



Métodos de Ordenação

Prof. Alneu de Andrade Lopes
Prof. Thiago A. S. Pardo



Ordenação por Inserção

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



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 $x[0], \dots, x[i-1]$, que já estão em ordem
 - Elementos são realocados

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



Inserção Simples

- Em grupos de 4 alunos (valendo nota)
 - Implementar Inserção Simples
 - Calcular complexidade



Inserção Simples

```
void insercao(int X[], int n) {  
    for (k = 1; k < n; k++) {  
        y = X[k];  
        for (i = k-1; i >= 0 && X[i] > y; i--)  
            X[i+1] = X[i];  
        X[i+1] = y;  
    }  
}
```



Inserção Simples

- $O(n^2)$
 - $(n-1) + (n-2) + \dots + 2 + 1 = (n-1) * n/2$ comparações
- Vetor ordenado: $O(n)$
- Vetor ordenado inversamente: $O(n^2)$
- Pouco espaço: $O(n)$
- Realiza menos comparações que o *Bubble-sort*
 - A parte ordenada não é comparada novamente a cada iteração



Shell-sort

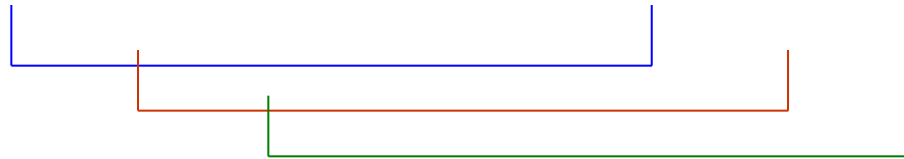
- Inserção simples é eficiente em arquivos quase ordenados
- Shell-sort: melhoria da inserção simples
 - Idéia básica: dividir a entrada em k sub-conjuntos e aplicar inserção simples a cada um, sendo que k é reduzido sucessivamente
 - A cada nova iteração, o vetor original está “mais” ordenado



Shell-sort: exemplo

■ 25 57 48 37 12 92 86 33

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



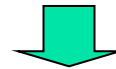
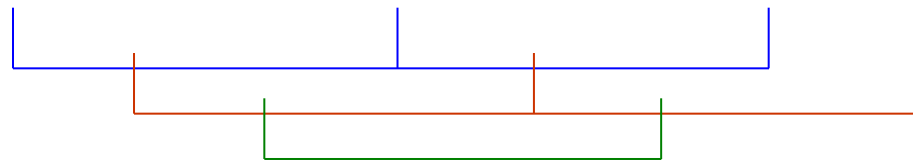
25 57 33 37 12 92 86 48



Shell-sort: exemplo

■ 25 57 48 37 12 92 86 33

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



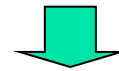
25 12 33 37 48 92 86 57



Shell-sort: exemplo

■ 25 57 48 37 12 92 86 33

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



12 25 33 37 48 57 86 92



Shell-sort

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



Shell-sort

- $k = 5, n = 15$

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:
 $x[(i-1) * k + j - 1]$



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])



Shell-sort

Vetor com incrementos (k)

Número de elementos
no vetor incrmnts

```
void shell-sort (int x[], int n, int incrmnts[], int numinc) {  
    int incr, j, k, span, y;  
    for (incr = 0; incr < numinc; incr++) {  
        span = incrmnts[incr];  
        for (j = span; j < n; j++) {  
            y = x[j];  
            for (k = j - span; k >= 0 && x[k] > y; k -= span)  
                x[k+span] = x[k];  
            x[k+span] = y;  
        }  
    }  
}
```




Shell-sort

- Exercício

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



Shell-sort

- Foi demonstrado que, com uma seqüência adequada de incrementos de k , shell-sort é aproximadamente $O(n(\log n)^2)$
 - Prova da eficiência do shell-sort está além do escopo desta disciplina



Shell-sort

- Escolha dos incrementos
 - Knuth sugere:
 - Defina uma função recursiva h tal que:
 - $h(1) = 1$ e $h(i + 1) = 3 * h(i) + 1$
 - Seja x o menor inteiro tal que $h(x) \geq n$:
 - numinc será $x - 2$
 - $\text{incrmnts}[i]$ será $h(\text{numinc} - i + 1)$ para i de 1 até numinc