

ESTRUTURA DE DADOS

Curso de Licenciatura em Ciências da Computação

Lucas Sampaio Leite



**INSTITUTO
FEDERAL**

Baiano

Professor

Lucas Sampaio Leite

lucas.leite@ifbaiano.edu.br



Objetivo Geral

- Compreender e aplicar os principais conceitos, estruturas e operações relacionadas às listas lineares, filas, pilhas, árvores binárias e suas variações, utilizando uma linguagem de programação para implementar, analisar e solucionar problemas computacionais.

Objetivo Específicos

- Ao final da disciplina, o estudante deverá ser capaz de:
 - Identificar as características, aplicações e diferenças entre listas lineares, filas, pilhas e árvores binárias.
 - Implementar listas, filas, pilhas e árvores binárias (e suas variações) em uma linguagem de programação.
 - Aplicar operações fundamentais sobre essas estruturas (inserção, remoção, busca, travessias, etc.).
 - Resolver problemas utilizando as estruturas estudadas, selecionando a estrutura adequada para cada contexto.
 - Analisar vantagens, limitações e complexidade das estruturas de dados abordadas.
 - Desenvolver soluções computacionais que integrem mais de uma estrutura de dados em cenários reais.

Ementa

- Listas lineares e suas variações.
- Filas e pilhas.
- Árvores binárias e suas variações.
- Utilização de uma linguagem de programação.

Conteúdo Programático

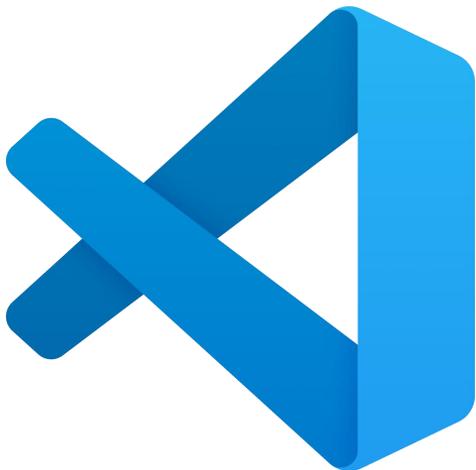
1. Introdução às Estruturas de Dados
2. Revisão de ponteiros e alocação dinâmica de memória
3. Listas Lineares: Conceitos Gerais
4. Listas Encadeadas Simples
5. Listas Encadeadas Duplas e Listas Circulares
6. Filas (Queue)
7. Pilhas (Stack)
8. Árvores: Estrutura e Conceitos Iniciais

Conteúdo Programático

9. Árvores Binárias
10. Travessias em Árvores Binárias
11. Árvores Binárias de Busca (BST)
12. Árvores Balanceadas
13. Árvores AVL

Ferramentas e ambientes de desenvolvimento

- Exemplos em Pseudocódigo, C/C++ e/ou Java
- IDEs: VS Code, Dev-C++, Codeblocks, etc - (livre)
- Implementações: C/C++, sugestões???
- Árbitro Virtual Beecrowd (listas de exercícios)



