



SHELL SORT:



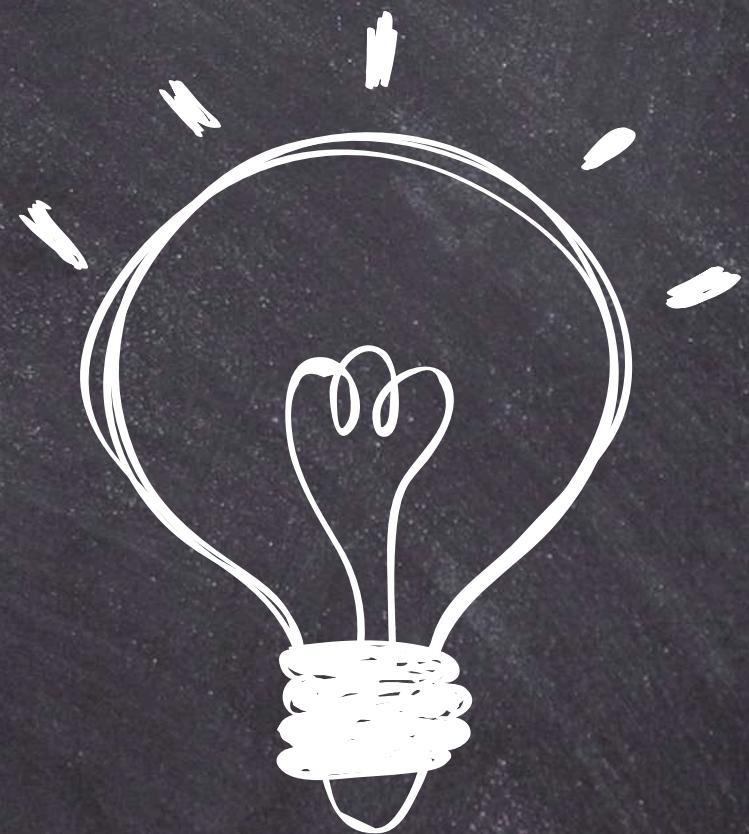
GRUPO 5:

- Ana Clara Guarizi de Souza •
- Daniel Henrique Neves Ferreira •
- Gabriel Pereira Alcântaro da Silva •
- Igor Leonardo Kades •
- Jorge Amorim Meneguz de Castro Neto •
- Matheus Endlich Silveira •
- Rafael Ferreira Bassul •
- Tallis Anholeti Azevedo •

INTRODUÇÃO:

CONTEXTUALIZAÇÃO AO SHELL SORT:

