Algoritmos de Ordenação (Parte 3)

Prof. Jefferson T. Oliva

Algoritmos e Estrutura de Dados I (AE22CP)
Engenharia de Computação
Departamento Acadêmico de Informática (Dainf)
Universidade Tecnológica Federal do Paraná (UTFPR)
Campus Pato Branco

DAINF
DEPARTAMENTO ACADÊMICO DE INFORMÁTICA
CÂMPUS PATO BRANCO

UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Sumário

- Ordenação por Inserção
  - Insert sort
  - Shell sort

Ordenação por Inserção

- Classifica um conjunto de registros inserindo registros em um arquivo classificado existente

- Exemplos de algoritmos de ordenação por inserção
  - *Insertion sort*
  - *Shell sort*

Ordenação por Inserção

Insertion sort

- É o método que consiste em inserir informações num conjunto já ordenado
- Algoritmo utilizado pelo jogador de cartas
  - As cartas são mantidas ordenadas nas mãos dos jogadores
  - Quando o jogador compra ou recebe uma nova carta, ele procura a posição que a nova carta deverá ocupar
- Bastante eficiente para pequenas entradas

Ordenação por Inserção

Insertion sort

- Implementação

```
void insertsort(int v[], int n){
   int i, x;

   for (i = 1; i < n; i++){
     x = v[i];

     for (j = i - 1; (j >= 0) && (x < v[j]); j--)
       v[j + 1] = v[j];

     v[j + 1] = x;
   }
}
```

Ordenação por Inserção

Insertion sort

- Exemplo

x	i	j	v[0]	v[1]	v[2]	v[3]	v[4]	v[5]	v[6]	v[7]
-	-	-	25	57	48	37	12	92	86	33
57	1	0	25	57	48	37	12	92	86	33
48	2	1	25	57	57	37	12	92	86	33
48	2	0	25	48	57	37	12	92	86	33
37	3	2	25	48	57	57	12	92	86	33
37	3	1	25	48	48	57	12	92	86	33
37	3	0	25	37	48	57	12	92	86	33
12	4	3	25	37	48	57	57	92	86	33
12	4	2	25	37	48	48	57	92	86	33
12	4	1	25	37	37	48	57	92	86	33
12	4	0	25	25	37	48	57	92	86	33
12	4	-1	12	25	37	48	57	92	86	33
92	5	4	12	25	37	48	57	92	86	33

Ordenação por Inserção

Insertion sort

- Exemplo

x	i	j	v[0]	v[1]	v[2]	v[3]	v[4]	v[5]	v[6]	v[7]
86	6	5	12	25	37	48	57	92	92	33
86	6	4	12	25	37	48	57	86	92	33
33	7	6	12	25	37	48	57	86	92	92
33	7	5	12	25	37	48	57	86	86	92
33	7	4	12	25	37	48	57	57	86	92
33	7	3	12	25	37	48	48	57	86	92
33	7	2	12	25	37	37	48	57	86	92
33	7	1	12	25	33	37	48	57	86	92

Ordenação por Inserção

Insertion sort

- Desempenho do método
  - Melhor caso: $O(n)$
  - Médio caso: $O(n^2)$
  - Pior caso: $O(n^2)$
- Deve ser utilizado quando o arquivo está "quase" ordenado
- Desvantagem: alto custo de movimentação de elementos

Ordenação por Inserção

Insertion sort

- Links interessantes:
  - Dança romana:
    `https://www.youtube.com/watch?v=ROalU379l3U`
  - Simulador gráfico do *insertion sort*:
    `https://visualgo.net/bn/sorting`

Ordenação por Inserção

Shell sort

- Extensão do *insertion sort*
- Contorna o principal problema do *insertion sort* possibilitando troca de registros que estão distantes um do outro
- Tem como objetivo aumentar o passo de movimento dos elementos ao invés das posições adjacentes
- Consiste em classificar sub-arranjos do original
- Esses sub-arranjos contêm todo h-ésimo elemento do arranjo original
- o valor de h é chamado de incremento

