Técnicas de Programação e Análise de Algoritmos

Prof. Dr. Lucas Rodrigues Costa

Aula 9: Busca

lucas.costa@idp.edu.br

@lucasrodri

www.linkedin.com/in/lucas-rodri

OBJETIVOS

- Compreender o conceito de busca e manipulação de estruturas
- Conhecer algoritmos de busca linear em em vetores e estruturas lineares
- Conhecer algoritmos de busca binária em em vetores e estruturas lineares

idp

RECORDANDO...

- Vimos na aula passada
  - O conceito de notação assintótica
  - Os diferentes tipos de análise assintótica
  - Classe de problemas

Algoritmos de Busca

Definição

- Ato de procurar por um elemento em um conjunto de dados
  - Recuperação de dados armazenados em um repositório ou base de dados
- A operação de busca visa responder se um determinado valor está ou não presente em um conjunto de elementos
  - Por exemplo, em um array ou em uma lista

Definição

- Baseada em uma chave
  - A chave de busca é o **campo** do item utilizado para comparação
    - Valor armazenado em um array de inteiros
    - Campo de uma struct
    - etc…
  - É por meio dela que sabemos se dado elemento é o que buscamos
    - No caso do item estar presente no conjunto de elementos, seus dados são retornados para o usuário

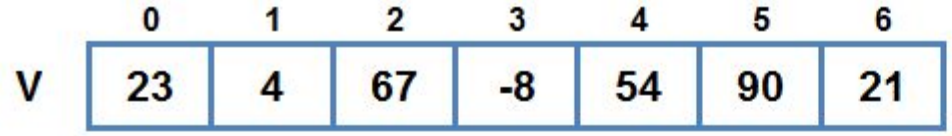
Definição

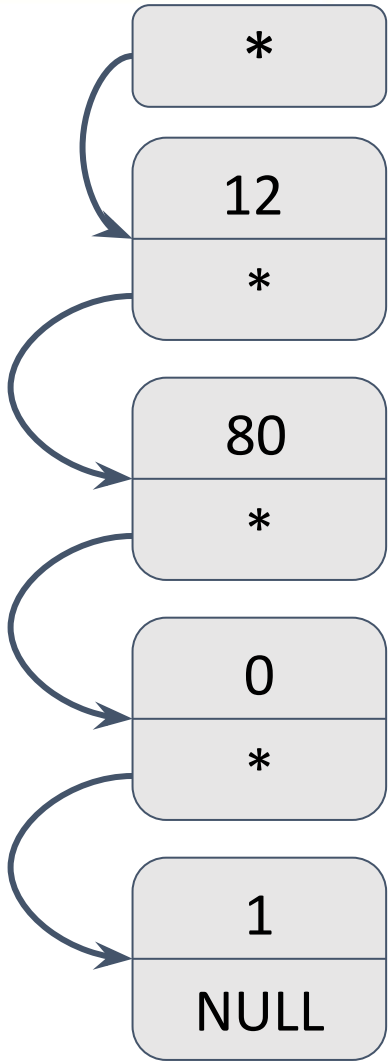
- Existem vários tipos de busca
- Sua utilização depende de como são estes dados
  - Os dados estão estruturados ?
    - Array, lista, árvore, etc.?
    - Existe também a busca em dados não estruturados
  - Os dados estão ordenados?
  - Existem valores duplicados?

idp

Definição

- Tipos de busca abordados
  - Dados armazenados em um array, lista
  - Dados ordenados ou não
- Métodos
  - Busca Sequencial ou Linear
  - Busca Sequencial Ordenada
  - Busca Binária





Busca Sequencial ou Linear

Busca Sequencial ou Linear

- Estratégia de busca mais simples que existe
  - Basicamente, esse algoritmo percorre o array que contém os dados desde a sua primeira posição até a última
    - Assume que os dados não estão ordenados, por isso a necessidade de percorrer o array do seu início até o seu fim

Busca Sequencial ou Linear

- Funcionamento
  - Para cada posição do array ou do elemento, o algoritmo compara se a posição atual do array, ou valor do elemento é igual ao valor buscado.
    - Se os valores forem iguais, a busca termina
    - caso contrário, a busca continua com a próxima posição do array ou elemento

