Programação 1

Departamento de Electrónica, Telecomunicações e Informática

Universidade de Aveiro

`http://moodle.ua.pt/`

Aula 11

deti universidade de aveiro
departamento de electrónica, telecomunicações e informática

- Pesquisa de valores em sequências
  - pesquisa sequencial
  - pesquisa binária
- Ordenação de sequências
  - ordenação sequencial
  - ordenação por flutuação
- 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 sequencia 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 pos = -1; // inicializamos com um valor inválido
  for(int i = 0 ; i < nElem ; i++)
  {
     if(seq[i] == valor)
     {
        pos = i;
        break;
     }
  }
  return pos;
 }
```

Pesquisa binária (1)

- Se tivermos informação à priori sobre os elementos da sequência, podemos acelerar o processo de pesquisa.
- Por exemplo, 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.

Pesquisa binária (2)


```
public static int PesquisaBinaria(int seq[], int nElem, int valor){
 int pos = -1;
 int inicio = 0, fim = nElem - 1, meio;
 while(inicio <= fim){
    meio = (fim + inicio) / 2;
    if(seq[meio] == valor){
       pos = meio;
       break;
    }
    if(seq[meio] > valor)
       fim = meio - 1;
    else
       inicio = meio + 1;
 }
 return pos;
}
```

Como utilizar...


```
...
System.out.print("Valor a procurar: “);
valor = sc.nextInt();
ind = PesquisaSequencial(seq_main, n_main, valor);
// ind = PesquisaBinaria(seq_main, n_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 sequencial** e **ordenação por flutuação**.
- Na ordenação sequencial vamos colocando em cada posição da sequência o valor correcto, começando no primeiro.
- Na ordenação por flutuação vamos comparando pares de valores da sequência e trocamos se fora de ordem. Repetimos o processo enquanto houver trocas.

Ordenação sequencial


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

Ordenação por flutuação


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

Como utilizar


```
...
int nElem = 0;
int seq[] = new int[100];

nElem = Leitura(seq);
Escrita(seq, nElem);
OrdenacaoSeq(seq, nElem);
// ou OrdenacaoFlutuacao(seq, nElem);
Escrita(seq, nElem); // os valores serão mostrados ordenados
...
```

Muitos outros algoritmos...

deti universidade de aveiro
departamento de electrónica, telecomunicações e informática

- Inserção
- Fusão
- QuickSort
- Odd–even sort
- ...