# Shell Sort

Criado por Donald Shell em 1959, publicado pela universidade de Cincinnati. É considerado um refinamento do método Insertion Sort. O algoritmo difere do Insertion Sort pelo facto de ao invés de considerar o vetor a ser ordenado como um único segmento, ele considera vários segmentos.

# Funcionalidade

O Shell Sort se baseia em uma variável chamada de incremento de sequência, ou incremento de shell, que é dado por h e ao decorrer da execução do algoritmo, é decrementada até 1.
Utilizando o incremento de shell, o algoritmo compara elementos distantes em um vetor, em vez de comparar os adjacentes.
No algoritmo, a ordenação é realizada em vários passos, usando uma sequência de valores do incremento de shell <h1, h2, h3…hN> onde começando por hN selecionamos apenas os valores que estão hN elementos distantes um do outro, então ordenamos esses elementos com algum algoritmo de ordenação simples. Deste modo, apenas os elementos selecionados serão ordenados, os outros são todos ignorados.

Como escolher valores para h?
Existem vários modos de escolher valores para *h* baseados em estudos de eficiência, podemos até escolher os valores que quisermos, apesar de ser altamente não recomendável por questões de eficiência. O método de seleção de valores para *h* proposto por Donald Shell foi *h = n/2* onde *n* é o número de elementos do vetor, mas já foi provado que sua eficiência é baixa. O outro método é a de *Donald Knuth*, explicado como:
h= 3*h+1
E a sequência gerada: 1, 4, 13, 40, 121, 364, 1093, 3280…

Partindo de h = 1[*] precisamos do maior valor possível que seja menor que o número de elementos no vetor, então basta calcular:
h1 = 3*1+1 = 4
h2 = 3*4+1 = 13
h 3 = 3*13+1 = 40
etc.
Para decrementar os valores de h até h=1 podemos usar a fórmula inversa:
h = (h-1) / 3
…3280, 1093, 364, 121, 40, 13, 4, 1.

# Complexidade

É o algoritmo mais eficiente de complexidade quadrática.

# Estabilidade

O método não é estável.

# Vantagens

Shellsort é uma ótima opção para arquivos de tamanho moderado;

Sua implementação é simples e requer uma quantidade de código pequena.

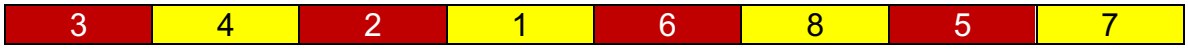
## Desvantagem

O tempo de execução do algoritmo é sensível à ordem inicial do arquivo.

## Representação gráfica (sequencia original do Shell)

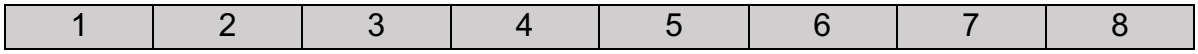
Começamos a organizar os elementos dividindo pela metade, como temos 8 elementos, o resultado é 4.


3 é maior que 6? Se a resposta for verdadeira trocamos o elemento de lugar, se não deixamos na mesma posição.




Dividimos os 8 elementos pela metade e o resultado foi 4, agora pegamos esse valor 4 e dividimos pela metade, o resultado é 2.


Pegamos o resultado anterior que foi 2 e dividimos pela metade, o resultado é 1.




Elementos organizados!

## Implementação em C

```
void shellsort(struct item *v, int n) {

int i, j, h;

struct item aux;

for(h = 1; h < n; h = 3*h+1);

    while(h > 0) {
      h = (h-1)/3
      for(i = h; i < n; i++) {

aux = v[i];

j = i;

while(v[j – h].chave > aux.chave) {
v[j] = v[j – h];
```

```
j -= h;
if(j < h) break;

}

v[j] = aux;
}

}

}
```

## Referencias Bibliográficas

https://www.treinaweb.com.br/blog/conheca-os-principais-algoritmos-de-ordenacao
https://cadernogeek.wordpress.com/tag/shell-sort/
https://youtu.be/9TGB6A4svMk
https://pt.slideshare.net/jackocap/aa-algoritmo-shell-sort
http://tiagodemelo.info/wp-content/uploads/2019/08/algoritmos-ordenacao-1.pdf