

Pesquisa e Ordenação

- Pesquisa de valores em sequências
 - pesquisa sequencial
 - pesquisa binária
- Ordenação de sequências
 - ordenação seleção
 - ordenação por bolha
- Exemplos

Pesquisa de valores em Arrays

- Em inúmeros problemas temos a necessidade de procurar por valores em sequências. A esta tarefa designa-se pesquisa.
- Existem vários algoritmos em programação para a pesquisa de valores em sequências mas nesta disciplina vamos apenas analisar dois dos mais simples: **pesquisa sequencial** e **pesquisa binária**.
- A pesquisa é uma tarefa computacionalmente dispendiosa se estivermos a tratar grandes quantidades de informação.
- O desenvolvimento de algoritmos eficientes torna-se essencial e, como vamos ver, a complexidade dos algoritmos não é sempre a mesma.

Pesquisa sequencial (1)

- A pesquisa sequencial consiste em analisar todos os elementos da sequência de forma metódica.
- A pesquisa começa por analisar o primeiro valor da sequência e percorre todos os seus valores até encontrar o valor pretendido ou até atingirmos o último elemento.
- Este método é normalmente demorado e depende da dimensão da sequência, mas não depende do arranjo dos valores.
- Em todos os algoritmos de pesquisa é sempre necessária uma forma de “sinalizar” que não encontrámos o valor pretendido.

Pesquisa sequencial (2)

```
public static int pesquisaSequencial(int seq[],int
    nElem, int valor ) {
    int n=0;
    int pos = -1; // inicia com um valor inválido

    do {
        if(seq[n++] == valor) {
            pos = n-1;
        }
    } while(pos == -1 && n <nElem);
    return pos;
}
```

Pesquisa binária (1)

- Se tivermos informação à priori sobre os elementos da sequência, podemos acelerar o processo de pesquisa.
- Se a sequência estiver ordenada por ordem crescente ou decrescente, podemos fazer pesquisa binária.
- O algoritmo começa por selecionar o elemento central da sequência e compara-o com o elemento procurado.
- Se o elemento foi maior, podemos excluir a primeira metade da sequência, caso contrário podemos excluir a segunda metade.
- O processo é repetido até que o elemento seja o procurado ou até deixarmos de ter elementos para analisar.
- Se a lista tiver tamanho 2^n , encontra o valor, no máximo, em n tentativas.

Pesquisa binária (2)

```
public static int pesquisaBinaria(int[] lista, int
valor) {
    int inicio=0, fim=lista.length-1, meio;
    int haValor= -1;
    do {
        meio=(inicio+fim)/2;
        if (valor > lista[meio] ) {
            inicio=meio+1;
        } else if (valor < lista[meio]) {
            fim=meio-1;
        } else {
            haValor = meio;
        }
    } while(haValor == -1 && inicio <= fim );
    return haValor;}
}
```

	1a	2a	3a
3	0 ← 0	0	0
23	1	1	1
32	2	2	2
65	3 ← 3	3	3
77	4	4 ← 4	4 ← 4
81	5	5 ← 5	5
95	6 ← 6	6 ← 6	6

Como utilizar...

...

```
System.out.print("Valor a procurar: ");  
valor = sc.nextInt();  
// ind = pesquisaSequencial(seq_main, nElem_main, valor);  
ind = pesquisaBinaria(seq_main, valor);  
if(ind != -1){  
    System.out.println("O numero está na pos " + ind);  
} else {  
    System.out.println("O numero não existe")  
}
```

...

Ordenação de sequências

- Em outros problemas temos a necessidade de manter as sequências ordenadas.
- Existem vários algoritmos em programação para a ordenação de sequências mas nesta disciplina vamos apenas analisar dois: **ordenação por seleção** e **ordenação por bolha**.
- Na ordenação por seleção vamos colocando em cada posição da sequência de n valores o valor correto, o máximo ou o mínimo, conforme a ordem da ordenação, começando no primeiro, depois o segundo, ... até $n-1$.
- Na ordenação por bolha vamos comparando pares de valores da sequência e trocamos se fora de ordem. Repetimos o processo enquanto houver trocas.

