Algoritmo de ordenação Shell Sort

Carlos, Jennifer, Jéssica, Pedro e Ramon.

SHELL SORT

I. Definição de “Shell Sort”;
II. Funcionamento do algoritmo;
   A. Passo a passo;
   B. Relembrando Insertion Sort;
   C. Ilustração;
   D. Algumas sequências.
III. Exemplos de utilização;
IV. Exemplos de implementação;
   A. Shell;
   B. Knuth;
   C. Ciura.
V. Vantagens e desvantagens;
VI. Conclusão;
VII. Referências.

I. Definição de “Shell Sort”;

- Algoritmo de ordenação in-place criado por Donald Shell em 1959;
- Extensão do InsertionSort e tem uma performance melhor;
- Permite a troca de elementos distantes dentro do array;
- Tempo de execução fortemente influenciado pela sequência que calcula a distância;
- Sequência perfeita ainda não foi encontrada;
- Complexidade de tempo depende da sequência utilizada.

II. Funcionamento do algoritmo;
A. Passo a passo;

- Escolher uma sequência para calcular a distância entre os elementos. Exemplos:
  - Shell: N / 2^k , k >= 1 , N == tam do array;
  - Knuth: (3^k - 1) / 2 , k >= 1;
  - Sedgewick: 4^(k) + 3*2^(k-1) + 1 , k >= 1.

- Começar o algoritmo com uma distância grande;

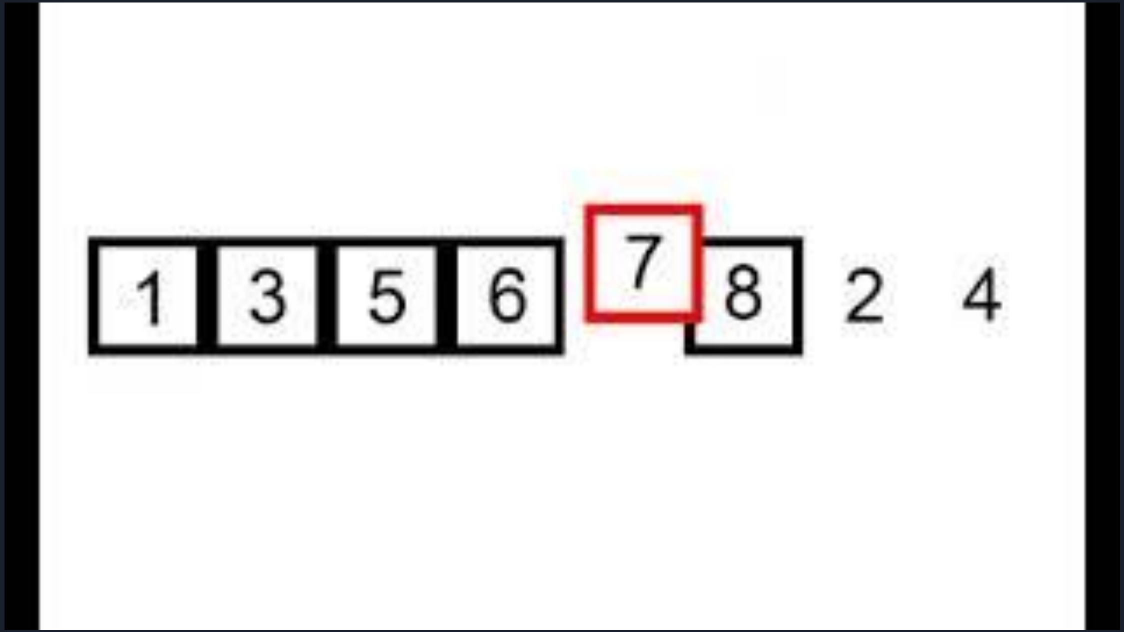
- De acordo com a distância, comparar e, se necessário, trocar de posição os elementos;

- Reduzir a distância;

