Métodos de Ordenação

Parte 2

SCC-214 Projeto de Algoritmos

Prof. Thiago A. S. Pardo

Baseado no material do Prof. Rudinei Goularte

Ordenação por Inserção

- Idéia básica: inserir um dado elemento em sua posição correta em um subconjunto já ordenado
  - Inserção Simples, ou inserção direta
  - Shell-sort, ou classificação de shell ou, ainda, classificação de incremento decrescente

2

Inserção Simples

- Idéia básica
  - Ordenar o conjunto inserindo os elementos em um subconjunto já ordenado
    - No i-ésimo passo, inserir o i-ésimo elemento na posição correta entre x[0],...,x[i-1], que já estão em ordem
      - Elementos são realocados

3

Inserção Simples

- Idéia básica
  - Exemplo

Vetor original

10	30	31	15	50	60	5	22	35	14

Realocando o elemento 15

10	30	31	15	50	60	5	22	35	14

30 e 31 são realocados e 15 é inserido

10	15	30	31	50	60	5	22	35	14

Por que o método se chama inserção simples?

Inserção Simples: exemplo

- X = (44 , 55 , 12 , 42 , 94 , 18 , 06 , 67)
- passo 1 (55) 44 55 12 42 94 18 06 67
- passo 2 (12) 12 44 55 42 94 18 06 67
- passo 3 (42) 12 42 44 55 94 18 06 67
- passo 4 (94) 12 42 44 55 94 18 06 67
- passo 5 (18) 12 18 42 44 55 94 06 67
- passo 6 (06) 06 12 18 42 44 55 94 67
- passo 7 (67) 06 12 18 42 44 55 67 94

5

Exercício

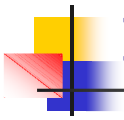
- Para entregar (valendo nota)
  - Em grupos de 2 alunos
  - Implementar em C a Inserção Simples
  - Calcular complexidade de tempo e espaço

6

Inserção Simples

- Implementação

7



Inserção Simples

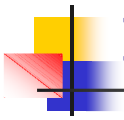
- Complexidade de tempo de melhor caso
  - ?
- Complexidade de tempo de pior caso
  - ?
- Complexidade de espaço: ?

8

Inserção Simples

- Complexidade de tempo de melhor caso
  - Vetor ordenado: O(n)
- Complexidade de tempo de pior caso
  - Vetor ordenado inversamente: $O(n^2)$
- Complexidade de espaço: O(n)

9



Inserção Simples

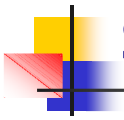
- Inserção simples é eficiente em arquivos "quase" ordenados
- Um dos métodos mais intuitivos

10

Shell-sort

- Shell-sort: melhoria da inserção simples
  - Inserção simples movimenta elementos adjacentes
    - Se o menor elemento estiver na posição mais a direita, n-1 comparações e movimentos são necessários
  - Shell-sort permite a troca de elementos distantes
    - Elementos separados por h posições são ordenados de tal forma que todo h-ésimo elemento está em uma seqüência ordenada
      - Essa seqüência é dita estar h-ordenada

11



Shell-sort

- Exemplo

Vetor original

10	30	31	15	50	60	5	22	35	14
0	1	2	3	4	5	6	7	8	9

12

Shell-sort

- Exemplo

Vetor original

10	30	31	15	50	60	5	22	35	14
0	1	2	3	4	5	6	7	8	9

h=4 ➔ elementos nas posições 0, 4 (0+4) e 8 (4+4)

10	30	31	15	50	60	5	22	35	14
0	1	2	3	4	5	6	7	8	9

13

Shell-sort

- Exemplo

Vetor original

10	30	31	15	50	60	5	22	35	14
0	1	2	3	4	5	6	7	8	9

h=4 ➔ elementos nas posições 0, 4 (0+4) e 8 (4+4)

10	30	31	15	50	60	5	22	35	14
0	1	2	3	4	5	6	7	8	9

4-ordenado

10	30	31	15	35	60	5	22	50	14
0	1	2	3	4	5	6	7	8	9

14

Shell-sort

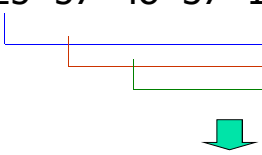
- Idéia básica: dividir a entrada em sub-conjuntos de elementos de distância h e aplicar inserção simples a cada um, sendo que h é reduzido sucessivamente
  - A cada nova iteração, o vetor original está "mais" ordenado