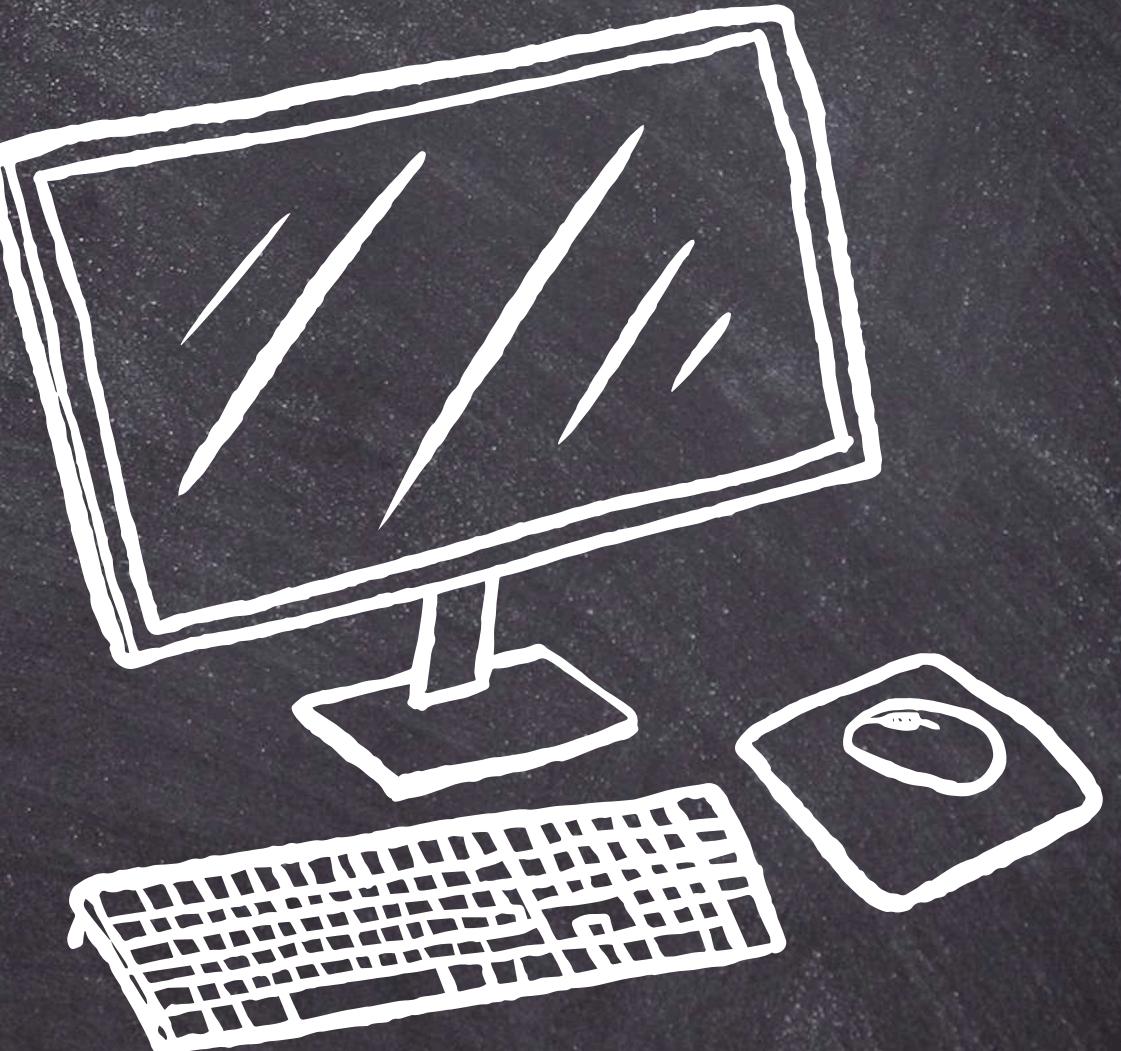
- Criado em Donald Shell por 1959;
- Extensão da ordenação por inserção
- Publicado pela Universidade de Cincinnati;
- “Dividir para conquistar!”;



VANTAGENS E DESVANTAGENS:

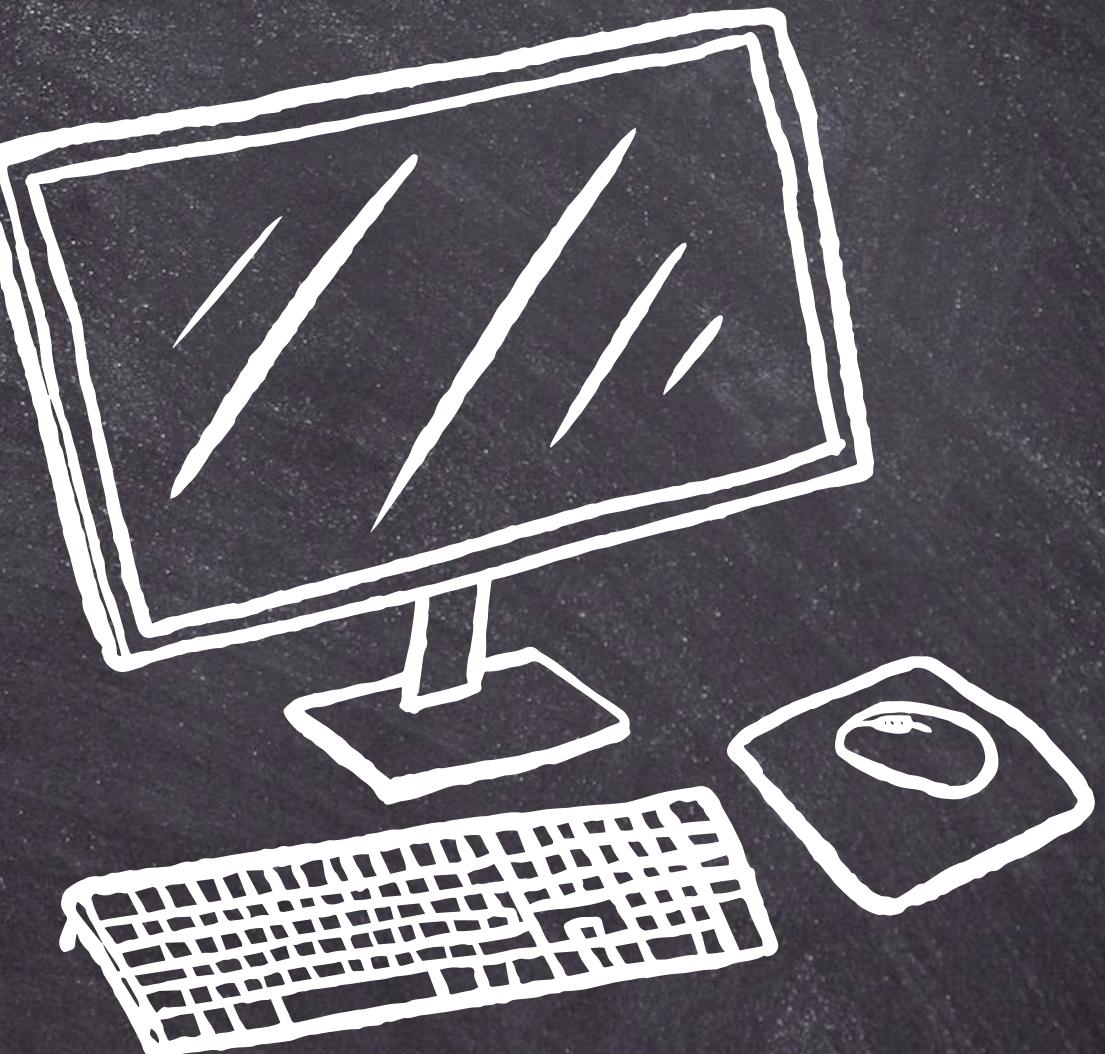
VANTAGENS:

