

Estruturas de Dados

Busca Linear e Binária

Busca em Vetor

- A operação de busca é encontrada com muita frequência em aplicações computacionais
- Exemplos:
 - Encontrar o salário de um dado funcionário
 - Encontrar o I.R.A. de um aluno
- Estratégias
 - Busca Linear em Vetor
 - Busca Linear em Vetor Ordenado
 - Busca Binária em Vetor Ordenado

Busca em Vetor

- **Entrada:** vetor **v** com **n** elementos e elemento **elem**
- **Saída:** **i** se o elemento ocorre em **v[i]**;
-1 se o elemento não ocorre em **v**.

Entrada:

4	2	5	1
---	---	---	---

+

=

Saída: 2

Elemento : 5

Entrada:

4	2	5	1
---	---	---	---

+

=

Saída: -1

Elemento : 3

Busca Linear

Percorra o vetor **v**, elemento a elemento verificando se **elem** é igual a um dos elementos de **v**

```
int busca(int n, int *v, int elem){
    int i;

    for(i=0; i<n; i++)
        if(elem==v[i])
            return i; // elemento encontrado

    return -1; // elemento não encontrado
}
```

5	4	2	5	1	i=0
5	4	2	5	1	i=1
5	4	2	5	1	i=2

3	4	2	5	1	i=0
3	4	2	5	1	i=1
3	4	2	5	1	i=2
3	4	2	5	1	i=3

-1

Complexidade

- **No melhor caso,**
 - apenas uma comparação
 - **$O(1)$**
- **No pior caso,**
 - n comparações, em que n é o número de elementos
 - **$O(n)$** , a complexidade varia linearmente em relação ao tamanho do problema.
- **No caso médio,**
 - $n / 2$ comparações
 - **$O(n)$** , a complexidade varia linearmente em relação ao tamanho do problema.

Busca pelo campo NOME

```
#include<stdio.h>
#include<string.h>
typedef struct aluno{
    char nome[81];
    float ira;
} Aluno;
#define MAX 10000
void inicializa(int n, Aluno **alunos);
void imprime(int n, Aluno **alunos, int i);
int buscaNome(int n, Aluno **alunos, char *nome);
void imprime_todos(int n, Aluno **alunos);
void atualiza(int n, Aluno **alunos, int i);
void exclui(int n, Aluno **alunos, int i);
int main (void){
    Aluno* alunos[MAX]; int i; int a;
    inicializa(MAX,alunos);
    for(i=0;i<MAX;i++)
        atualiza(MAX,alunos,i);
    a = buscaNome(MAX,alunos,"JOSE") ;
    imprime(MAX,alunos,a) ;
    for(i=0;i<MAX;i++)
        exclui(MAX,alunos,i);
    return 0; }
```

Busca pelo campo NOME

```
int buscaNOME(int n, Aluno **alunos, char *nome) {  
    int i;  
    for(i=0; i<n; i++)  
        if(strcmp(alunos[i]->nome,nome)==0)  
            return i; // elemento encontrado  
  
    return -1; // elemento não encontrado  
}
```

Busca em Vetor Ordenado

- **Entrada:** vetor ordenado **v** com **n** elementos e elemento **elem**
- **Saída:** **i** se o elemento ocorre em **v[i]**;
-1 se o elemento não ocorre em **v**.

Entrada:

1	2	4	5
---	---	---	---

+

=

Saída: 2

Elemento : 4

Entrada:

1	2	4	5
---	---	---	---

+

=

Saída: -1

Elemento : 3

Busca Linear - Vetor em Ordem Crescente

Percorra os elementos do vetor ordenado **v**,
que forem menores do que **elem**
verificando se **elem** é igual a um dos elementos de **v**

```
int busca(int n, int *v, int elem){  
    int i;
```

```
    for(i=0; i<n; i++)  
        if(elem==v[i])  
            return i;  
    else if (v[i] > elem)  
        return -1
```

```
    return -1;
```

```
}
```

3	1	2	4	5	i=0
3	1	2	4	5	i=1
3	1	2	4	5	i=2

-1

Quando o elemento não pertence ao vetor, a busca é ligeiramente
mais eficiente. Contudo, **a complexidade continua sendo linear $O(n)$**