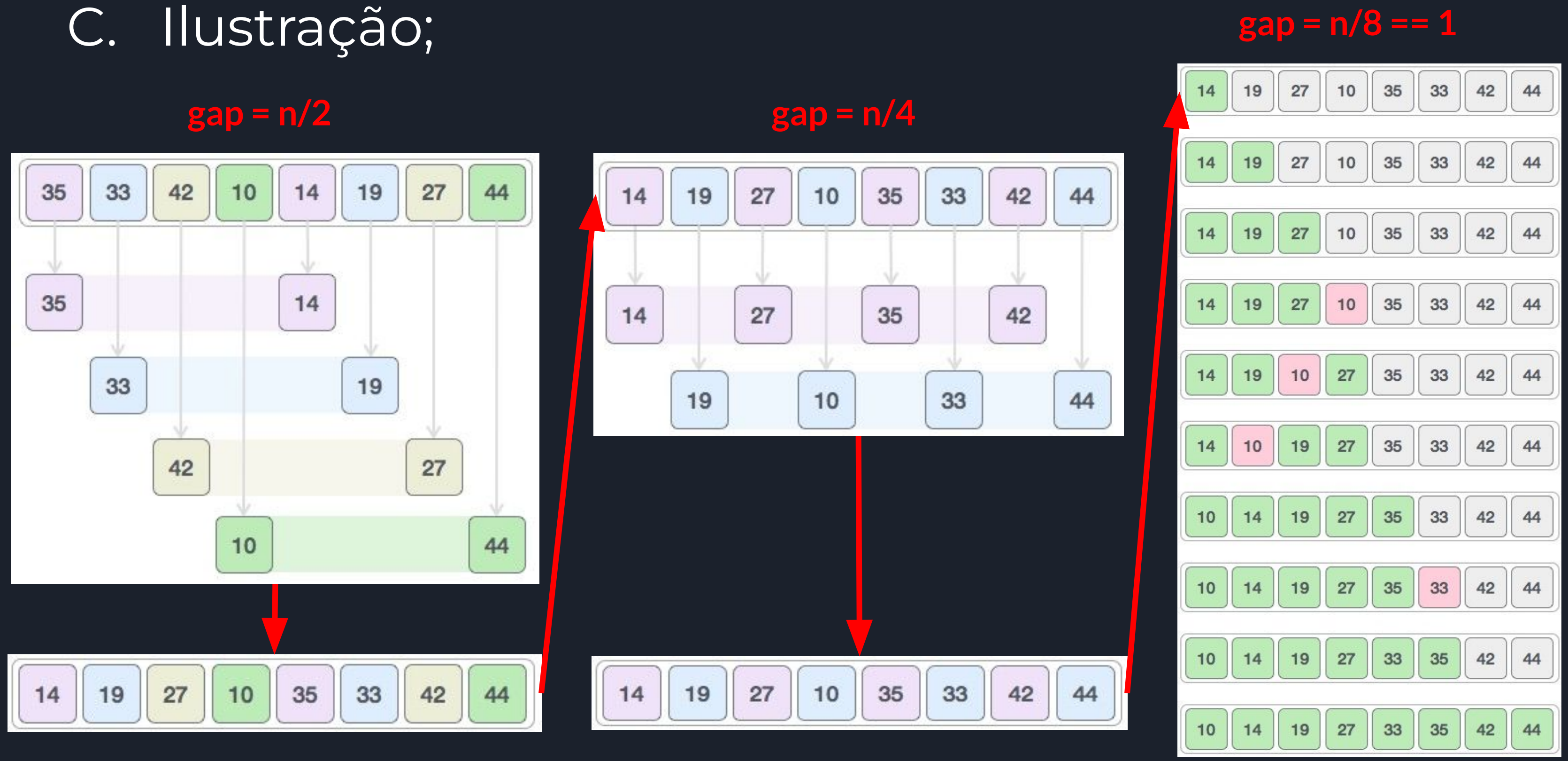
- Repetir o processo até que a distância seja igual a 1;

- Com a distância igual a 1, utilizar o Insertion Sort.

