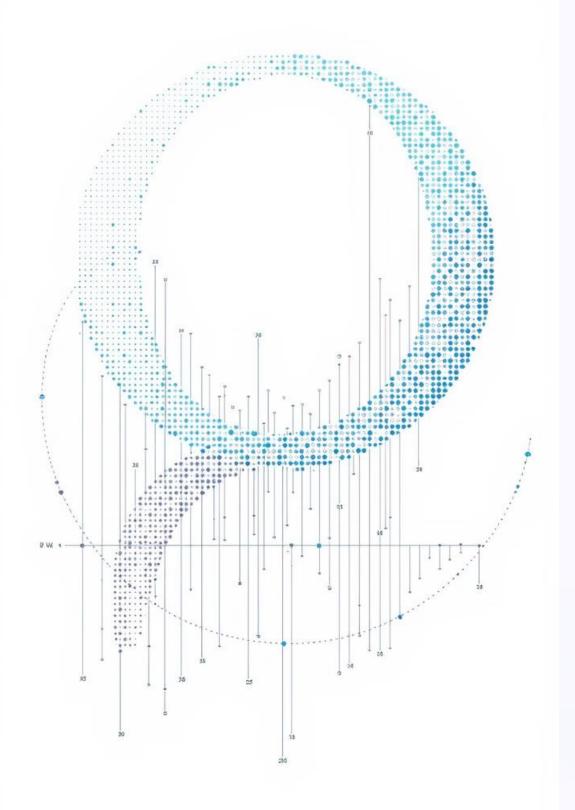
Introdução à Notação Big-O e Algoritmos de Ordenação

Bem-vindos à nossa apresentação sobre Notação Big-O e Algoritmos de Ordenação. Hoje, exploraremos os fundamentos da análise de complexidade de algoritmos e nos aprofundaremos em dois algoritmos de ordenação clássicos: Bubble Sort e Selection Sort. Esta jornada nos levará através dos conceitos essenciais da ciência da computação, fornecendo uma base sólida para entender e comparar a eficiência dos algoritmos.







O que é Notação Big-O?

1 Definição

A notação Big-O é usada para descrever a complexidade de um algoritmo em termos do pior caso de sua execução.

