BCC202 - Estrutura de Dados I

Aula 15: Ordenação: ShellSort

Reinaldo Fortes

Universidade Federal de Ouro Preto, UFOP
Departamento de Computação, DECOM

Website: **www.decom.ufop.br/reifortes**
Email: **reifortes@iceb.ufop.br**

Material elaborado com base nos slides do Prof. Túlio Toffolo (curso de 2013/01).

2013/02

Conteúdo

Conteúdo

ShellSort

- Proposto por **Donald Shell** em 1959.
- É uma extensão do ***InsertionSort***.
- Problema com o algoritmo de *ordenação por inserção*:
  - Troca itens adjacentes para determinar o ponto de inserção.
  - São efetuadas $\boldsymbol{n-1}$ comparações e movimentações quando o menor item está na posição mais à direita do vetor.
- O método de **Shell** contorna este problema permitindo trocas de registros distantes um do outro.

Idéia básica

- Os itens separados de ***h*** posições são rearranjados, gerando sequências ordenadas.
- É dito que cada seqüência está **h-ordenada**.
- Depois, o valor de ***h*** é reduzido progressivamente, até atingir o valor 1, que resultará no vetor completamente ordenado.

Exemplo

Vetor Inicial:	45	56	12	43	95	19	8	67

Exemplo

Vetor Inicial:	45	56	12	43	95	19	8	67

h = 4	45	56	12	43	95	19	8	67

Exemplo

Vetor Inicial:	45	56	12	43	95	19	8	67

h = 4	45	19	12	43	95	56	8	67

Exemplo

Vetor Inicial:	45	56	12	43	95	19	8	67

h = 4	45	19	8	43	95	56	12	67

Exemplo

Vetor Inicial:	45	56	12	43	95	19	8	67

h = 4	45	19	8	43	95	56	12	67

Exemplo

Vetor Inicial:	45	56	12	43	95	19	8	67

h = 4	45	19	8	43	95	56	12	67

Exemplo

Vetor Inicial:	45	56	12	43	95	19	8	67
h = 4	45	19	8	43	95	56	12	67
h = 2	8	19	12	43	45	56	95	67

Exemplo

Vetor Inicial:	45	56	12	43	95	19	8	67
h = 4	45	19	8	43	95	56	12	67
h = 2	8	19	12	43	45	56	95	67

Exemplo

Vetor Inicial:	45	56	12	43	95	19	8	67
h = 4	45	19	8	43	95	56	12	67
h = 2	8	19	12	43	45	56	95	67

Exemplo

Vetor Inicial:	45	56	12	43	95	19	8	67
h = 4	45	19	8	43	95	56	12	67
h = 2	8	19	12	43	45	56	95	67
h = 1	8	12	19	43	45	56	67	95

Exemplo

Vetor Inicial:	45	56	12	43	95	19	8	67
h = 4	45	19	8	43	95	56	12	67
h = 2	8	19	12	43	45	56	95	67
h = 1	8	12	19	43	45	56	67	95

Observações

- À medida que h decresce, o vetor vai ficando cada vez mais próximo da ordenação.
- Com $h = 1$, o algoritmo se comporta exatamente igual ao **InsertionSort**.
- Mas, como podemos escolher uma boa sequência de valores para h?

Escolha de h

- Como escolher o valor de $\boldsymbol{h}$?
  - Para $\boldsymbol{s = 1}$: $\boldsymbol{h(s) = 1}$;
  - Para $\boldsymbol{s > 1}$: $\boldsymbol{h(s) = 3h(s - 1) + 1}$
- A sequência para $\boldsymbol{h}$ corresponde a:

$$\mathbf{1, 4, 13, 40, 121, 364, 1.093, 3.280, ...}$$

- *Knuth (1973, p. 95)* mostrou experimentalmente que esta seqüência é difícil de ser batida por mais de 20% em eficiência.
- Para conhecer outras sequências, clique aqui.

Conteúdo

ShellSort

```
1  void shellSort(TItem *v, int n) {
2    int i, j, h;
3    TItem aux;
4
5    for(h = 1; h < n; h = 3 * h + 1); /* h inicial. */
6
7    do {
8      h = (h - 1) / 3; /* atualiza h. */
9      for(i = h ; i < n ; i++) {
10       aux = v[i];
11       j = i;
12       while (v[j - h].chave > aux.chave) {
13         v[j] = v[j - h];
14         j -= h;
15         if (j < h) break;
16       }
17       v[j] = aux;
18     }
19   } while (h != 1);
20 }
```

Conteúdo

Análise

- A complexidade do algoritmo ainda não é conhecida.
- Ninguém foi capaz de analisar o algoritmo até hoje.
  - Depende de problemas matemáticos muito difíceis.
  - A começar pela própria sequência de incrementos.
  - O que se sabe é que cada incremento não deve ser múltiplo do anterior.
    - Por quê? Observe o exemplo do slide 6.

Análise

- Conjecturas referentes ao número de comparações para a sequência de Knuth:
  - Conjectura 1: $\boldsymbol{C(n) = O(n^{1.25})}$.
  - Conjectura 2: $\boldsymbol{C(n) = O(n\ (ln\ n)^2)}$.

Conteúdo

Características

Vantagens

- Shellsort é uma ótima opção para arquivos de tamanho moderado.
- Sua implementação é simples e requer uma quantidade de código pequena.

Desvantagens

- O tempo de execução do algoritmo é sensível à ordem inicial do arquivo.
- O método **não é estável**.

Conteúdo

Conclusão

- Nesta aula tivemos contato com o *algoritmo de ordenação* chamado **Shellsort**.
- Trata-se de uma adaptação do algoritmo **InsertionSort**.
- Sua complexidade não é bem conhecida, mas pode ser eficiente para arquivos pequenos.

Conclusão

- Quadro comparativo dos métodos de ordenação:

Algoritmo	Comparações			Movimentações			Espaço	Estável	In situ
	Melhor	Médio	Pior	Melhor	Médio	Pior			
Bubble	$O(n^2)$			$O(n^2)$			$O(1)$	**Sim**	**Sim**
Selection	$O(n^2)$			$O(n)$			$O(1)$	**Não***	**Sim**
Insertion	$O(n)$	$O(n^2)$		$O(n)$	$O(n^2)$		$O(1)$	**Sim**	**Sim**
Merge	$O(n\ log\ n)$			–			$O(n)$	**Sim**	**Não**
Quick	$O(n\ log\ n)$		$O(n^2)$	–			$O(n)$	**Não***	**Sim**
Shell	$O(n^{1.25})$ ou $O(n\ (ln\ n)^2)$			–			$O(1)$	**Não**	**Sim**

* Existem versões estáveis.

Conclusão

- A tarefa de ordenação é muito importante, ela é uma necessidade básica para a solução de muitos problemas.
- *Próxima aula*: *Filas de Prioridade*.
- Dúvidas?

Conteúdo

Exercício 01

- Dada a sequência de números: `3 4 9 2 5 1 8`.
- Ordene em ordem crescente utilizando o algoritmo aprendido em sala (**ShellSort**), apresentando a sequência dos números a cada passo (Teste de Mesa).