II. Funcionamento do algoritmo;
B. Relembrando Insertion Sort;

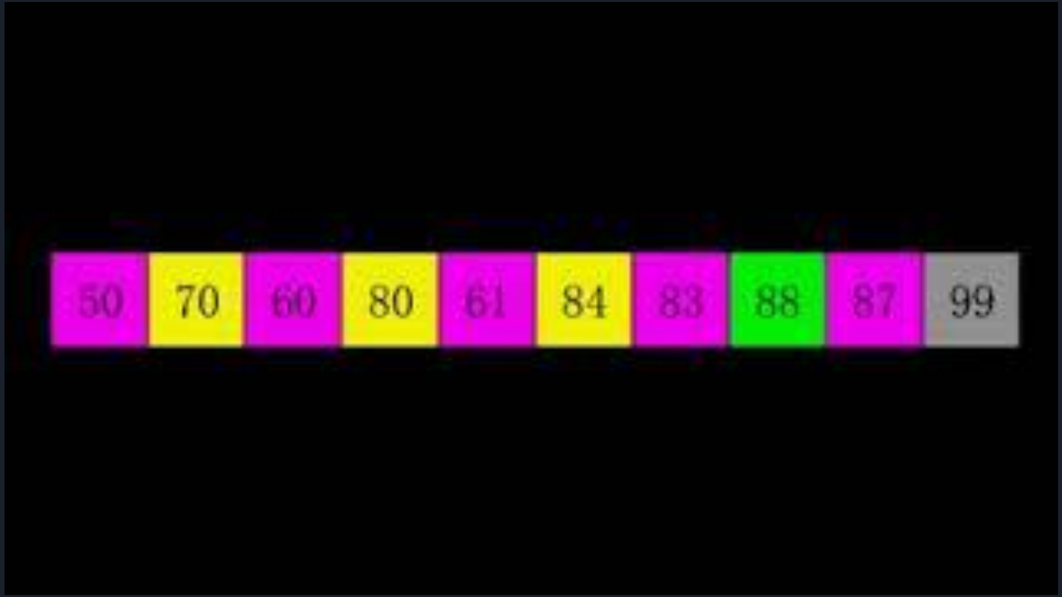


II. Funcionamento do algoritmo;
C. Ilustração;



II. Funcionamento do algoritmo;

C. Ilustração;



II. Funcionamento do algoritmo;
D. Algumas sequências.

- Donald Shell (1959):
  - Fórmula: [N/2k]
  - Sequência: [N/2], [N/4], [N/8], … 1
- Hibbard (1963):
  - Fórmula: 2k-1
  - Sequência: 0, 1, 3, 7, 15, 31
- Papernov & Stasevich (1965):
  - Fórmula: 2k+1, prefixado com 1
  - Sequência: 1, 3, 5, 9, 17, 33, 65
- Knuth (1973):
  - Fórmula: (3^n – 1)/2
  - Sequência: 0, 1, 4, 13, 40, 121, 364

II. Funcionamento do algoritmo;
D. Algumas sequências.

- Sedgewick (1982 e 1986):
  - Fórmula 1: 4^(n+1) + 3*2^n + 1, prefixado com 1
  - Sequência: 1, 8, 23, 77, 281, 1073, 4193
  - Fórmula 2:
    - Para arrays de tamanho par: 9*2^n – 9*2^(n/2) + 1
    - Para arrays de tamanho ímpar: 8*2^n – 6*2^((n+1)/2) + 1
  - Sequência 2: 1, 5, 19, 41, 109, 209, 505, 929, 2161
- Tokuda (1992):
  - Fórmula: ceiling((9 * (9/4)^n – 4) / 5).
  - Sequência: 1, 4, 9, 20, 46, 103, 233
- Ciura (2001):
  - “Melhores incrementos para o caso médio de ShellSort.”
  - Sequência: 1, 4, 10, 23, 57, 132, 301, 701, 1750

II. Funcionamento do algoritmo;
D. Algumas sequências.

OEIS	General term ($k \geq 1$)	Concrete gaps	Worst-case time complexity	Author and year of publication
	$\left\lfloor \frac{N}{2^k} \right\rfloor$	$1, 2, \ldots, \left\lfloor \frac{N}{4} \right\rfloor, \left\lfloor \frac{N}{2} \right\rfloor$	$\Theta\left(N^2\right)$ [e.g. when $N = 2^p$]	Shell, 1959[4]
	$2\left\lfloor \frac{N}{2^{k+1}} \right\rfloor + 1$	$1, 3, \ldots, 2\left\lfloor \frac{N}{8} \right\rfloor + 1, 2\left\lfloor \frac{N}{4} \right\rfloor + 1$	$\Theta\left(N^{\frac{3}{2}}\right)$	Frank & Lazarus, 1960[8]
A000225	$2^k - 1$	$1, 3, 7, 15, 31, 63, \ldots$	$\Theta\left(N^{\frac{3}{2}}\right)$	Hibbard, 1963[9]
A083318	$2^k + 1$, prefixed with 1	$1, 3, 5, 9, 17, 33, 65, \ldots$	$\Theta\left(N^{\frac{3}{2}}\right)$	Papernov & Stasevich, 1965[10]
A003586	Successive numbers of the form $2^p 3^q$ (3-smooth numbers)	$1, 2, 3, 4, 6, 8, 9, 12, \ldots$	$\Theta\left(N \log^2 N\right)$	Pratt, 1971[1]
A003462	$\frac{3^k - 1}{2}$, not greater than $\left\lceil \frac{N}{3} \right\rceil$	$1, 4, 13, 40, 121, \ldots$	$\Theta\left(N^{\frac{3}{2}}\right)$	Knuth, 1973,[3] based on Pratt, 1971[1]