- { SHELLSORT É UMA **ÓTIMA** OPÇÃO PARA ARQUIVOS DE TAMANHO MODERADO.
- { SUA IMPLEMENTAÇÃO É SIMPLES E REQUER UMA QUANTIDADE DE **CÓDIGO PEQUENA**.
- { SHELL SORT É O **MAIS** EFICIENTE ALGORITMO DE CLASSIFICAÇÃO DENTRE OS DE COMPLEXIDADE QUADRÁTICA.



DESVANTAGENS:

- { O TEMPO DE EXECUÇÃO DO ALGORITMO É SENSÍVEL À ORDEM INICIAL DO ARQUIVO.
- { O MÉTODO NÃO É ESTÁVEL, (OUTRAS FONTES DIZEM QUE É SIM!).
- { POUCAS LACUNAS(GAP) TORNAM AS PASSAGENS MAIS LENTAS, E MUITAS LACUNAS PRODUZEM UMA SOBRECARGA.



DEMONSTRAÇÃO EM C:

```
// Função de ordenação Shellsort que recebe o tamanho do array e um ponteiro para o array
void shellsort(int size, int *v){
    int i, j, h = 1, aux;

    // Define o valor inicial de 'h' usando a sequência  $3x + 1$  de Knuth
    for(; h < size; h = h * 3 + 1);| 

    // Loop principal do Shellsort, onde 'h' é reduzido a cada iteração
    do {
        h /= 3; // Reduz o intervalo 'h'

        // Realiza a ordenação por inserção para elementos espaçados por 'h'
        for(i = h; i < size; i++){
            aux = v[i]; // Guarda o elemento atual
            j = i;

            // Move os elementos do array para criar espaço para o 'aux' na posição correta
            while (j >= h && v[j - h] > aux) {
                v[j] = v[j - h];
                j -= h; // Ajusta o índice com o intervalo 'h'
            }
            v[j] = aux; // Coloca o elemento 'aux' na posição correta
        }
    } while (h != 1); // Continua até que 'h' seja 1 (última iteração)
}
```

COMPLEXIDADE:

ANÁLISE DE COMPLEXIBILIDADE DO SHELL SORT:

COMPLEXIDADE PIOR CASO

$O(n^2)$ (pior sequência de lacunas do pior caso conhecido)

$O(n \log^2 n)$ (sequência de lacuna de pior caso mais conhecida).