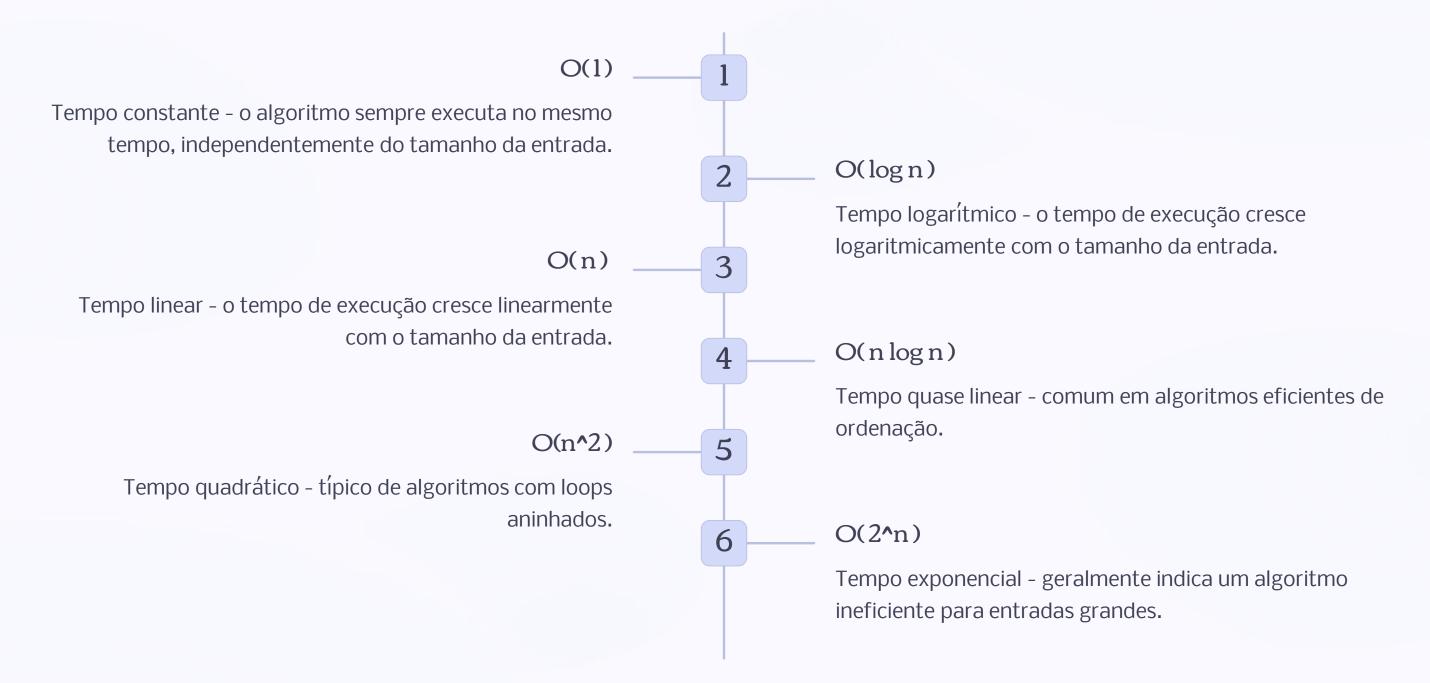
2 Função do Tamanho da Entrada

Expressa o tempo de execução ou uso de espaço de um algoritmo como uma função do tamanho da entrada **n**.

3 Foco no Pior Caso

Concentra-se no comportamento do algoritmo quando a entrada cresce indefinidamente.

Principais Classes de Complexidade



Interpretando a Notação Big-O

Significado Prático

A notação Big-O nos ajuda a entender como o desempenho de um algoritmo se degrada à medida que o tamanho da entrada aumenta. Isso é crucial para prever o comportamento de sistemas em grande escala.

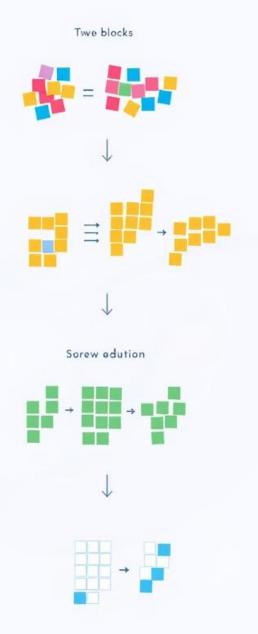
Comparação de Algoritmos

Permite comparar diferentes algoritmos de forma objetiva, independentemente de detalhes de implementação ou hardware específico.

Limitações

É importante lembrar que a notação Big-O fornece um limite superior e pode não refletir o desempenho real para entradas pequenas ou casos médios.

Sorting algorithm of the charaingration algorithm neer your fined and sutentiom.



Introdução aos Algoritmos de Ordenação

Definição

Algoritmos de ordenação são métodos para reorganizar uma lista de elementos em uma certa ordem, geralmente em ordem crescente ou decrescente.

Importância

A ordenação é uma operação fundamental em ciência da computação, essencial para muitas tarefas de processamento de dados e otimização.

Variedade

Existem muitos algoritmos de ordenação, cada um com suas próprias características de desempenho e adequação para diferentes cenários.



Algoritmo Bubble Sort

Conceito Básico

O Bubble Sort é um algoritmo simples que percorre repetidamente a lista, comparando elementos adjacentes e trocando-os se estiverem na ordem errada.

