DCC
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

UFMG
UNIVERSIDADE FEDERAL DE MINAS GERAIS

Estrutura de Dados

Ordenação: Shellsort

Professores: Luiz Chaimowicz e Raquel Prates

Introdução

- Proposto por Donald Shell em 1959.
- Extensão do algoritmo de ordenação inserção.
  - Alto custo de movimentação de elementos
  - Como melhorar isso?

Método Inserção

```
void Insercao(Item *v, int n) {
  int i,j;
  Item aux;
  for (i = 1; i < n; i++) {
    aux = v[i];
    j = i - 1;
    while (( j >= 0 ) && (aux.Chave < v[j].Chave)) {
      v[j + 1] = v[j];
      j--;
    }
    v[j + 1] = aux;
  }
}
```

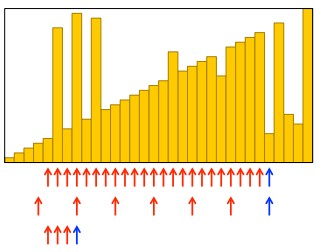
6	5	3	1	8	7	2	4

Motivação

- Problema do método de inserção
  - Troca itens adjacentes para determinar o ponto de inserção.
  - Qual é o pior caso?
    - São efetuadas n - 1 comparações e movimentações quando o menor item está na posição mais à direita no vetor.
- Shell sort contorna este problema permitindo trocas de registros distantes um do outro.

Beginning with large values of h, this rearrangement allows elements to move long distances in the original list, reducing large amounts of disorder quickly, and leaving less work for smaller h-sort steps to do.

Motivação



```
while (( j >= 0 ) && (aux.Chave < v[j].Chave)) {
    v[j + 1] = v[j];
    j--;
}
```

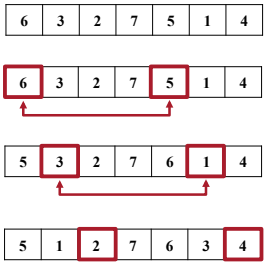
11 comp.
24 comp.

Algoritmo

- Os itens separados de *h* posições são rearranjados.
- Todo *h*-ésimo item leva a uma sequência ordenada.
- Tal sequência é dita estar *h*-ordenada.
- Quando h=1, o algoritmo é equivalente ao algoritmo de inserção

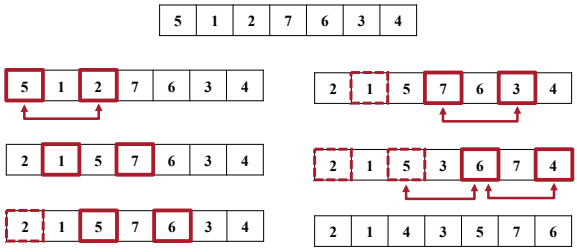
Exemplo

- h = 4



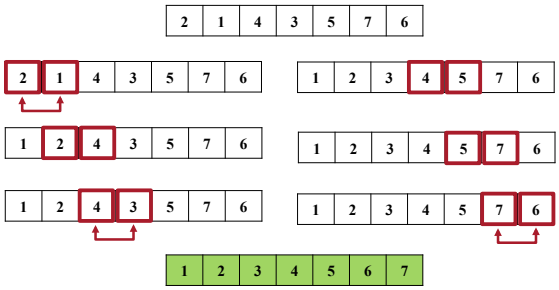
Exemplo

- h = 2



Exemplo

- h = 1



Escolha do h

- Sequência para h:

$$h(s) = 1, \text{para } s = 1$$

$$h(s) = 3h(s\text{-}1) + 1 \text{ para } s > 1$$

- A sequência corresponde a 1, 4, 13, 40, 121, 364, 1093, 3280, ...
  - Knuth (1973, p. 95) mostrou experimentalmente que esta sequência é difícil de ser batida por mais de 20% em eficiência.