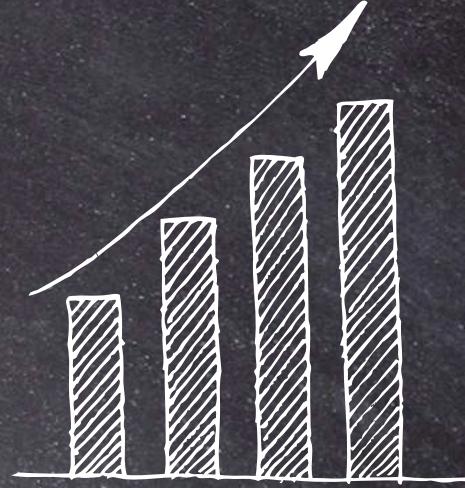
COMPLEXIDADE CASO MÉDIO

Depende da sequência do gap.

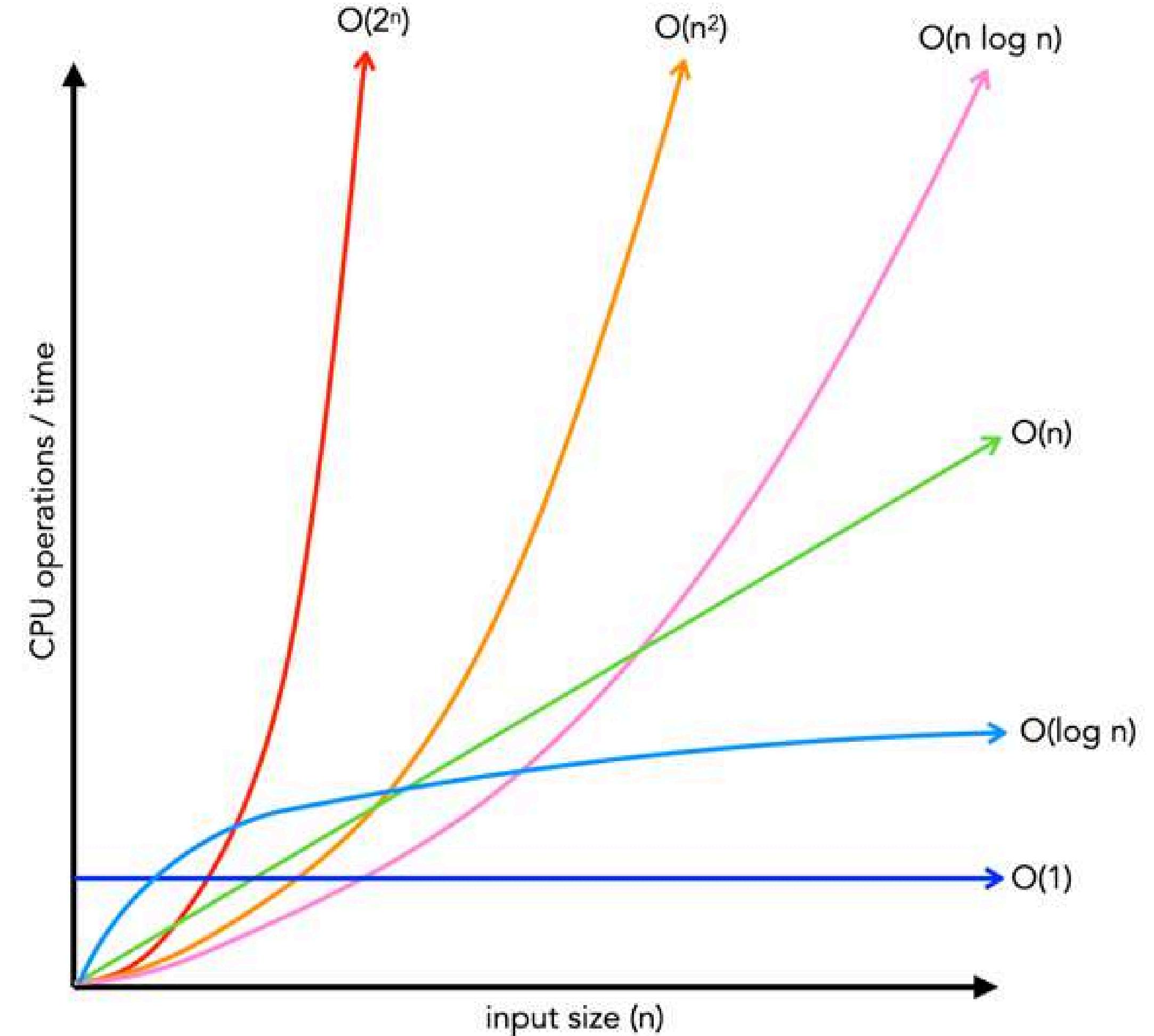
COMPLEXIDADE MELHOR CASO

$O(n \log n)$ (a maioria das sequências de lacunas)

$O(n \log^2 n)$ (sequência de gap de pior caso mais conhecida).

