Shell Sort

Algoritmo de Ordenação

- Maria Eduarda
- Thiago Rezende
- Giovane cardoso

Sobre o Algoritmo

Criado por Donald Shell em 1959, publicado pela Universidade de Cincinnati, Shell sort é o mais eficiente algoritmo de classificação dentre os de complexidade quadrática. Basicamente o algoritmo passa várias vezes pela lista dividindo o grupo maior em menores. Nos grupos menores é aplicado o método da ordenação por inserção (SHELL..., 2019a).

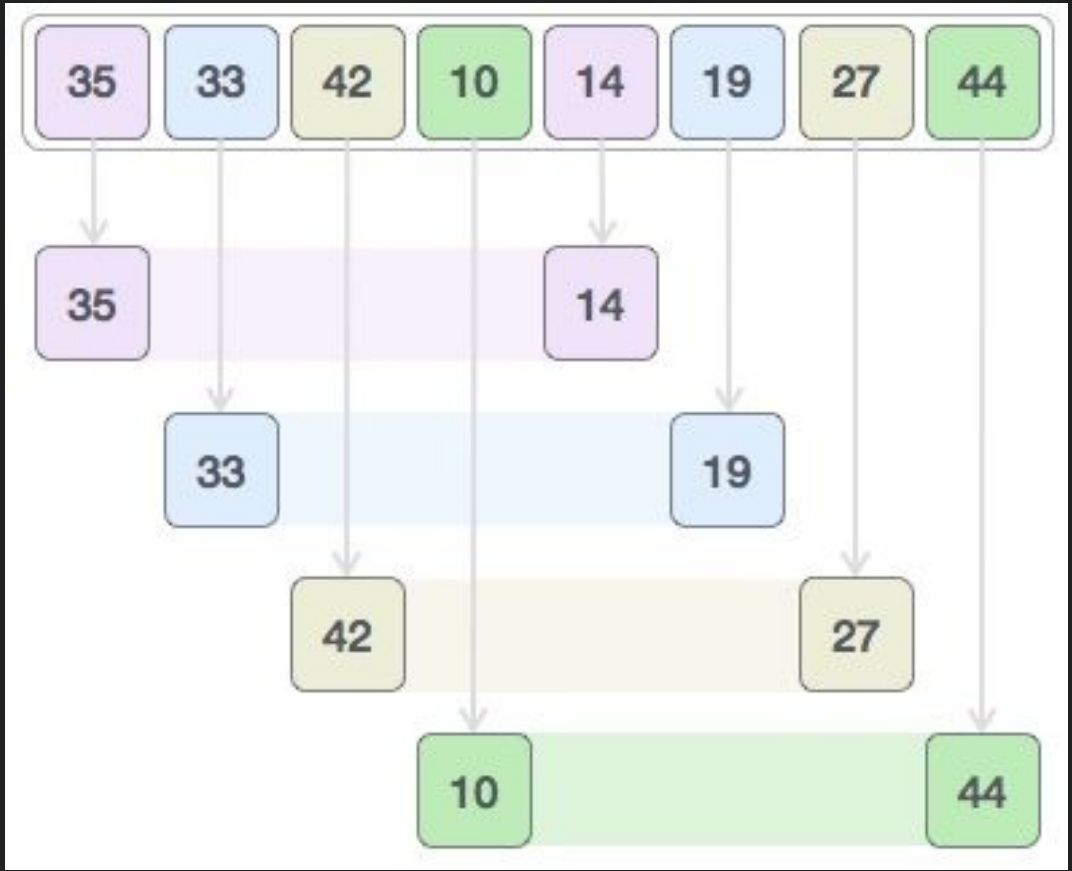
Funcionamento

Divide a lista em grupos menores, determinados por uma distância “gap” entre os elementos.

“gap” inicialmente tem o valor de “n”/2, sendo “n” o tamanho da lista.

Após a verificação e ordenação desse grupos menores, o valor de “gap” e dividido por 2 novamente.

