

TECNOLOGIA EM INFORMÁTICA

Estrutura de Dados

Prof. Glauco da Silva
contato: glaucoslv@gmail.com

Roteiro

- *Objetivos*
- *Métodos de ordenação eficientes*
 - *Shellsort*
 - *Heapsort**
 - *Quicksort*
 - *Mergesort*
- *Exercícios*

Roteiro

- ***Objetivos***
- *Métodos de ordenação eficientes*
 - *Shellsort*
 - *Heapsort**
 - *Quicksort*
 - *Mergesort*
- *Exercícios*

Objetivos

- *Apresentar os métodos de ordenação eficientes, mostrando sua forma de trabalho e seu funcionamento para ordenação de valores*
- *Mostrar porque os métodos são melhores que os métodos elementares*

Roteiro

- *Objetivos*
- *Métodos de ordenação eficientes*
 - *Shellsort*
 - *Heapsort**
 - *Quicksort*
 - *Mergesort*
- *Exercícios*

Shell Sort

- *Extensão da idéia de inserção direta*
- *Inserção faz trocas adjacentes*
- *Pior caso: fazer $n-1$ trocas*
- *Shell faz trocas a uma certa distância*
- *Leva elementos mais rapidamente para o ponto correto*

Shell Sort

- *Fragmentar o arquivo de h em h elementos e ordená-los*

$$H = 2$$

12	57	48	33	25	92	86	37
12	57	48	33	25	92	86	37
12	57	48	33	25	92	86	37
12	57	25	33	48	92	86	37
12	57	25	33	48	92	86	37
12	57	25	33	48	92	86	37
12	57	25	33	48	92	86	37
12	33	25	57	48	92	86	37
12	33	25	57	48	92	86	37
12	33	25	57	48	37	86	92
12	33	25	37	48	57	86	92

Shell Sort

- *Fragmentar o arquivo de h em h elementos e ordená-los*

$$H = 1$$

12	33	25	37	48	57	86	92
12	33	25	37	48	57	86	92
12	33	25	37	48	57	86	92
12	25	33	37	48	57	86	92
12	25	33	37	48	57	86	92
12	25	33	37	48	57	86	92
12	25	33	37	48	57	86	92
12	25	33	37	48	57	86	92
12	25	33	37	48	57	86	92

Shell Sort

- *Na última passada, nenhum elemento pode se mover mais do que uma posição*
- *Como definir a seqüência de h 's? (Kurth, 1973)*

```
h = 1;  
do {  
    h = 3 * h + 1;  
} while (h < limite);
```

1, 4, 13, 40, 121, 364, 1093, 3280...

Shell Sort

- *Algoritmo Básico*

Identificar o maior h possível

Enquanto $h > 0$

// Ordena por inserção em incrementos de h registros

// Calcula o próximo incremento

$h /= 3$

Shell Sort

- *Possível implementação (Linguagem C)*

```
void shellsort (int x[], int limite) {
    int i, j, t, h;
    limite--;
    h=1;
    do {
        h=3*h+1;
    } while (h<limite);

    while (h>0) {
        for (i=h; i<=limite; i++) {
            t=x[i];
            for (j=i-h; ((j>=0) && (x[j]>t)); j=j-h)
                x[j+h]=x[j];
            x[j+h]=t;
        }
        h/=3;
    }
}
```

Roteiro

- *Objetivos*
- ***Métodos de ordenação eficientes***
 - *Shellsort*
 - *Heapsort**
 - ***Quicksort***
 - *Mergesort*
- *Exercícios*

Quick Sort

- *Baseado em trocas a longa distância*
- *Procedimento*
 - *Particionar o arquivo em porções desordenadas, cada vez menores*
 - *Cada pivô da partição está na posição correta*
 - *Recursivamente, o arquivo está ordenado*

Quick Sort

- *Passos do algoritmo*
 - *Tomar aleatoriamente uma chave X*
 - *Percorrer o arquivo da esquerda para a direita até achar alguma chave maior ou igual à X*
 - *Percorrer o arquivo da direita para a esquerda até achar alguma chave menor ou igual à X*
 - *Trocar os elementos*
 - *Repetir até que haja cruzamento de percurso*
 - *Aplicar o mesmo procedimento à esquerda e à direita de X*

Quick Sort



Aplicar o mesmo procedimento às duas partições

Quick Sort

- *Ponto chave*
 - *Escolha do pivô*
- *Pior caso: escolher sempre o maior ou o menor valor*
- *Melhor caso: escolher sempre o valor médio*

Quick Sort

– Algoritmo Básico

Ordenar (arq, 0, n)

Ordenar (arq, i, j)

Escolher x

Repetir

Enquanto $arq[i] < x$, $i++$

Enquanto $arq[j] > x$, $j--$

Se $i \leq j$

Trocar $arq[i]$ com $arq[j]$

$i++$; $j--$

Enquanto $i > j$

Ordenar porção entre o início e x

Ordenar porção entre x e o final

Roteiro

- *Objetivos*
- ***Métodos de ordenação eficientes***
 - *Shellsort*
 - *Heapsort**
 - *Quicksort*
 - ***Mergesort***
- *Exercícios*

Merge Sort

- *Em geral utilizado na ordenação de grandes volumes de dados, mantidos em arquivos*
 - *Por isso é tido como um método de ordenação externa*
- *Faz a intercalação de arquivos ordenados*
 - *Intercalação é feita diretamente de e para memória externa*

Merge Sort

- *Passos do algoritmo*
 - *Ler o que for possível do arquivo de entrada*
 - *Ordenar e gravar em memória externa*
 - *Se ainda houver registros na entrada*
 - *Ler mais o que for possível*
 - *Ordenar e gravar em memória externa*
 - *Intercalar com o arquivo da passagem anterior e armazenar em memória externa*
 - *Renomear o último arquivo para o nome final*

Merge Sort

- Algoritmo de intercalação

Ler o primeiro registro do primeiro arquivo (v1)

Se não houver mais registros, v1 = null

Ler o primeiro registro do segundo arquivo (v2)

Se não houver mais registros, v2 = null

Enquanto houver registros em qualquer arquivo

Se $v1 < v2$

Escrever v1 na saída

Ler próximo registro do primeiro arquivo (v1)

Se não houver mais registros, v1 = null

Senão

Escrever v2 na saída

Ler próximo registro do segundo arquivo (v2)

Se não houver mais registros, v2 = null

Roteiro

- *Objetivos*
- *Métodos de ordenação eficientes*
 - *Shellsort*
 - *Heapsort**
 - *Quicksort*
 - *Mergesort*
- ***Exercícios***

Exercícios

- *Implementar os algoritmos dados em Java*
- *Ordenar o vetor a seguir pelos métodos apresentados em aula e verificar o número de trocas necessárias*

43	31	47	59	92	23	2
----	----	----	----	----	----	---