elementos já ordenados, movendo-os para a direita se necessário.

No Shell Sort a lista é dividida em sublistas, utilizando intervalos (ou *gaps*), que irão diminuindo a cada iteração. Depois, os elementos dessas sublistas serão ordenados pelo método da inserção. Quando o *gap* se torna 1, a lista será ordenada novamente pelo método da inserção, e então, a lista estará ordenada.

A complexidade de tempo desse algoritmo pode variar a depender do tamanho do *gap*, mas de maneira geral costuma ser O(n log n).

5. HEAP SORT

Para esse algoritmo de ordenação, a lista é organizada em uma estrutura de dados chamada Heap, especificamente Heap Máximo, que é uma árvore binária onde o maior elemento fica no topo e cada elemento, ou nó pai é maior ou igual aos seus nós filhos.

Após a organização em Heap, o maior elemento é trocado com o último elemento da lista, e a lista é restaurada para manter suas propriedades de Heap. Esse processo se repete até que todos os elementos sejam extraídos e posicionados no fim da lista, e a lista esteja ordenada.

A complexidade de tempo desse algoritmo é O(n log n) em todos os casos.

6. CONCLUSÃO

Com isso, vemos tanto a complexidade de criação como a complexidade de tempo de cada um desses algoritmos, podendo ter uma visão mais ampla de seus funcionamentos e situações onde seu uso é mais ou menos adequado. O conhecimento de algoritmos de ordenação é de suma importância para a implementação eficiente em programas, extraindo o melhor de cada um.

7. REFERÊNCIAS BIBLIOGRÁFICAS

THE ART of Computer Programming: Sorting and Searching. 2. ed. [*S. l.*: *s. n.*], 1998. Disponível em: https://seriouscomputerist.atariverse.com/media/pdf/book/Art%20of%20Computer%20Programming%20-%20Volume%203%20(Sorting%20&%20Searching).pdf. Acesso em: 19 set. 2024.

INTRODUCTION to Algorithms. [*S. l.*: *s. n.*], 1989. Disponível em: https://dl.ebooksworld.ir/books/Introduction.to.Algorithms.4th.Leiserson.Stein.Rivest.Cormen.MIT.Press.9780262046305.EBooksWorld.ir.pdf. Acesso em: 19 set. 2024.

PERFORMANCE Comparison between Merge and Quick Sort Algorithms in Data Structure. West Yorkshire: Science and Information (SAI) Organization Limited, [*s. l.*], 1 jan. 2018. Disponível em: https://redlands.primo.exlibrisgroup.com/view/action/uresolver.do?operation=resolveService&package_service_id=6577303890003896&institutionId=3896&customerId=3895&VE=true. Acesso em: 19 set. 2024.