ShellSort

```
void Shellsort (Item* A, int n) {
int i, j;
int h = 1;
Item aux;

  do /* calcula o h inicial */
    h = h * 3 + 1;
  while (h < n);

  do {
    h = (h-1)/3; /* atualiza o valor de h */
    for(i = h; i < n; i++) {
      aux = A[i]; j = i;

      /* efetua comparações entre elementos com distância h */
      while (A[j - h].Chave > aux.Chave) {
        A[j] = A[j - h]; j -= h;
        if (j < h) break;
      }
      A[j] = aux;
    }
  } while (h != 1);
}
```


ShellSort

- **Análise**
  - A razão da eficiência do algoritmo ainda não é completamente conhecida.
  - A complexidade depende da escolha da sequência de incrementos
  - Existem várias possibilidades com diferentes resultados. Mas há algumas regras básicas:
    - Por exemplo, cada incremento não deve ser múltiplo do anterior
  - De qualquer forma, a análise de complexidade do shellsort leva a problemas matemáticos complexos

ShellSort

- **Análise**
  - Conjecturas referente ao número de comparações para a sequência de Knuth:

    Conjectura 1 : $C(n) = O(n^{1,25})$

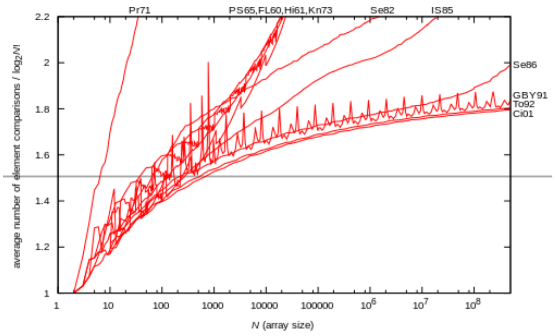
    Conjectura 2 : $C(n) = O(n\ (ln\ n)^2)$

Shellsort complexity and gap sequences

General term ($k \geq 1$)	Concrete gaps	Worst-case time complexity	Author and year of publication
$\left\lfloor \frac{N}{2^k} \right\rfloor$	$\left\lfloor \frac{N}{2} \right\rfloor, \left\lfloor \frac{N}{4} \right\rfloor, \ldots, 1$	$\Theta\left(N^2\right)$ [e.g. when $N = 2^p$]	Shell, 1959[4]
$2^k - 1$	$1, 3, 7, 15, 31, 63, \ldots$	$\Theta\left(N^{\frac{1}{2}}\right)$	Hibbard, 1963[9]
Successive numbers of the form 2^p3^q (3-smooth numbers)	$1, 2, 3, 4, 6, 8, 9, 12, \ldots$	$\Theta\left(N \log^2 N\right)$	Pratt, 1971[11]
$\frac{3^k - 1}{2}$, not greater than $\left\lceil \frac{N}{3} \right\rceil$	$1, 4, 13, 40, 121, \ldots$	$\Theta\left(N^{\frac{1}{2}}\right)$	Pratt, 1971,[11] Knuth, 1973[3]
$\left\lceil \frac{1}{5}\left(9 \cdot \left(\frac{9}{4}\right)^{k-1} - 4\right)\right\rceil$	$1, 4, 9, 20, 46, 103, \ldots$	Unknown	Tokuda, 1992[14]

Tokuda's sequence, defined by the simple formula $h_k = \lceil h'_k \rceil$, where $h'_k = 2.25h'_{k-1} + 1$, $h'_1 = 1$, can be recommended for practical applications

Shellsort: experimental comp. complexity



ShellSort

- Vantagens:
  - Shellsort é uma boa opção para arquivos de tamanho moderado.
  - Sua implementação é simples e requer uma quantidade de código pequena.
  - In-place e adaptável
  - Não realiza chamadas recursivas (não usa *call stack*)
- Desvantagens:
  - O tempo de execução do algoritmo é sensível à escolha do tamanho do *gap*
  - Análise de complexidade: dependendo do gap, ainda é um problema em aberto
  - O método não é estável.

Exercícios

1. Mostre um exemplo de entrada que demonstra que o método de ordenação seleção não é estável.
2. Mostre um exemplo que demonstra que o Shellsort é instável para sequencia h=1,2
3. O método da bolha não é adaptável, altere o código para que ele se torne adaptável.
4. Qual dos métodos: bolha, inserção e seleção executa menos comparações para um vetor de entrada contendo valores idênticos.