II. Funcionamento do algoritmo;
D. Algumas sequências.

A036569	$\prod_I a_q$, where $a_0 = 3$ $a_q = \min\left\{n \in \mathbb{N}: n \geq \left(\frac{5}{2}\right)^{q+1}, \forall p: 0 \leq p < q \Rightarrow \gcd(a_p, n) = 1\right\}$ $I = \left\{0 \leq q < r \mid q \neq \frac{1}{2}\left(r^2 + r\right) - k\right\}$ $r = \left\lfloor \sqrt{2k + \sqrt{2k}} \right\rfloor$	$1, 3, 7, 21, 48, 112, \ldots$	$O\left(N^{1+\sqrt{\frac{8\ln(5/2)}{\ln(N)}}}\right)$	Incerpi & Sedgewick, 1985,[11] Knuth[3]
A036562	$4^k + 3 \cdot 2^{k-1} + 1$, prefixed with 1	$1, 8, 23, 77, 281, \ldots$	$O\left(N^{\frac{4}{3}}\right)$	Sedgewick, 1982[6]
A033622	$\begin{cases} 9\left(2^k - 2^{\frac{k}{2}}\right) + 1 & k \text{ even}, \\ 8 \cdot 2^k - 6 \cdot 2^{(k+1)/2} + 1 & k \text{ odd} \end{cases}$	$1, 5, 19, 41, 109, \ldots$	$O\left(N^{\frac{4}{3}}\right)$	Sedgewick, 1986[12]
	$h_k = \max\left\{\left\lfloor \frac{5h_{k-1} - 1}{11} \right\rfloor, 1\right\}, h_0 = N$	$1, \ldots, \left\lfloor \frac{5}{11}\left\lfloor \frac{5N-1}{11} \right\rfloor - \frac{1}{11} \right\rfloor, \left\lfloor \frac{5N-1}{11} \right\rfloor$	Unknown	Gonnet & Baeza-Yates, 1991[13]
A108870	$\left\lceil \frac{1}{5}\left(9 \cdot \left(\frac{9}{4}\right)^{k-1} - 4\right) \right\rceil$	$1, 4, 9, 20, 46, 103, \ldots$	Unknown	Tokuda, 1992[14]
A102549	Unknown (experimentally derived)	$1, 4, 10, 23, 57, 132, 301, 701$	Unknown	Ciura, 2001[15]
	$\left\lceil \frac{\gamma^k - 1}{\gamma - 1} \right\rceil, \gamma = 2.243609061420001\ldots$	$1, 4, 9, 20, 45, 102, \ldots$	Unknown	Lee, 2021[16]

III. Exemplos de utilização;

- Ordenar arrays com uma quantidade moderada ou grande de dados como, por exemplo, microcontroladores que não possuem muita memória e precisam de um algoritmo in-place eficiente.

- Linux Kernel: É utilizado na biblioteca uClibc, que é utilizada em sistemas embarcados.

- Algoritmos de compressão: Pode ser usado como sub-rotina nesses algoritmos para ordenar os dados antes de comprimi-los, reduzindo o tamanho do arquivo. É utilizado na biblioteca de compressão bzip2 utilizando os incrementos de Knuth.

III. Exemplos de utilização;

- Criptografia: Pode ser utilizado em alguns algoritmos de criptografia para embaralhar a ordem dos dados antes de criptografar. O que torna a criptografia mais segura e mais difícil de ser quebrada.

- Indexação de banco de dados: Utilizado para ordenar dados em um índice de banco de dados. No SQLite, a estrutura de dados B-tree é utilizada para indexação, e o Shell Sort é usado como algoritmo de ordenação para construir e manter a B-tree.

- Visualização de dados: Utilizado em algumas ferramentas de visualização de dados para ordenar os dados antes de exibi-los. Tornando os dados mais legíveis e mais fáceis de interpretar.