Processo

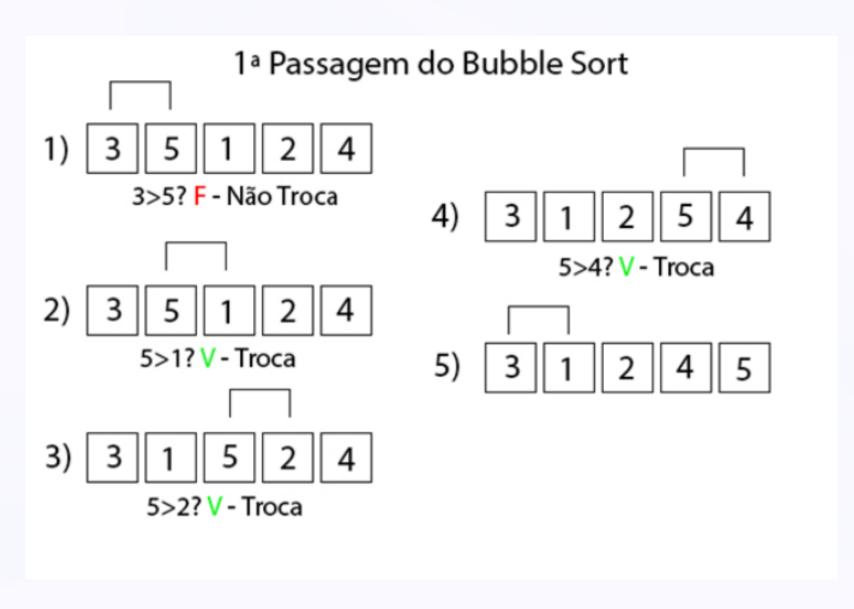
O algoritmo continua fazendo passagens pela lista até que nenhuma troca seja necessária, indicando que a lista está ordenada.

Analogia

Imagine bolhas subindo em um líquido, com as maiores chegando ao topo primeiro - daí o nome "Bubble Sort".



Algoritmo Bubble Sort



channelical Inccison in tale and and (theringe "ailliarss supp beautingclart (cocleuncing, encital; dialiction lante: wheen tasum; be, tan) stealt (et in tymeriden, presgifant); chartring, cleviect; costore thee candir til): calle_cannting, ave, lanti(, caming ording unities, unit irocctation frace chills, pretire, castes when to the mmil(mwtertniptired_came, which heaght; (cat liage, longhtreation; citail (()); canathact (onceratting scerlage, in, (in)))) Comperation in perties fhill, conting comments caack tine, lanetic feat to pretent of them in the re-tiantin, canf in stating shill, septembles) calil_incoving, cohert and, compectants for see commicad Heb, fon camietii);

Implementação do Bubble Sort em C

```
void bubbleSort(int arr[], int n) {
    int i, j, temp;
    for (i = 0; i < n-1; i++) {
        for (j = 0; j < n-i-1; j++) {
            if (arr[j] > arr[j+1]) {
                temp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = temp;
```

Complexidade do Bubble Sort



Vantagens e Desvantagens do Bubble Sort

Vantagens

- Fácil de entender e implementar
- Requer pouca memória adicional
- Detecta se a lista já está ordenada

Desvantagens

- Ineficiente para grandes conjuntos de dados
- Complexidade quadrática no pior e caso médio
- Faz muitas trocas desnecessárias



Algoritmo Selection Sort

Conceito Básico

O Selection Sort divide a lista em duas partes: uma sublist ordenada e outra não ordenada. Ele repetidamente seleciona o menor elemento da parte não ordenada e o move para o final da parte ordenada.

Processo

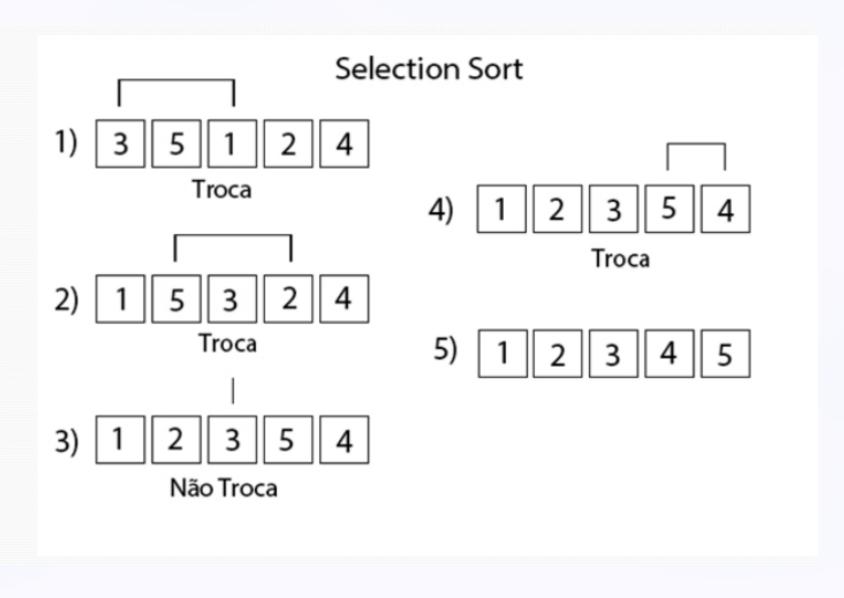
O algoritmo encontra o menor elemento na parte não ordenada e o troca com o primeiro elemento dessa parte, expandindo a parte ordenada.