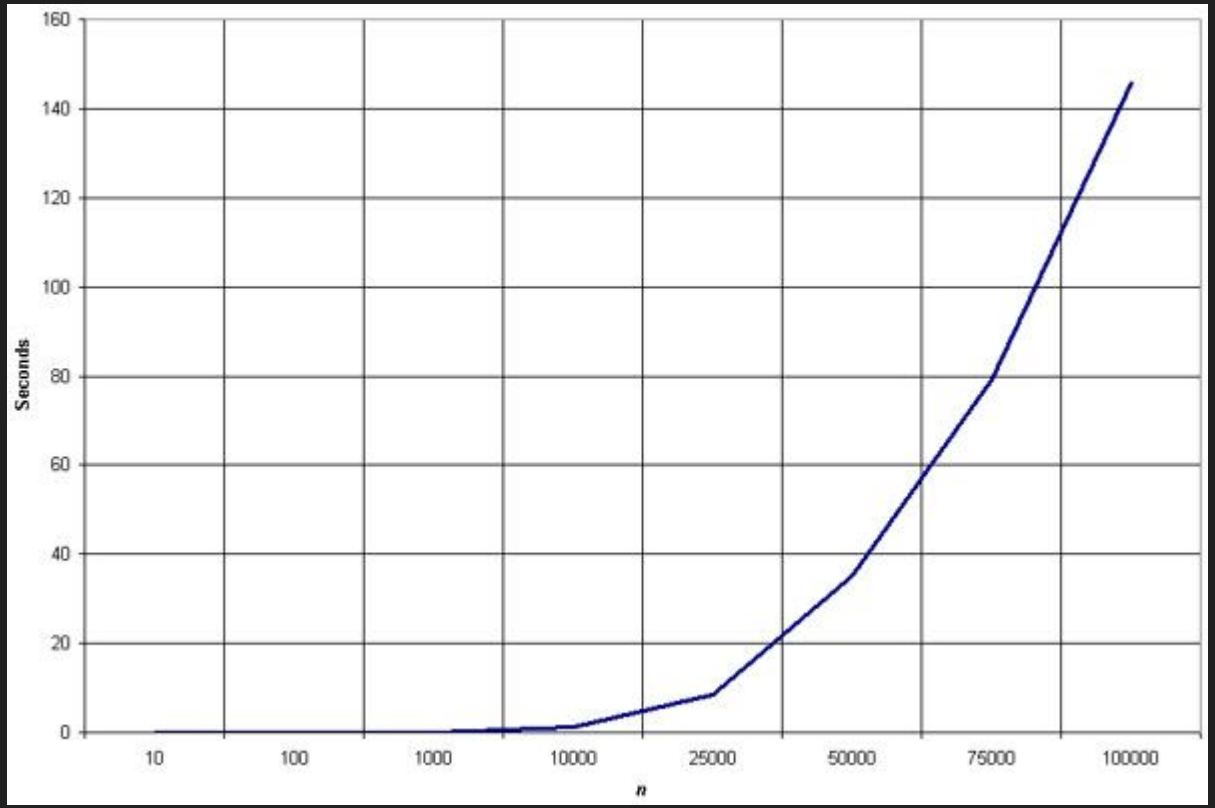
O processo se repete até que “gap” seja menor que 1.



fonte: https://www.tutorialspoint.com/data_structures_algorithms/shell_sort_algorithm.htm

Eficiência

Eficiente em coleções de tamanho pequeno e médio (SHELL..., 2019b).



fonte: http://www.hts.stevenwood.com/ics4u/cos/units/3/activity2/linux-wku-edu/lamonml/algor/sort/shell.html

Implementação

Implementação do algoritmo de Shell Sort feita em java (infelizmente).

```java
public static void shell(Integer[] arr) {
        int comparations = 0;
        int movements = 0;
        int gap = arr.length / 2;

        System.out.println(" - sorting [" + arr.length + "] -");

        for (; gap > 0; gap /= 2) {
            for (int i = gap; i < arr.length; i += 1) {

                int temp = arr[i];
                int j;

                for (j = i; j >= gap && arr[j - gap] > temp; j -= gap) {
                    arr[j] = arr[j - gap];

                    if (j != i)
                        comparations++;

                    movements++;
                }
                arr[j] = temp;
                comparations++;
            }
        }

        System.out.println("Copmarations: " + comparations);
        System.out.println("Movements: " + movements + "\n");
    }
```

Saída dos Casos de Teste

```
<Ascending Order>

 - sorting [10] -
Copmarations: 22
Movements: 0

 - sorting [100] -
Copmarations: 503
Movements: 0

 - sorting [1000] -
Copmarations: 8006
Movements: 0
```

```
<Rescending Order>

 - sorting [10] -
Copmarations: 24
Movements: 13

 - sorting [100] -
Copmarations: 507
Movements: 260

 - sorting [1000] -
Copmarations: 8334
Movements: 4700
```

```
<Random Order>

 - sorting [10] -
Copmarations: 22
Movements: 7

 - sorting [100] -
Copmarations: 647
Movements: 355

 - sorting [1000] -
Copmarations: 11783
Movements: 7738
```

Resultados

Vetor em ordem crescente

	10	100	1000
Comparações	22	503	8006
Movimentações	0	0	0

Resultados

Vetor em ordem decrescente

	10	100	1000
Comparações	24	507	8334
Movimentações	13	260	4700

Resultados

Vetor em ordem randômica

	10	100	1000
Comparações	22	647	11783
Movimentações	7	355	7738

GitHub

https://github.com/thiago-rezende/shell-sort-java

Referências

- CARBON. [S. l.]. Disponível em: https://carbon.now.sh/. Acesso em: 28 nov. 2019.
- SHELL Sort. [S. l.], 28 nov. 2019. Disponível em: https://pt.wikipedia.org/wiki/Shell_sort. Acesso em: 28 nov. 2019.
- SHELL Sort. [S. l.]. Disponível em: http://www.hts.stevenwood.com/ics4u/cos/units/3/activity2/linux-wku-edu/lamonml/algor/sort/shell.html. Acesso em: 28 nov. 2019.
- DATA Structure and Algorithms - Shell Sort. [S. l.], 28 nov. 2019. Disponível em: https://www.tutorialspoint.com/data_structures_algorithms/shell_sort_algorithm.htm. Acesso em: 28 nov. 2019.
- INSERTION Sort. [S. l.], 28 nov. 2019. Disponível em: https://pt.wikipedia.org/wiki/Insertion_sort. Acesso em: 28 nov. 2019.