Metodologia

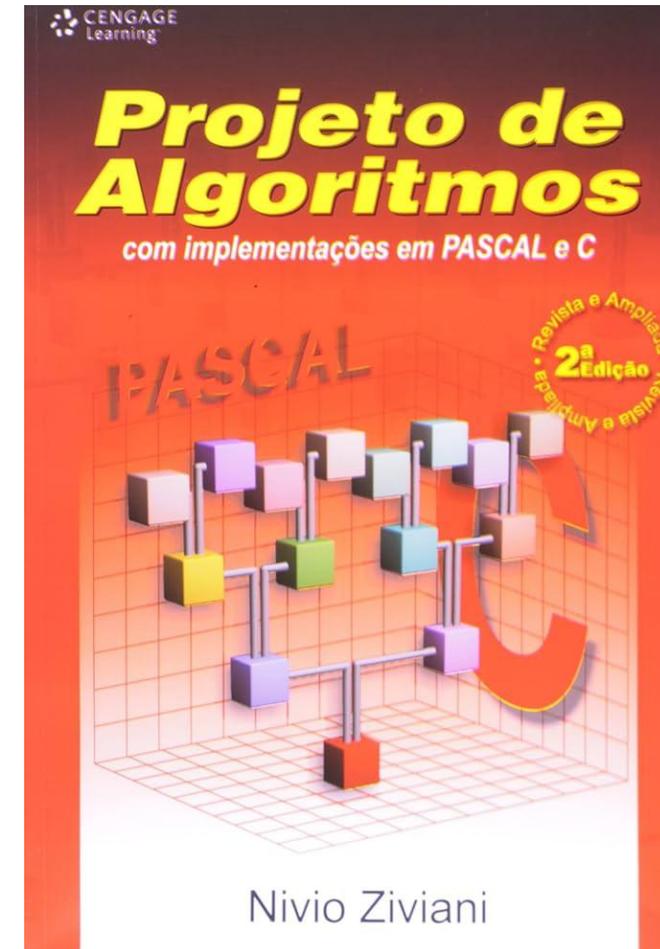
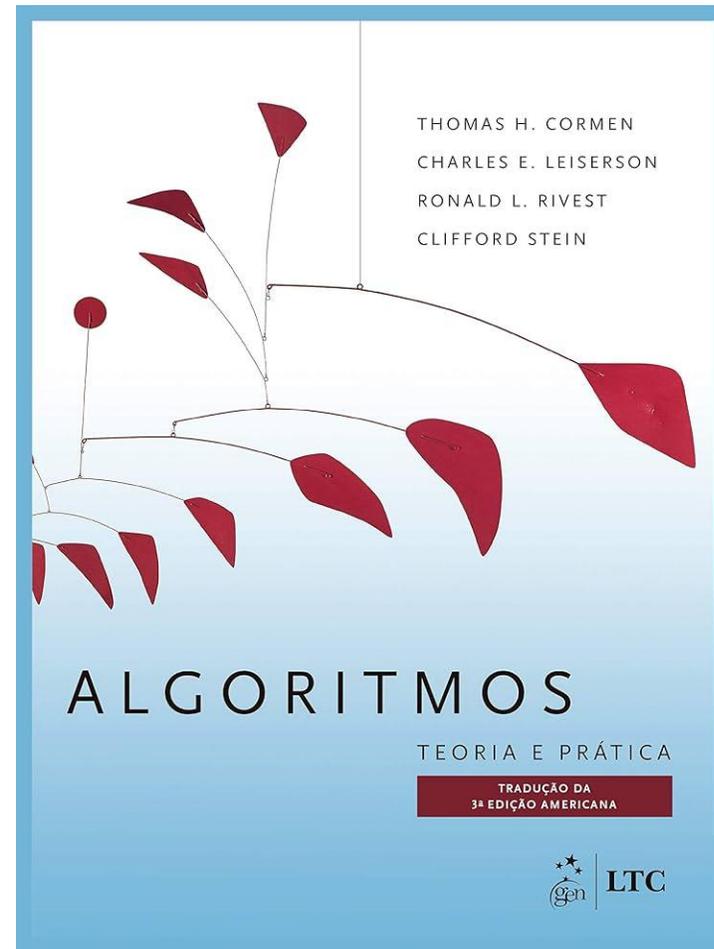
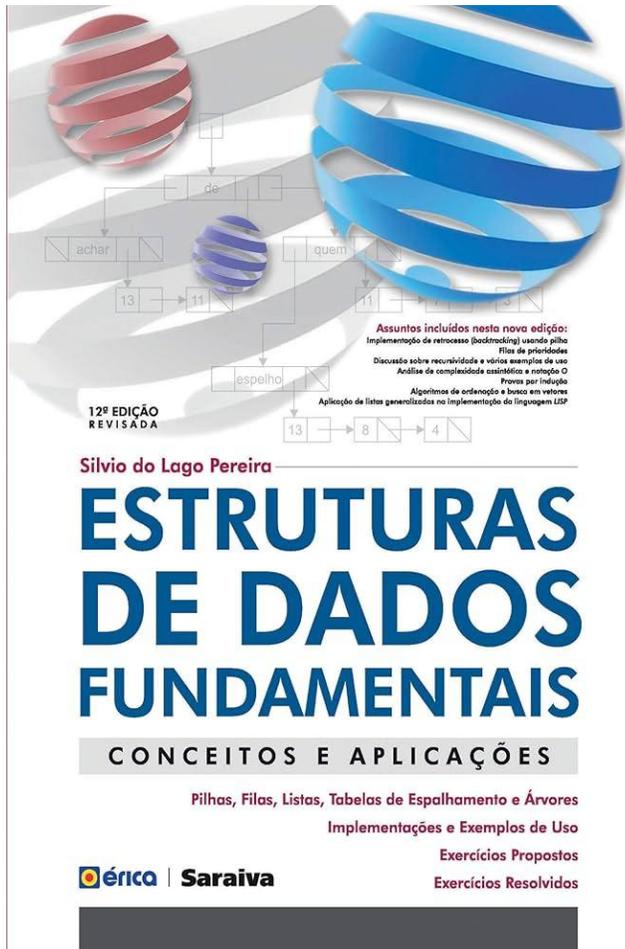
- Aulas síncronas e presenciais
 - Aulas expositivas e dialogadas
 - Abordagem prática
 - Aulas de resolução de exercícios
- **Com chamada**