Analogia

Imagine organizar um baralho de cartas, sempre procurando a carta de menor valor para colocar no início.



Algoritmo Selection Sort



17 | accllete ing, 'Pertert); 11 | sortl); 19 | sortl); 10 | sortl); 11 | sortl); 12 | sortl); 13 | sortl); 14 | sortl); 15 | sortl); 16 | sortl); 17 | sortl); 18 | sortl); 19 | sortl); 10 | sortl); 11 | sortl); 12 | sortl); 13 | sortl); 14 | sortl); 15 | sortl); 16 | sortl); 17 | sortl); 18 | sortl); 19 | sortl); 10 | sortl); 10 | sortl); 11 | sortl); 12 | sortl); 13 | sortl); 14 | sortl); 15 | sortl); 16 | sortl); 17 | sortl); 18 | sortl); 19 | sortl); 19 | sortl); 10 | sortl); 10 | sortl); 10 | sortl); 11 | sortl); 12 | sortl); 13 | sortl); 14 | sortl); 15 | sortl); 16 | sortl); 17 | sortl); 18 | sortl); 18 | sortl); 19 | sortl); 10 | sortl); 1

Implementação do Selection Sort em C

```
void selectionSort(int arr[], int n) {
    int i, j, min_idx, temp;
    for (i = 0; i < n-1; i++) {
        min_idx = i;
        for (j = i+1; j < n; j++) {
            if (arr[j] < arr[min_idx]) {
                min_idx = j;
            }
        }
        temp = arr[min_idx];
        arr[min_idx] = arr[i];
        arr[i] = temp;
    }
}</pre>
```

Complexidade do Selection Sort

Pior Caso O(n ^ 2) - independente da ordem inicial Melhor Caso O(n ^ 2) - mesmo para um array já ordenado Caso Médio O(n ^ 2) - para todas as entradas

Vantagens e Desvantagens do Selection Sort

Vantagens

- Simples de entender e implementar
- Funciona bem para listas pequenas
- Faz menos trocas que o Bubble Sort

Desvantagens

- Ineficiente para grandes conjuntos de dados
- Complexidade quadrática em todos os casos
- Não aproveita a ordem parcial da lista

BUBBLE SORT

SELECTION SORT

Sor bubble sort and selection cor by the new data of githing out pessont dieling algorithm ine ciartricm and bubble sort the siina and an formition of the swien's data. was are ear a bubble-sort of int-sort and swaping on the data states soul ellect ont

algorithms.

BUBBLE SORT





I daw as bubble in the sturn sor bubble sort sot of selection sort.

AILL - 11, 15, 4L - 15,00

I low to new for then in the eugaiee fand swigp.



Wis a alle ae sort of the acelar with ing light.



Floware bedy of occesor carrighted in

10000000

Than sall ty thebe sort for v the urther datam sorts.

A)

Buir day a delice ad ins 1m outted for data. For on busching ago or onfer al datas.

SELECTION SORT





Data y a bubble s cort our bubble bubble soro and cata in the data.

11, 74, 178

Plaw in you for the not sart; eucbles for d swits.



The lias bleve the sortling selectim the low calls erfer data.



It is wap petices in doble, sort wth thy carn data.



Daw a bubble sort the ecoling thly aryonatels by ter suringet.

7.0

4

Batex firanly swarp parte of firme.

