

Programação 2

Jordana S. Salamon

jssalamon@inf.ufes.br

DEPARTAMENTO DE INFORMÁTICA
CENTRO TECNOLÓGICO
UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO

Busca em Vetores

- Imagine como seria buscar um número nos contatos de seu telefone se os nomes das pessoas não estivessem listados em ordem alfabética.

Busca em Vetores

- ▶ Seria muito complicado, certo?
- ▶ Para 5 registros, fácil.
- ▶ Para 50 registros, nem tanto.
- ▶ Para 500...



Ordenação de Vetores

- ▶ A ordenação ou classificação de registros consiste em organizá-los em ordem crescente ou decrescente e assim facilitar a recuperação desses dados.
- ▶ A ordenação tem como objetivo **facilitar as buscas e pesquisas de ocorrências de determinado elemento em um conjunto ordenado.**

Ordenação de Vetores

- ▶ Existem vários tipos de algoritmos de ordenação.
- ▶ Alguns deles são:
 - ▶ Ordenação por bolha (Bubble Sort)
 - ▶ Ordenação por seleção (Selection Sort)
 - ▶ Ordenação por Inserção (Insertion Sort)
 - ▶ Ordenação “rápida” (Quick Sort)
- ▶ Por enquanto, veremos a ordenação por bolha!

Bubble Sort

- ▶ Percorra o vetor da esquerda para direita, comparando elementos vizinhos.
 - ▶ Os que estiverem fora de ordem, troque-os de posição.
- ▶ Considerando o vetor inicial abaixo:

7	5	9	3	2
---	---	---	---	---

Bubble Sort

► Primeira iteração:

7	5	9	3	2
----------	----------	----------	----------	----------

- Comparando $v[0]$ com $v[1]$ → troca
- Comparando $v[1]$ com $v[2]$ → não troca
- Comparando $v[2]$ com $v[3]$ → troca
- Comparando $v[3]$ com $v[4]$ → troca

► O vetor ficará

5	7	3	2	9
----------	----------	----------	----------	----------



Bubble Sort

► Iterações seguintes:

7	5	9	3	2
---	---	---	---	---

disposição inicial

5	7	3	2	9
---	---	---	---	---

1ª iteração

5	3	2	7	9
---	---	---	---	---

2ª iteração

3	2	5	7	9
---	---	---	---	---

3ª iteração

2	3	5	7	9
---	---	---	---	---

4ª iteração

Bubble Sort

► E o algoritmo?

```
void bolha( int v[], int n){  
  
    int i, j, aux;  
    for(j = n; j > 0; j--){  
        for(i = 0; i < j; i++){  
            if (v[i] > v[i+1]){  
                aux = v[i];  
                v[i] = v[i+1];  
                v[i+1] = aux;  
            }  
        }  
    }  
}
```



Bubble Sort

► E como chama a função?

```
int main() {  
  
    int v[10] = {33, 17, -11, 45, 1, 26, 54, 67, 21, 10}, i;  
    bolha(v, 10);  
    printf("Imprimindo o vetor ordenado");  
  
    for(i = 0; i < 10; i++){  
        printf(" %d", v[i]);  
    }  
    return 0;  
}
```



Bubble Sort

- ▶ O problema do método da bolha é:
- ▶ Ele é pouco eficiente se o vetor é muito grande.
- ▶ Realiza muitas trocas entre elementos.



Busca em Vetores

- ▶ Voltando ao problema da busca....
- ▶ Temos dois tipos de busca em vetores:
 - ▶ Busca Sequencial
 - ▶ Busca Binária



Busca em Vetores

► Busca Sequencial

- O algoritmo Busca Sequencial executa a pesquisa de um elemento em um vetor comparando-o aos elementos do vetor, um após o outro, até obter o resultado “verdadeiro” ou chegar ao fim da vetor.
- Este tipo de busca só é viável se o vetor for pequeno (ou médio) e/ou não estiver ordenado.
- Devido ao seu modo de operação, a mesma costuma consumir tempo.

Busca em Vetores

► Busca Sequencial

- O algoritmo Busca Sequencial executa a pesquisa de um elemento em um vetor comparando-o aos elementos do vetor, um após o outro, até obter o resultado “verdadeiro” ou chegar ao fim da vetor.
- Este tipo de busca só é viável se o vetor for pequeno (ou médio) e/ou não estiver ordenado.
- Devido ao seu modo de operação, a mesma costuma consumir tempo.