15

Shell-sort: exemplo

- 25 57 48 37 12 92 86 33

Passo1, h=5: 25 57 48 37 12 92 86 33



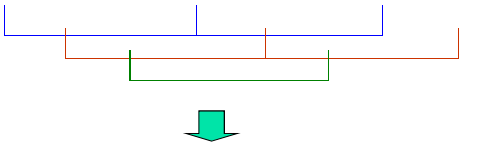
25 57 33 37 12 92 86 48

16

Shell-sort: exemplo

- 25 57 48 37 12 92 86 33

Passo2, h=3: 25 57 33 37 12 92 86 48



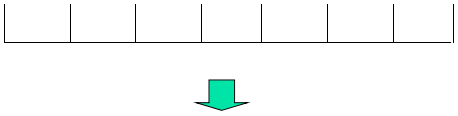
25 12 33 37 48 92 86 57

17

Shell-sort: exemplo

- 25 57 48 37 12 92 86 33

Passo 3, h=1: 25 12 33 37 48 92 86 57



12 25 33 37 48 57 86 92

Quando h=1, shellsort=?

18

Shell-sort: exemplo

- 25 57 48 37 12 92 86 33

Passo 3, h=1: 25 12 33 37 48 92 86 57

12 25 33 37 48 57 86 92

Quando h=1, shellsort=inserção simples

19

Shell-sort

- Os índices h são os incrementos que são adicionados a cada posição do vetor para se ter o próximo elemento do sub-conjunto
- A cada iteração, h decresce
  - Daí o nome "incrementos decrescentes" do método
- O último incremento deve sempre ser 1

20

Shell-sort

- n=15, h=5

1 – x[0] x[5] x[10]
2 – x[1] x[6] x[11]
3 – x[2] x[7] x[12]
4 – x[3] x[8] x[13]
5 – x[4] x[9] x[14]

- O i-ésimo elemento do j-ésimo conjunto é: x[(i-1)*h+j–1]

21

Shell-sort

- 25 57 48 37 12 92 86 33

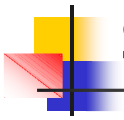
Passo 1 (incremento 5):
(x[0], x[5])
(x[1], x[6])
(x[2], x[7])
(x[3])
(x[4])
Passo 2 (incremento 3):
(x[0], x[3], x[6])
(x[1], x[4], x[7])
(x[2], x[5])
Passo 3 (incremento 1):
(x[0], x[1], x[2], x[3], x[4], x[5], x[6], x[7])

22

Shell-sort

- Por que o método tem esse nome?

23



Shell-sort

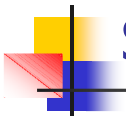
- Implementação

24

Shell-sort

```
void shellsort(int v[], int n, int incrementos[], int numinc)
{
        int incr, i, j, h, aux;
        for (incr=0; incr<numinc; incr++) {
                h=incrementos[incr];
                for (i=h; i<n; i++) {
                        aux=v[i];
                        for (j=i-h; j>=0 && v[j]>aux; j-=h)
                                v[j+h]=v[j];
                        v[j+h]=aux;
                }
        }
}
```

25



Shell-sort

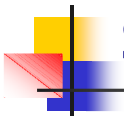
- Exercício
  - Executar o algoritmo anterior para o vetor (25 57 48 37 12 92 86 33)
  - 3 incrementos: 5, 3 e 1

26

Shell-sort

- Foi demonstrado que, com uma seqüência adequada de incrementos de h, shell-sort é aproximadamente $O(n(\log n)^2)$
  - Tarefa para casa: buscar essa demonstração/prova da complexidade do shellsort

27



Shell-sort

- Escolha dos incrementos é importante
  - Knuth (1973) sugere
    - Defina uma função recursiva h tal que:
      - h(1) = 1 e h(i + 1) = 3 * h(i) + 1
    - Exemplo de seqüência de incrementos: 1, 4, 13, 40, 121, 364, 1.093, 3.280, etc.
      - Aplicada no sentido inverso

28