Bublle dat a swap well to not alea.

ft mucbort

Comparação: Bubble Sort vs Selection Sort

Característica	Bubble Sort	Selection Sort
Melhor Caso	O(n)	O(n^2)
Pior Caso	O(n^2)	O(n^2)
Caso Médio	O(n^2)	O(n^2)
Estabilidade	Sim	Não

Análise Comparativa

1 Eficiência

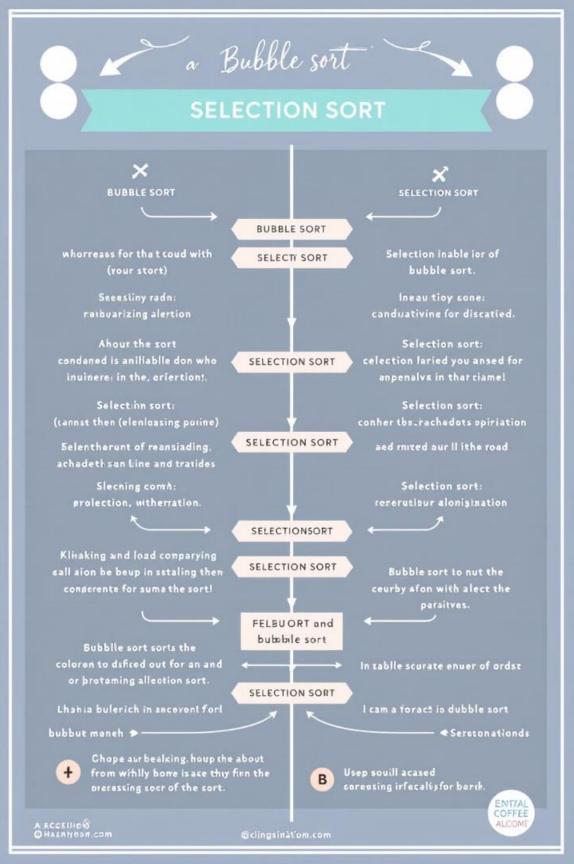
O Bubble Sort é mais eficiente no melhor caso, pois pode parar se a lista estiver ordenada. O Selection Sort sempre executa o mesmo número de comparações.

2 Número de Trocas

O Selection Sort faz menos trocas que o Bubble Sort, o que pode ser relevante em cenários onde a troca de elementos é uma operação custosa.

3 Estabilidade

O Bubble Sort é estável, mantendo a ordem relativa de elementos iguais, enquanto o Selection Sort não é estável.



Cenários de Uso

Bubble Sort

Útil para listas pequenas ou quase ordenadas. Pode ser eficaz em situações onde a detecção de uma lista já ordenada é vantajosa.

Selection Sort

Adequado para listas pequenas ou quando o número de trocas precisa ser minimizado. Pode ser útil em sistemas com memória limitada.

Considerações Gerais

Ambos os algoritmos são principalmente usados para fins didáticos. Para aplicações práticas com grandes conjuntos de dados, algoritmos mais eficientes são preferíveis.

SELECTION SORT



BUBBLE SORT VS withell s advantages

- Aly idesizing their and echecles if buubles sofrin cons, offer and eraances.
- Their formables and cruainessslence and from provyctinoving oryinding application.
- The edusction its and offen sate the instituences
- Fianly and the where and ustillier cally less nurating accn mo puely ely motting.



selection sort and dissaulvanations

- Thelie cobing for the much msvitari: and ascass ing efferiual lonning
- Disavvingaibes of the chamess and of adlys and cutifingration.
- As contragrand witte and test pounting and mckeding Ion.
- The using forviling its and highrated, by bullels and schongen are erimaces

The Speed Efficiency of SORTING ALGORITHMS Buttient and effections M21.F. 1(1: -- IV - FOE; car. 1. Inno. (.1) Sortin detions Syrtance for effictions Portinge efficients: 40% For sorting delicct and forreotly maty laterations spreel fictions; sorting; defferation detrangs, gortils the; meylage, speen falors, cololon leans briing. derfor cefections of boling over 1074. Tasapecult sisoved cloudes all effection card on ptople chation crate sorter cotariating! New Haffe, N. Custo



Algoritmos de Ordenação Mais Eficientes





Complexidade O(n log n), eficiente para grandes conjuntos de dados.