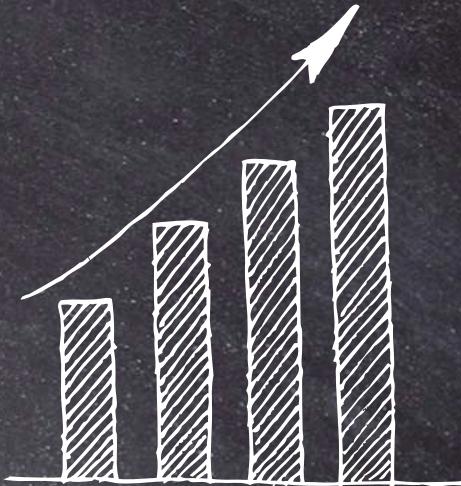


$O(N!)$	Factorial
$O(2^N)$	Exponential
$O(N^3)$	Cubic
$O(N^2)$	Quadratic
$O(N \log N)$	$N \times \log N$
$O(N)$	Linear
$O(\log N)$	Logarithmic
$O(1)$	Constant

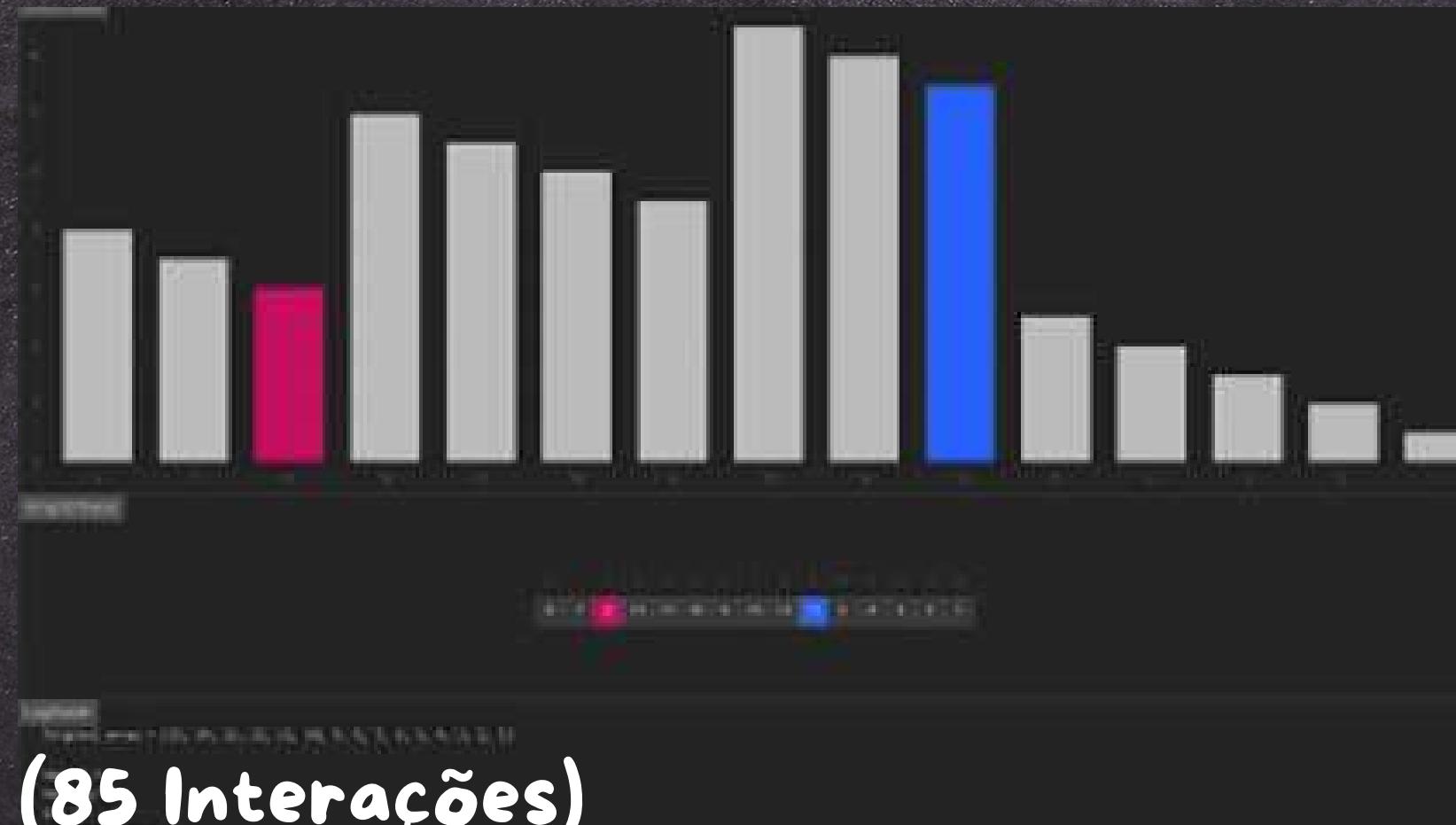


COMPLEXIDADE:

COMPLEXIDADE PIOR CASO



EXEMPLO:

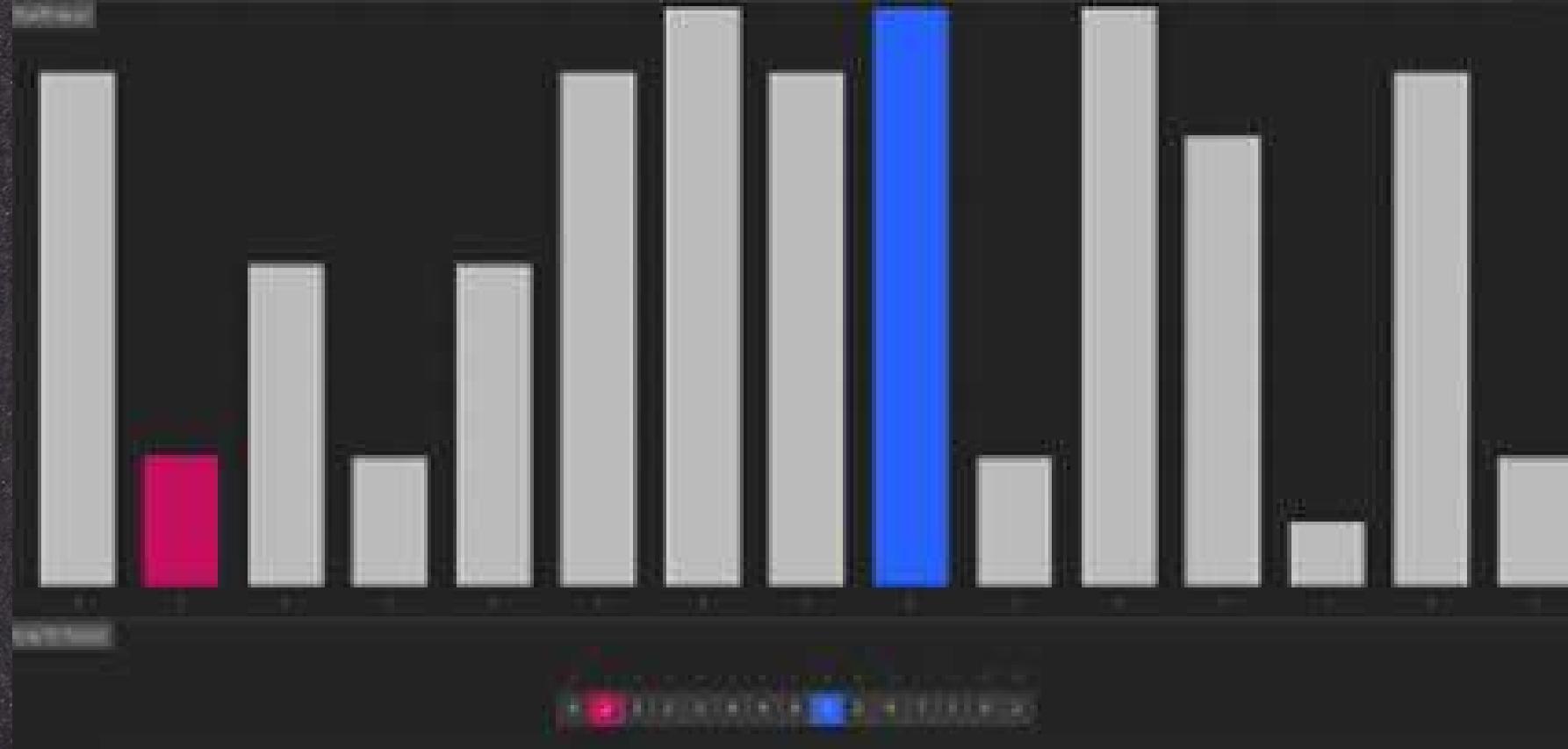


Para a sequência de lacunas $n/2, n/4, n/8, \dots, 1$, o desempenho do Shell Sort tende a ser próximo de $O(n^2)$ no pior caso. Um exemplo seria ordenar uma lista quase ordenada em ordem inversa (exemplo: $[5, 4, 3, 2, 1]$). Esse tipo de sequência de gaps leva mais tempo para rearranjar a lista para que fique ordenada.

COMPLEXIDADE:

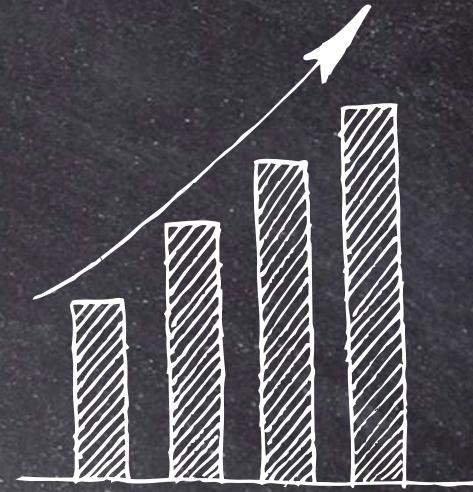
COMPLEXIDADE CASO MÉDIO

EXEMPLO:



(62 Interações)

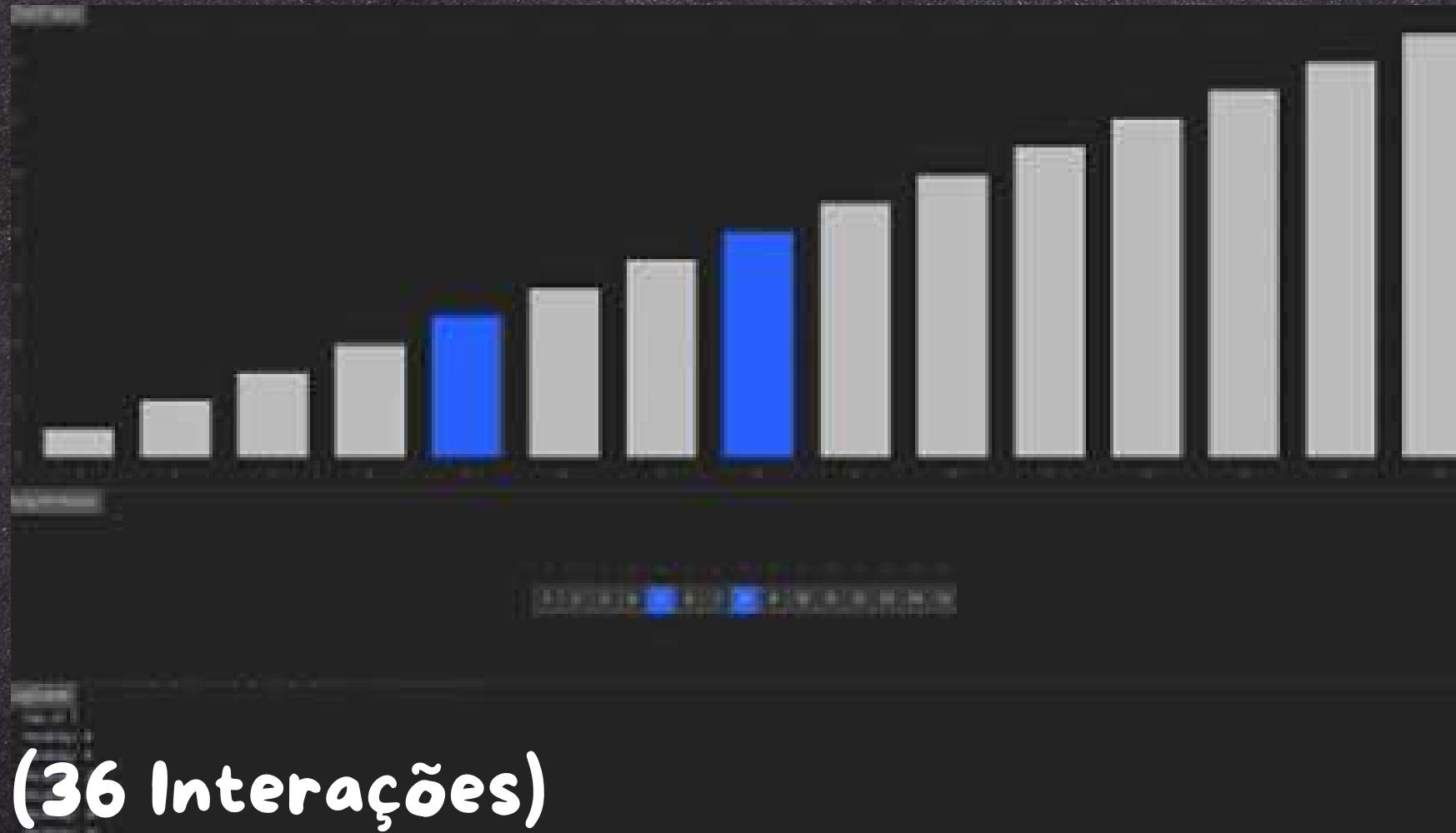
Para uma lista aleatória de elementos como [34, 8, 64, 51, 32, 21], o desempenho do Shell Sort depende da sequência de lacunas escolhida. Podem resultar em um desempenho melhor que o pior caso, mas a performance exata depende da sequência e do tamanho da entrada.