Avaliação

- Avaliação 1: Listas de exercício 30% + Prova 70%
- Avaliação 2: Listas de exercício 30% + Prova 70%
- Prova final



Bibliografia Básica



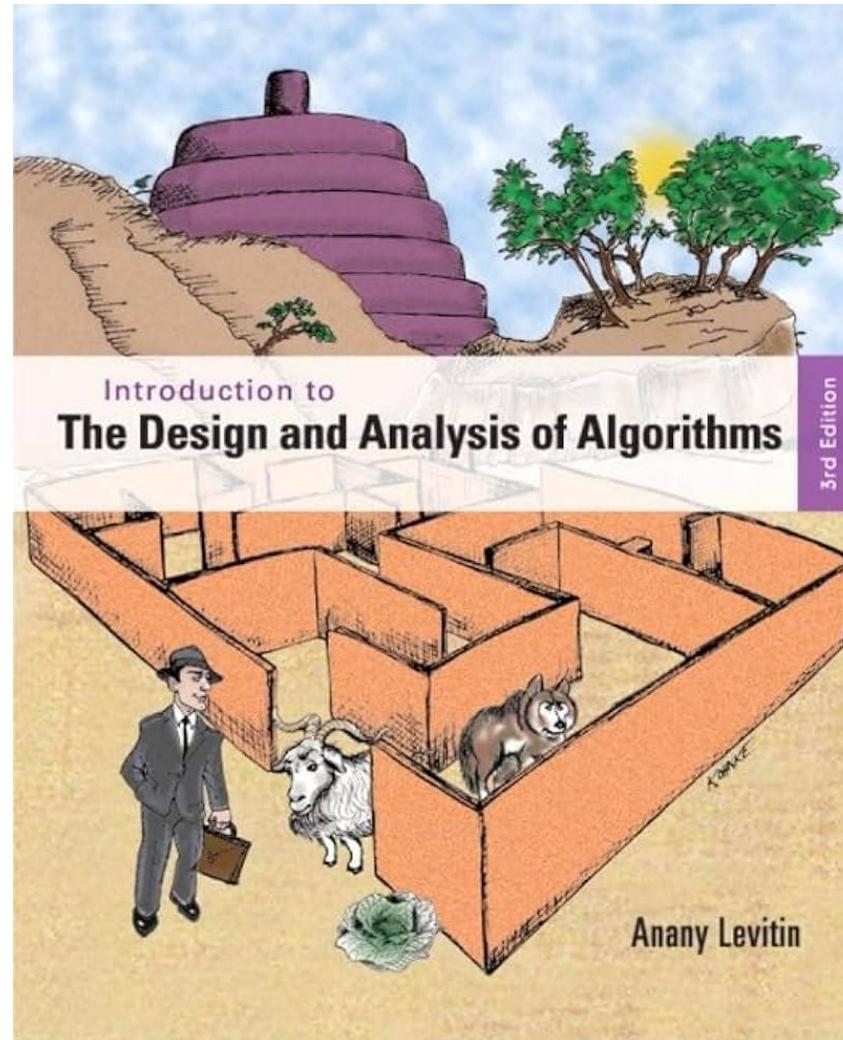
Bibliografia Básica

- PEREIRA, Silvio do Lago. Estruturas de dados fundamentais: conceitos e aplicações . 12. ed.,. São Paulo: Érica, 2012. 264 p. ISBN 9788571943704.
- CORMEN, Thomas H. Algoritmos: teoria e prática. Rio de Janeiro: Campus, 2012. ISBN: 9788535236996.
- ZIVIANI, Nivio. Projeto de algoritmos com implementações em Pascal e C. São Paulo: Pioneira Thomson Learning. ISBN: 8522105251