Busca Binária

- **Entrada:** vetor ordenado **v** com **n** elementos e elemento **elem**
- **Saída:** **i** se o elemento ocorre em **v[i]**;
-1 se o elemento não ocorre em **v**.
- **Busca:**
 - Compare **elem** com o elemento do meio de **v**
 - Se **elem for igual**, retorne a posição
 - Se **elem for menor**, pesquise na primeira metade de **v**
 - Se **elem for maior**, pesquise na segunda metade de **v**
 - Continue o processo até encontrar o elemento ou chegar a uma parte de tamanho 0

Busca Binária

```
int buscaBinaria(int n, int *v, int elem){
    int ini = 0, fim = n-1, meio;
    while(fim-ini>=0){
        meio = (ini+fim)/2;
        if(elem == v[meio])
            return meio;
        else if (elem > v[meio])
            ini = meio+1;
        else if (elem < v[meio])
            fim = meio-1;
    }
    return -1 //elemento não encontrado
}
```

92

12	25	35	37	57	63	86	92	100
----	----	----	----	----	----	----	----	-----

 ini=0; meio=4; fim=8

92

					63	86	92	100
--	--	--	--	--	----	----	----	-----

 ini=5; meio=6; fim=8

92

							92	100
--	--	--	--	--	--	--	----	-----

 ini=7; meio=7; fim=8

Exemplo Busca Binária

```
int buscaBinaria(int n, int *v, int elem){
    int ini = 0, fim = n-1, meio;
    while(fim-ini>=0){
        meio = (ini+fim)/2;
        if(elem == v[meio])
            return meio;
        else if (elem > v[meio])
            ini = meio+1;
        else if (elem < v[meio])
            fim = meio-1;
    }
    return -1 //elemento não encontrado
}
```

24	12	25	35	37	57	63	86	92	100
24	12	25	35	37					
24	12								
-1 24									

ini=0; meio=4; fim=8

ini=0; meio=1; fim=3

ini=0; meio=0; fim=0

ini=1; meio=0; fim=0

Complexidade – Busca Binária

- **No melhor caso,**
 - apenas uma comparação
 - **$O(1)$**
- **No pior caso,**
 - 2 comparações a cada ciclo, em que n é o número de elementos
 - A cada repetição, a parte considerada na busca é dividida à metade, ou seja, **$O(\log n)$**

Repetição	Tamanho do Problema
1	n
2	$n / 2$
3	$n / 4$
...	...
$\log n$	1

Busca Binária pelo campo NOME

```
#include<stdio.h>
typedef struct aluno{
    char nome[81];
    float ira;
} Aluno;
#define MAX 10000
void inicializa(int n, Aluno **alunos);
void imprime(int n, Aluno **alunos, int i);
int buscaBinariaNome(int n, Aluno **alunos, char *nome);
void imprime_todos(int n, Aluno **alunos);
void atualiza(int n, Aluno **alunos, int i);
void exclui(int n, Aluno **alunos, int i);
int main (void){
    Aluno* alunos[MAX]; int i; int a;
    inicializa(MAX,alunos);
    for(i=0;i<MAX;i++)
        atualiza(MAX,alunos,i);
    a = buscaBinariaNome(MAX,alunos,"JOSE") ;
    imprime(MAX,alunos,a) ;
    for(i=0;i<MAX;i++)
        exclui(MAX,alunos,i);
    return 0; }
```

Busca pelo campo NOME

```
int buscaBinariaNOME(int n, Aluno **alunos, char *nome)
{
    int ini = 0, fim = n-1, meio, r;
    while(fim-ini>=0){
        meio = (ini+fim)/2;
        r = strcmp(nome, alunos[meio]->nome);
        if(r == 0)
            return meio;
        else if (r > 0)
            ini = meio+1;
        else if (r < 0)
            fim = meio-1;
    }
    return -1; //elemento não encontrado
}
```

Referência

Introdução a Estruturas de Dados, Waldemar Celes, Renato Cerqueira e José Lucas Rangel, Editora Campus, 2004.