Busca Sequencial ou Linear

- Algoritmo (array)

```
int buscaLinear(int *V, int N, int elem){
    int i;
    for(i = 0; i<N; i++){
        if(elem == V[i])
            return i;//elemento encontrado
    }
    return -1;//elemento não encontrado
}
```

Busca Sequencial ou Linear

- Algoritmo (estrutura)

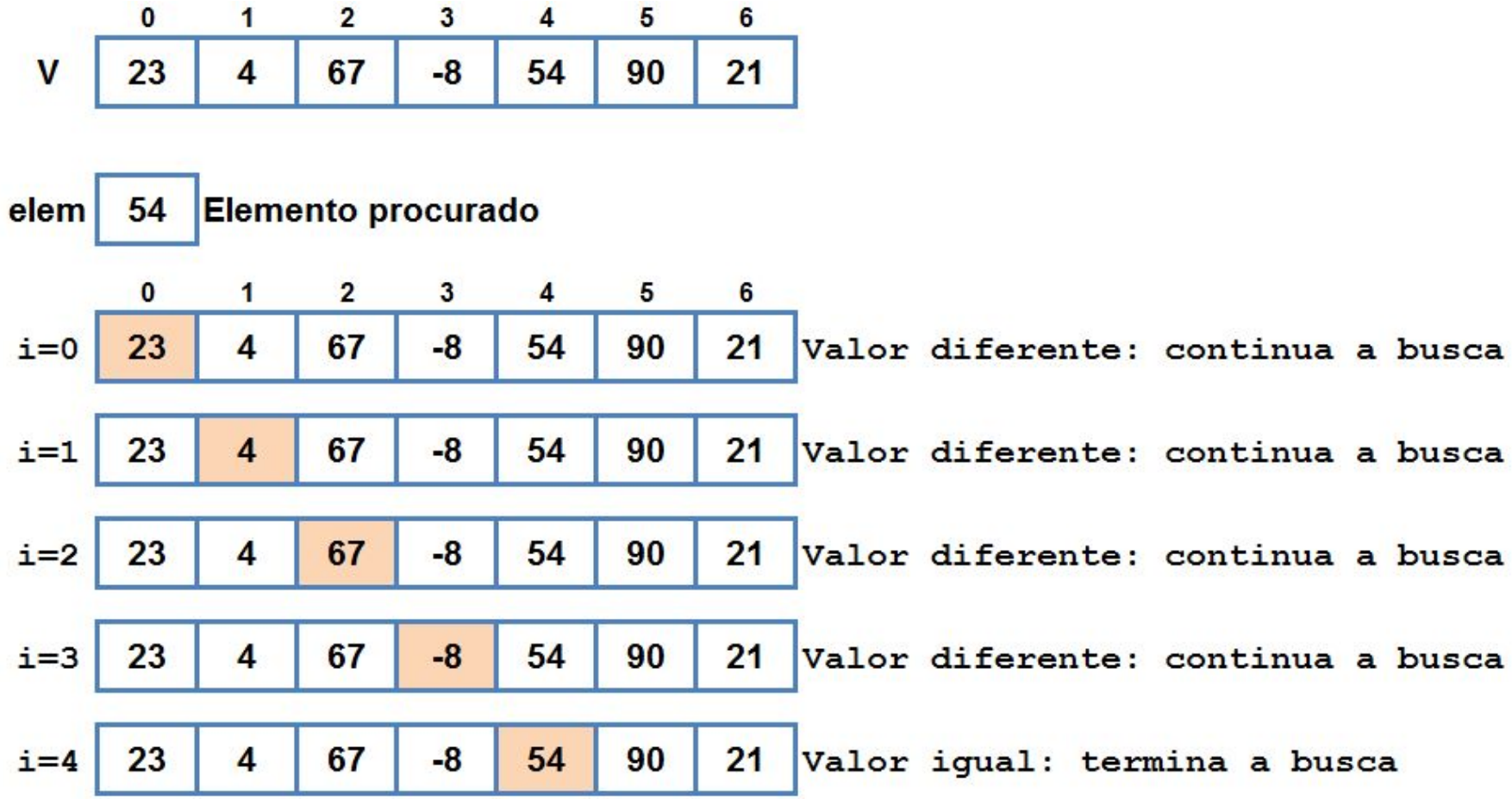
```c
typedef struct aluno {
    int matricula;
    char nome[50];
    float media;
    struct aluno* prox;
} Aluno;

Aluno* buscaLinear(Aluno* lista, int matricula) {
    Aluno* atual = lista;
    while (atual != NULL && atual->matricula != matricula) {
        atual = atual->prox;
    }
    return atual;
}
```