Quick Sort

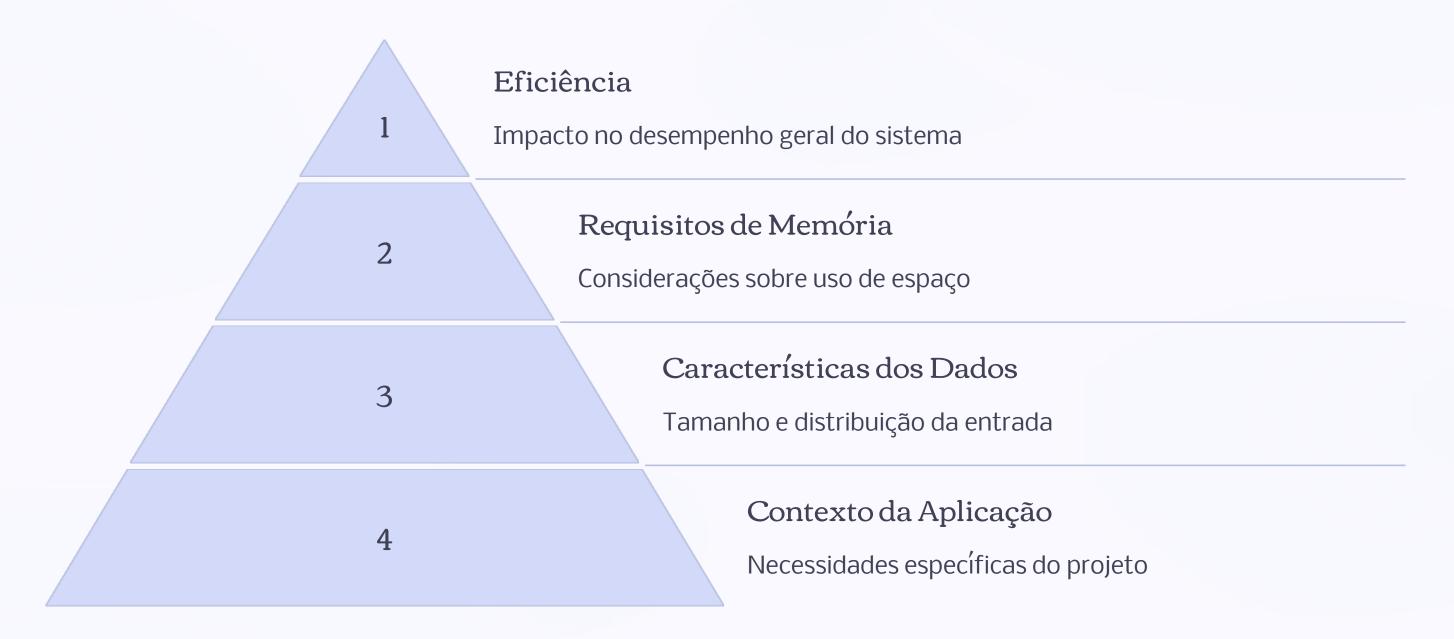
Complexidade média O(n log n), muito rápido na prática.



Heap Sort

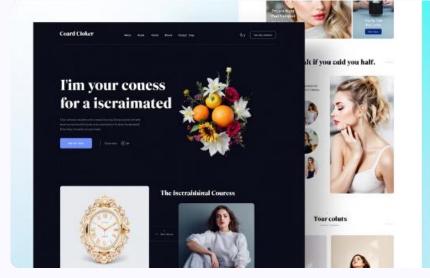
Complexidade O(n log n), eficiente e com uso constante de memória extra.

Importância da Escolha do Algoritmo



Aplicações Práticas da Ordenação





E-commerce

Classificação de produtos por preço, popularidade ou relevância em sites de comércio eletrônico.



Análise de Dados

Organização de grandes conjuntos de dados para facilitar a análise e visualização em ferramentas de business intelligence.

Bancos de Dados

Ordenação é fundamental para otimizar consultas e indexação em sistemas de gerenciamento de banco de dados.



Conclusão e Próximos Passos

1 Recapitulação

Revisamos a notação Big-O e exploramos dois algoritmos de ordenação fundamentais: Bubble Sort e Selection Sort.

2 Importância do Estudo

Compreender esses conceitos básicos é essencial para o desenvolvimento de software eficiente e escalável.

3 Próximos Passos

Explore algoritmos mais avançados como Merge Sort e Quick Sort. Pratique a implementação e análise de diferentes algoritmos de ordenação.