Ordenação por Inserção

Shell sort

- Por exemplo, se h é 5, o sub-arranjo consiste dos elementos $x[0]$, $x[5]$, $x[10]$, etc
  - Sub-arranjo 1: $x[0]$, $x[5]$, $x[10]$
  - Sub-arranjo 2: $x[1]$, $x[6]$, $x[11]$
  - Sub-arranjo 3: $x[2]$, $x[7]$, $x[12]$
  - Sub-arranjo 4: $x[3]$, $x[8]$, $x[13]$

- Após a ordenação dos sub-arranjos:
  - Define-se um novo incremento menor que o anterior
  - Gera-se novos sub-arquivos
  - Aplica-se novamente o método da inserção

- O processo é realizado repetidamente até que h seja igual a 1
- O valor de h pode ser definido de várias formas
  - $h(s) = 3h(s-1) + 1$, para $s > 1$
  - $h(s) = 1$, para $s = 1$
- Nessa recorrência, s é o tamanho do conjunto

Ordenação por Inserção

Shell sort

- Implementação

```
void shellsort(int v[], int n){
    int h = 1;
    int x, i, j;

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

    h /= 3;

    while (h >= 1){
      for (i = h; i < n; i++){
          x = v[i];
          j = i;

          while ((j >= h) && (x < v[j - h])){
              v[j] = v[j - h];

              j -= h;
          }

          v[j] = x;
      }

      h /= 3;
    }
}
```

- Exemplo

x	h	i	v[0]	v[1]	v[2]	v[3]	v[4]	v[5]	v[6]	v[7]
-	-	-	25	57	48	37	12	92	86	33
12	4	4	**25**	57	48	37	**12**	92	86	33
92	4	5	12	**57**	48	37	25	**92**	86	33
86	4	6	12	57	**48**	37	25	92	**86**	33
33	4	7	12	57	48	**37**	25	92	86	**33**
33	4	8	12	57	48	33	25	92	86	37
57	1	1	**12**	**57**	48	33	25	92	86	37
48	1	2	**12**	**57**	**48**	33	25	92	86	37
33	1	3	**12**	**48**	**57**	**33**	25	92	86	37
25	1	4	**12**	**33**	**48**	**57**	**25**	92	86	37
92	1	5	**12**	**25**	**33**	**48**	**57**	**92**	86	37
86	1	6	**12**	**25**	**33**	**48**	**57**	**92**	**86**	37
37	1	7	12	25	33	48	57	86	92	37
37	1	8	12	25	33	37	48	57	86	92

- Quando $h = 1$, o comportamento é o mesmo em comparação com o *insertion sort*

Ordenação por Inserção

Shell sort

- Um problema com o *shell sort* ainda não resolvido é a escolha dos incrementos que fornece os melhores resultados
- É desejável que ocorra o maior número possível de interações entre as diversas cadeias
- Ótima opção para arquivos de tamanho moderado
- Tempo de execução sensível à ordem dos dados
- Não estável

Ordenação por Inserção

Shell sort

- A complexidade do algoritmo ainda não é conhecida
- Acredita-se que o custo de tempo é por volta de:
  - Pior caso: $O(n^2)$
  - Melhor caso: proporcional a $O(n^{1,25})$
- Links interessante:
  `https://www.youtube.com/watch?v=CmPA7zE8mx0`

Referências I


Cormen, T. H., Leiserson, C. E., Rivest, R. L., Stein, C.
Introduction to Algorithms.
Third edition, The MIT Press, 2009.


Horowitz, E., Sahni, S. Rajasekaran, S.
Computer Algorithms.
Computer Science Press, 1998.


Rosa, J. L. G.
Métodos de Ordenação. SCE-181 – Introdução à Ciência da Computação II.
Slides. Ciência de Computação. ICMC/USP, 2018.


Ziviani, N.
Projeto de Algoritmos - com implementações em Java e C++.
Thomson, 2007.