idp

Busca Sequencial ou Linear

- Exemplo (array)



Busca Sequencial ou Linear

- Complexidade
  - Considerando um array com **N** elementos
    - **O(1)**, melhor caso: o elemento é o primeiro do array
    - **O(N)**, pior caso: o elemento é o último do array ou não existe
    - **O(N/2)**, caso médio

Busca Sequencial Ordenada

Busca Sequencial Ordenada

- Procurar por um determinado valor em um array ou estrutura desordenada é uma tarefa bastante cara.

- Como melhorar isso?
  - Organizando o array/estrutura segundo alguma ordem, isto é, devemos ordenar o array/estrutura
  - Isso facilita a tarefa de busca

Busca Sequencial Ordenada

- Funcionamento em array
  - Assume que os dados estão ordenados
  - Se o elemento procurado for **menor** do que o valor em uma determinada posição do array, temos a certeza de que ele não estará no restante do array
    - Isso evita a necessidade de percorrer o array do seu início até o seu fim

Busca Sequencial Ordenada

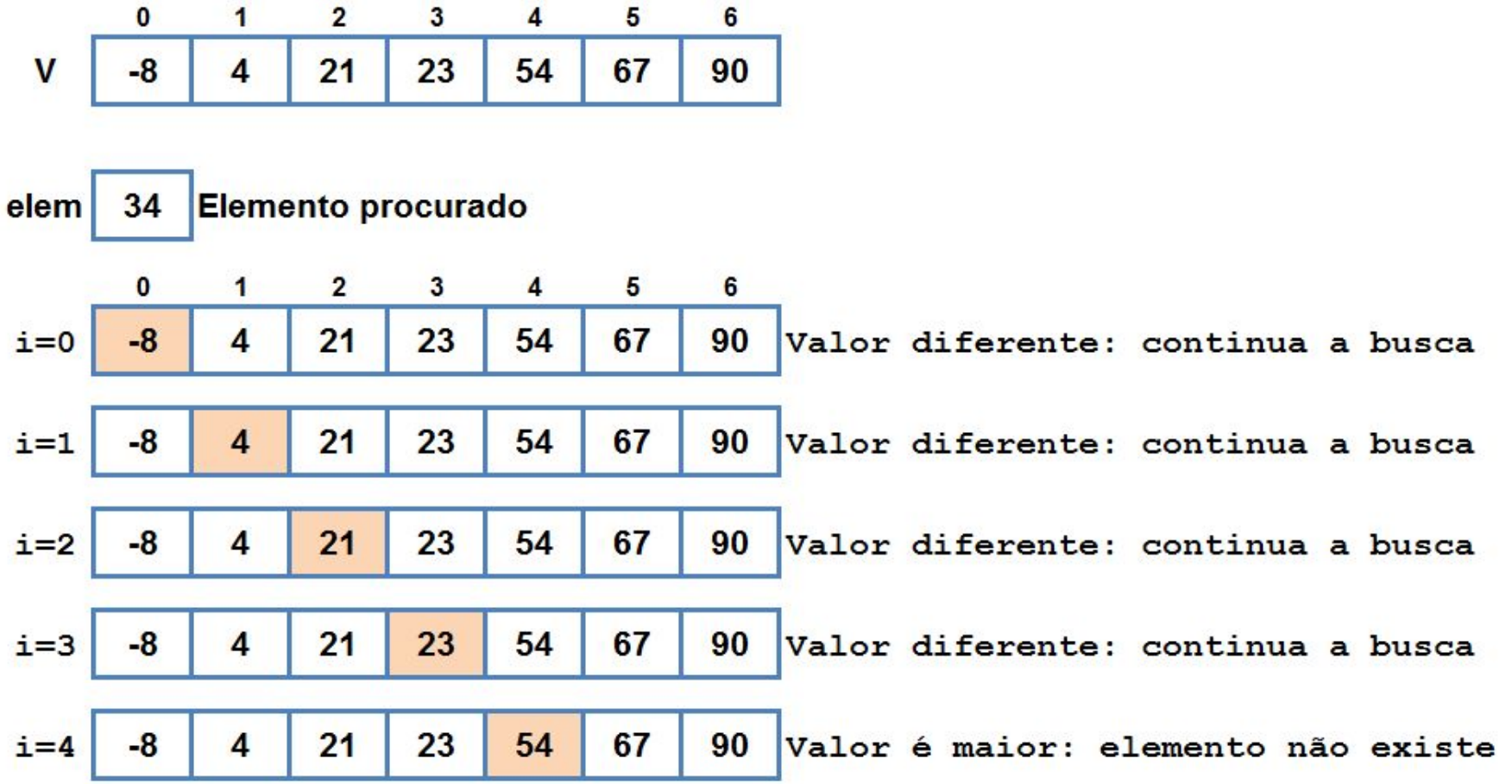
- Algoritmo (array)

```
int buscaOrdenada(int *V, int N, int elem){
    int i;
    for(i = 0; i<N; i++){
        if(elem == V[i])
            return i;//elemento encontrado
        else
            if(elem < V[i])
                return -1;//para a busca
    }
    return -1;//elemento não encontrado
}
```