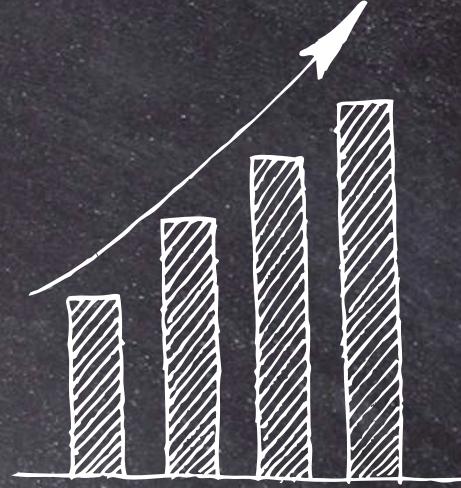
COMPLEXIDADE:

COMPLEXIDADE MELHOR CASO

EXEMPLO:



Um exemplo de melhor caso é quando os dados já estão parcialmente ordenados ou quando uma sequência de gaps eficiente é usada em uma lista já quase ordenada, como [1, 2, 3, 4, 5]. Nesse caso, o algoritmo Shell Sort pode se aproximar de $O(n \log n)$, pois os elementos não precisam ser movidos muito para encontrar sua posição final.

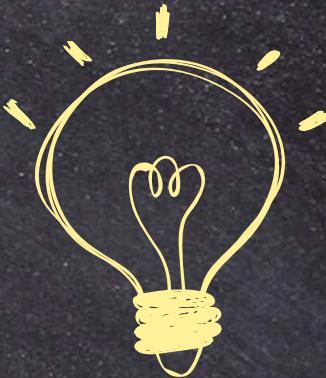


CONCLUSÃO:

CENÁRIOS DE USOS E VIABILIDADE:

O Shell Sort é viável para cenários com uma grande base de dados pela sua implementação dividir os dados em diferentes sessões antes da ordenação, ou seja, ao invés do algoritmo percorrer o vetor como um só e ordena-lo, ele divide o problema e ordena-o de forma isolada.

REFERÊNCIAS:



- Shell Sort (Wikipedia) : https://pt.wikipedia.org/wiki/Shell_sort
- Shell Sort (Geek For Geeks): <https://www.geeksforgeeks.org/shell-sort/>
- Medium (GIF): <https://medium.com/@ricardo.macedo/desempenho-e-complexidade-um-estudo-comparativo-dos-algoritmos-de-ordena%C3%A7%C3%A3o-b488bcaba3ff>
- <https://algorithm-visualizer.org/brute-force/shellsort>