IV. Exemplos de implementação;

A. Shell;

```cpp
// Shell Sort Algorithm (Shell's Sequence) C++ Implementation
void shell_sort (int arr[], int size)
{
    int inner, outer, valueToInsert, gap = size/2; // Shell's sequence: n/2
    while(gap > 0)
    {
        for(outer = gap; outer < size; outer++)
        {
            valueToInsert = arr[outer]; // select value to be inserted
            inner = outer;
            while(inner > gap-1 && arr[inner-gap] >= valueToInsert)
            {
                arr[inner] = arr[inner-gap]; // shift element towards right
                inner = inner-gap;
            }
            arr[inner] = valueToInsert; // insert the number at hole position
        }
        gap /= 2; // calculate gap using Shell's sequence: n/4, n/8, ...
    }
}
```

IV. Exemplos de implementação;
B. Knuth;

```
// Shell Sort Algorithm (Knuth's Sequence) C++ Implementation
void shell_sort (int arr[], int size)
{
    int inner, outer, valueToInsert, gap = 0;
    while(gap < size/3) gap = gap*3 + 1; // calculate gap using Knuth's Sequence
    while(gap > 0)
    {
        for(outer = gap; outer < size; outer++)
        {
            valueToInsert = arr[outer]; // select value to be inserted
            inner = outer;
            while(inner > gap-1 && arr[inner-gap] >= valueToInsert)
            {
                arr[inner] = arr[inner-gap]; // shift element towards right
                inner = inner-gap;
            }
            arr[inner] = valueToInsert; // insert the number at hole position
        }
        gap = (gap-1)/3; // calculate gap using Knuth's sequence
    }
}
```

IV. Exemplos de implementação;
C. Ciura.

```
// Shell Sort Algorithm (Ciura's Sequence) C++ Implementation
void shell_sort (int arr[], int size)
{
    int ciura[] = {701, 301, 132, 57, 23, 10, 4, 1, 0};
    int inner, outer, valueToInsert, gap = ciura[0], counter = 0;
    while(gap > size)
    {
        counter++;
        gap = ciura[counter];
    }
    while(gap > 0)
    {
        for(outer = gap; outer < size; outer++)
        {
            valueToInsert = arr[outer]; // select value to be inserted
            inner = outer;
            while(inner > gap-1 && arr[inner-gap] >= valueToInsert)
            {
                arr[inner] = arr[inner-gap]; // shift element towards right
                inner = inner-gap;
            }
            arr[inner] = valueToInsert; // insert the number at hole position
        }
        counter++;
        gap = ciura[counter]; // calculate gap
    }
}
```

V. Vantagens e desvantagens;

- Vantagens:
    - Implementação simples;
    - Desempenho médio mais rápido do que outros algoritmos, como o Bubble Sort e o Selection Sort.
    - Ordenação in-place: Não requer espaço adicional de memória para a ordenação.
- Desvantagens:
    - Não é estável: Não preserva a ordem dos elementos iguais nos dados de entrada.
    - Tempo de execução sensível à ordem inicial dos dados.
    - Já que a sequência perfeita não foi encontrada, pode ser difícil encontrar a melhor sequência para o problema. É necessário pesquisar e comparar os resultados com testes.

VI. Conclusão;

- O Shell Sort acaba sendo uma opção geralmente melhor do que outros algoritmos de ordenação com complexidade O(n^2) porque permite ordenar elementos que estão distantes dentro do array e isso otimiza a utilização do Insertion Sort, que é o último passo no algoritmo.

- Apesar do algoritmo ser bom para tamanhos moderados, o foco principal mesmo é a sua simplicidade de implementação e a eficiência no uso da memória, já que o ShellSort utiliza apenas o espaço já alocado para o array.

VII. Referências.

- https://codecrucks.com/shell-sort/
- https://www.shiksha.com/online-courses/articles/shell-sort-advantages-and-disadvantages/
- https://web.archive.org/web/20181026010135/http://www.stoimen.com:80/blog/2012/02/27/computer-algorithms-shell-sort/
- https://www.scholarhat.com/tutorial/datastructures/shell-sort-in-data-structures
- https://www.mycplus.com/featured-articles/shell-sort-algorithm/
- https://www.treinaweb.com.br/blog/conheca-os-principais-algoritmos-de-ordenacao
- https://web.archive.org/web/20180923235211/http://sun.aei.polsl.pl/~mciura/publikacje/shellsort.pdf
- https://en.wikipedia.org/wiki/Shellsort
- https://www.programiz.com/dsa/shell-sort