idp

Busca Sequencial Ordenada

- Exemplo (array)



Busca Sequencial Ordenada

- Desvantagens
  - Ordenar um array também tem um custo
    - Esse custo é superior ao custo da busca sequencial no seu pior caso
  - Se for para fazer a busca de um único elemento, não compensa ordenar o array
    - Porém, se mais de um elemento for recuperado do array, o esforço de ordenar o array pode compensar

Busca Sequencial Ordenada

- E o funcionamento em uma lista encadeada?
  - Não temos vantagens?

Busca Binária

Busca Binária

- Fazer a busca em um array ordenado representa um ganho de tempo
  - Podemos terminar a busca mais cedo se o elemento procurado for menor que o valor da posição atual do array

Busca Binária

- A Busca Sequencial Ordenada é uma estratégia de busca extremamente simples
  - Ela percorre todo o array linearmente
  - Não utiliza adequadamente a ordenação dos dados
- Uma estratégia de busca mais sofisticada é a Busca Binária
  - Muito mais eficiente do que a Busca Sequencial Ordenada

Busca Binária

- Funcionamento
  - É uma estratégia baseada na idéia de dividir para conquistar
  - A cada passo, esse algoritmo analisa o valor do meio do array
    - Caso esse valor seja igual ao elemento procurado, a busca termina
    - Caso contrário, a busca continua na metade do array que condiz com o valor procurado

Busca Binária

- Algoritmo

```
int buscaBinaria(int *V, int N, int elem){
    int i, inicio, meio, final;
    inicio = 0;
    final = N-1;
    while(inicio <= final){
        meio = (inicio + final)/2;
        if(elem < V[meio])
            final = meio-1;//busca na metade da esquerda
        else
            if(elem > V[meio])
                inicio = meio+1;//busca na metade da direita
            else
                return meio;
    }
    return -1;//elemento não encontrado
}
```

Busca Binária

- Exemplo

Busca Binária

- Complexidade
  - Considerando um array com **N** elementos, o tempo de execução é:
    - **O(1)**, melhor caso: o elemento procurado está no meio do array;
    - **O($\log_2$ N)**, pior caso: o elemento não existe;
    - **O($\log_2$ N)**, caso médio.