Ordenar por Seleção (*Selection Sort*)

Lista de tamanho n

1a	2a	3a	4a	5a	6a	
55	7	7	7	7	7	7
23	23	14	14	14	14	14
32	32	32	23	23	23	23
7	55	55	55	27	27	27
14	14	23	32	32	32	32
27	27	27	27	55	55	45
45	45	45	45	45	45	55

- Repetir $n - 1$ vezes

- Procurar o mínimo (ou máximo) da lista não ordenada (**verde**);
- Trocar mínimo com 1ª posição da lista não ordenada;
- lista ordenada (**amarelo**)
corresponde às primeiras posições que vão ficando com os mínimos sucessivos;

Ordenar por seleção - código

```
public static void ordenacaoSel(int seq[], int n){
    int tmp, i, j;
    for(i = 0 ; i < n - 1 ; i++){ // fixamos uma posição
        for(j = i + 1 ; j < n ; j++){ //percorremos as outras
            if(seq[i] > seq[j]) // se mínimo, trocamos
            {
                tmp = seq[i];
                seq[i] = seq[j];
                seq[j] = tmp;
            }
        }
    }
}
```

Ordenar por Bolha (*Bubble sort*)

1a	2a	3a	4a	5a	6a		1a	2a	3a	4a	5a	
55	23	23	23	23	23	23	23	23	23	23	23	23
23	55	32	32	32	32	32	32	32	7	7	7	7
32	32	55	7	7	7	7	7	7	32	14	14	14
7	7	7	55	14	14	14	14	14	14	32	27	27
14	14	14	14	55	27	27	27	27	27	32	45	45
27	27	27	27	27	55	45	45	45	45	45	55	55
45	45	45	45	45	45	55	55	55	55	55	55	55

- Percorre a lista não ordenada (verde) comparando elementos sucessivos e troca-os se o 1º for maior que o 2º
- No fim da 1ª passagem o maior fica no fim da lista (azul), ficando ordenado (para a ordenação decrescente);
- No fim da 2ª passagem o 2º maior fica na penúltima posição;

Ordenar por Bolha (2)

1a	2a	3a	4a	
23	7	7	7	7
7	23	14	14	14
14	14	23	23	23
27	27	27	27	27
32	32	32	32	32
45	45	45	45	45
55	55	55	55	55

1a	2a	3a	
7	7	7	7
14	14	14	14
23	23	23	23
27	27	27	27
32	32	32	32
45	45	45	45
55	55	55	55

1a	2a	
7	7	7
14	14	14
23	23	23
27	27	27
32	32	32
45	45	45
55	55	55

1a	
7	7
14	14
23	23
27	27
32	32
45	45
55	55

Ordenada –
Não há trocas

Ordenada
– fim lista

- Repete o processo até que falem ordenar apenas 2 elementos (para lista de tamanho n , são $n-1$ vezes);
- Ou até quando não hajam trocas (melhora eficiência);

Ordenar por Bolha -código

```
public static void ordenarBolha(int[] seq, int n){
    int tmp, i, j;
    int nlo = 0; // número de valores da lista ordenada
    boolean trocas;
    do{
        trocas = false; // partimos do principio que já está...
        for(i = 0 ; i < n -1 - nlo; i++){
            if(seq[i] > seq[i+1]){
                tmp = seq[i];
                seq[i] = seq[i+1];
                seq[i+1] = tmp;
                trocas = true; // houve trocas...
            }
        }
        nlo++; // aumenta lista de valores ordenados
    }while(trocas); // enquanto houver trocas repetimos
}
```

Como utilizar

...

```
int nElem = 0;
```

```
int seq[] = new int[100];
```

```
nElem = Leitura(seq);
```

```
escrita(seq, nElem);
```

```
ordenacaoSel(seq, nElem);
```

```
// ou ordenacaoBolha(seq, nElem);
```

```
escrita(seq, nElem); // os valores serão mostrados ordenados
```

...

Muitos outros algoritmos...

- Inserção
- Fusão
- *QuickSort*
- ...