Busca em Vetores

► Busca Sequencial

```
// se achar retorna 1, se não retorna 0  
int busca_seq(int v[], int n, int num) {  
    int i;  
    for(i = 0; i < n; i++) {  
        if (v[i] == num) {  
            return 1;  
        }  
    }  
    return 0;  
}
```



Busca em Vetores

► Busca Sequencial

```
int main() {  
  
    int v[10] = {33, 17, -11, 45, 1, 26, 54, 67, 21, 10}, i;  
    bolha(v, 10);  
  
    int achou = busca_seq(v, 10, -1);  
    if(achou) {  
        printf("O numero -1 esta no vetor");  
    }  
    else {  
        printf("O numero -1 nao esta no vetor");  
    }  
  
    return 0;  
}
```



Busca em Vetores

► Busca Binária

- Uma alternativa à busca sequencial (que leva tempo e precisa verificar todas as posições do vetor), é a busca binária.
- Ela segue o paradigma de divisão e conquista. Ela parte do pressuposto de que o vetor está **ordenado** e realiza sucessivas divisões do espaço de busca comparando o elemento buscado com o elemento no meio do vetor.

Busca em Vetores

► Busca Binária

► Considerando o vetor inicialmente ordenado:

0	1	k	n-2	n-1
71	94	130	321	429	515	600	770	812

$$x = 320$$




► Verificamos qual a posição situada no meio do vetor - podemos chama-la de K

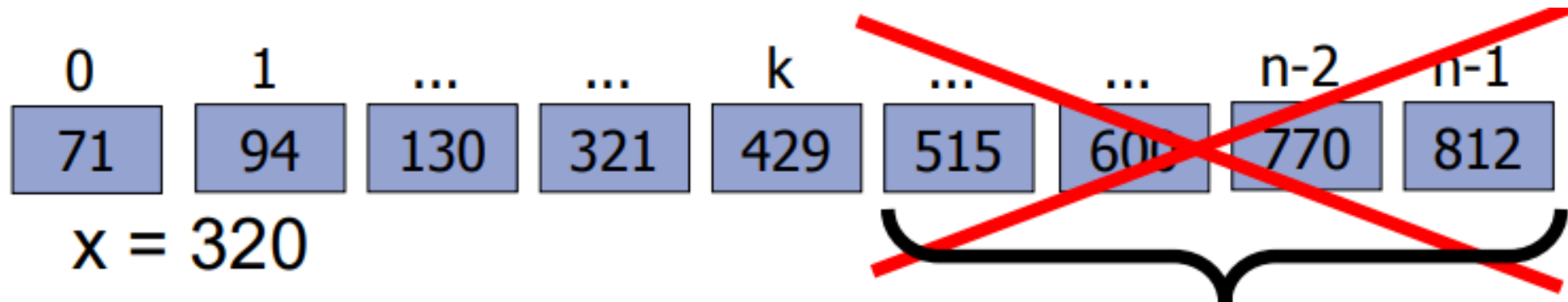
Busca em Vetores

► Busca Binária

► Então, comparamos X com o valor da posição K .

► Há 3 possíveis situações:

- $X = v[k]$  ACHOU O ELEMENTO
- $X < v[k]$  descartamos a metade direita do vetor
- $X > v[k]$  descartamos a metade esquerda do vetor



Busca em Vetores

► Busca Binária

- Repita o processo, reduzindo o vetor pela metade, até que:
 - x seja encontrado, ou
 - que o intervalo de busca termine



Busca em Vetores

► Busca Binária

- E como fica o algoritmo?
- Existem duas implementações: iterativa e recursiva.

```
//Implementação Iterativa:
int PesquisaBinaria (int vet[], int chave, int Tam)
{
    int inf = 0;      // limite inferior (o primeiro índice de vetor em C é zero )
    int sup = Tam-1; // limite superior (termina em um número a menos. 0 a 9 são 10 números)
    int meio;
    while (inf <= sup)
    {
        meio = (inf + sup)/2;
        if (chave == vet[meio])
            return meio;
        if (chave < vet[meio])
            sup = meio-1;
        else
            inf = meio+1;
    }
    return -1; // não encontrado
}
```



Busca em Vetores

► Busca Binária

- E como fica o algoritmo?
- Existem duas implementações: iterativa e recursiva.

```
//Implementação Recursiva:
// x => chave | v[] => vetor ordenado | e => limite inferior (esquerda) | d => limite superior (direita)
int PesquisaBinaria (int x, int v[], int e, int d)
{
    int meio = (e + d)/2;
    if (v[meio] == x)
        return meio;
    if (e >= d)
        return -1; // não encontrado
    else
        if (v[meio] < x)
            return PesquisaBinaria(x, v, meio+1, d);
        else
            return PesquisaBinaria(x, v, e, meio-1);
}
```



Busca em Vetores

► Busca Binária

- Mais eficiente que a busca sequencial pois realiza metade das verificações!!!
- Lembrando: a busca binária só pode ser utilizada se o vetor estiver **ordenado**.
- Se não, não faria sentido!

Referencias:

Adaptação da aula de recursividade e algoritmos de busca e ordenação da profa. Dra. Laura Rodríguez da Universidade da Madeira (http://cee.uma.pt/edu/eda/eda_200506/Aula05.pdf)

That's all Folks!



nemo