Busca Binária

- Complexidade
  - Para se ter uma idéia da sua vantagem, em um array contendo **N = 1000** elementos, no pior caso
    - A Busca Sequencial irá executar **1000 comparações**
    - A Busca Binária irá executar apenas **10 comparações**

Busca em array de struct

Busca em array de struct

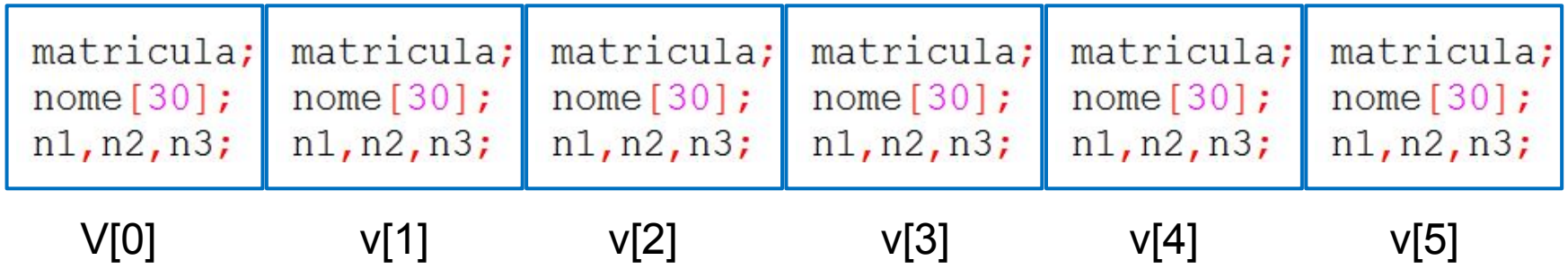
- A busca em um array de inteiros é uma tarefa simples
  - Na prática, trabalhamos com dados um pouco mais complexos, como estruturas
  - Mais dados para manipular

```
struct aluno{
    int matricula;
    char nome[30];
    float n1,n2,n3;
};
```

Busca em array de struct

➔ Como fazer a busca quando o que temos é um array de struct?

```
struct aluno V[6];
```



Busca em array de struct

- A busca é baseada em uma chave
  - A chave de busca é o **campo** do item utilizado para comparação
    - Valor armazenado em um array de inteiros
    - **Campo de uma struct**
    - etc
  - É por meio dela que sabemos se dado elemento é o que buscamos
    - No caso do item estar presente no conjunto de elementos, seus dados são retornados para o usuário

Busca em array de struct

- Ou seja, devemos modificar o algoritmo para que a comparação das chaves seja feita utilizando um determinado campo da struct
- Exemplo
  - Vamos modificar a busca linear
    - Essa modificação vale para os outros tipos de busca

```
int buscaLinear(int *V, int N, int elem){
    int i;
    for(i = 0; i<N; i++){
        if(elem == V[i])
            return i;//elemento encontrado
    }
    return -1;//elemento não encontrado
}
```

Busca em array de struct

- Duas novas buscas
  - Busca por **matricula**
  - Busca por **nome**

```c
int buscaLinearMatricula(struct aluno *V, int N, int elem){
    int i;
    for(i = 0; i<N; i++){
        if(elem == V[i].matricula)
            return i;//elemento encontrado
    }
    return -1;//elemento não encontrado
}


int buscaLinearNome(struct aluno *V, int N, char* elem){
    int i;
    for(i = 0; i<N; i++){
        if(strcmp(elem,V[i].nome)==0)
            return i;//elemento encontrado
    }
    return -1;//elemento não encontrado
}
```

idp

Perguntas?

Exercícios de Fixação

Exercícios

1. Forneça um exemplo de aplicação real que envolva o problema de ordenação e de encontrar o menor valor em uma lista encadeada.
2. Escreva um algoritmo que procure por um dado número em vetor e em uma fila duplamente encadeada.

idp

Referências

- BACKES, A. Ricardo. Algoritmos e estruturas de dados em linguagem C. 1. ed. Rio de Janeiro: LTC, 2023.
- Prof. Dr. André Backes; Estrutura de Dados 2; 2012

INSTITUTO BRASILEIRO DE ENSINO,
DESENVOLVIMENTO E PESQUISA

lucas.costa@idp.edu.br

@lucasrodri

www.linkedin.com/in/lucas-rodri