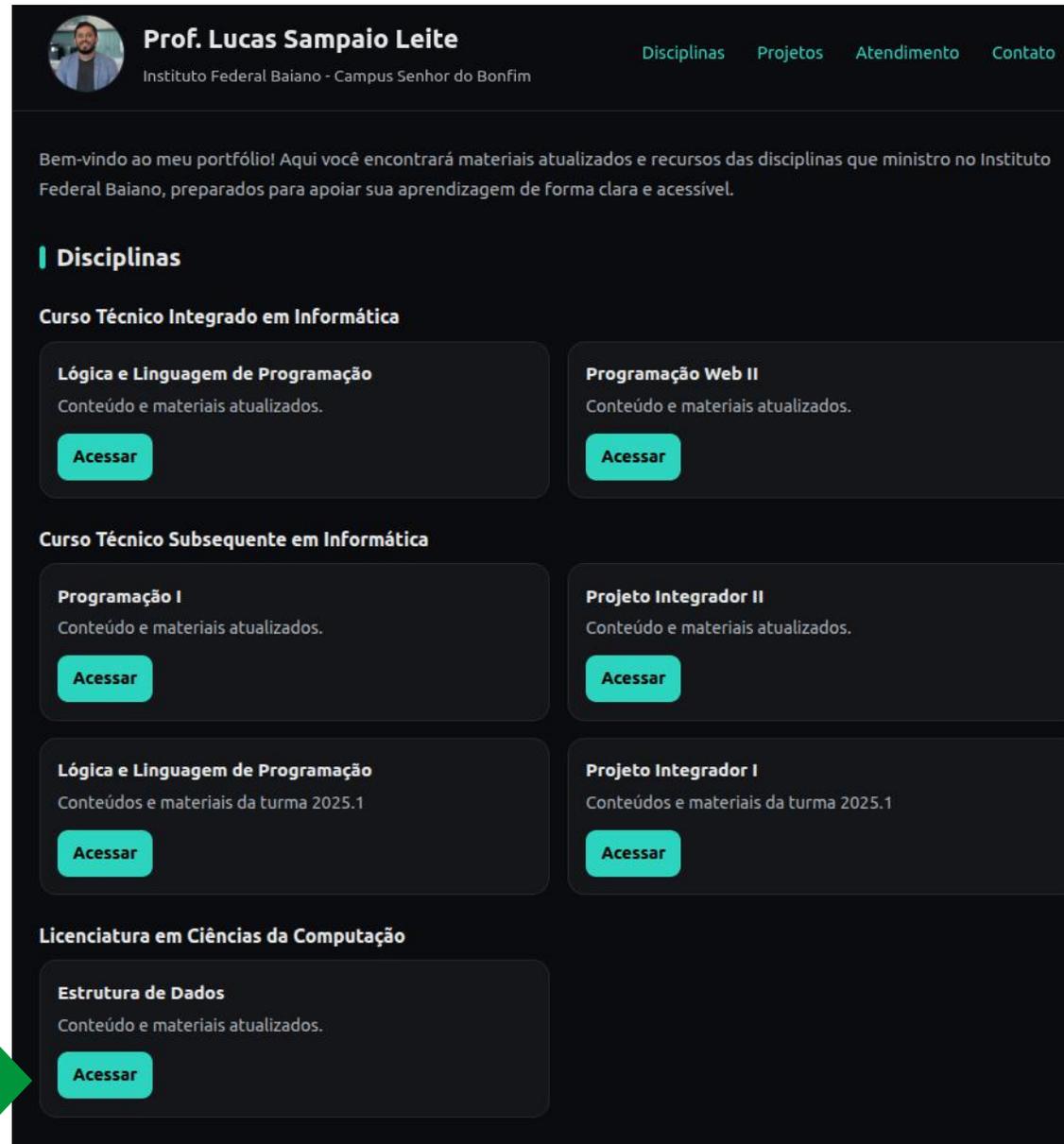
Bibliografia Complementar

- ASCENCIO, Ana Fernanda Gomes; CAMPOS, Edilene Aparecida Veneruchi de. Fundamentos da programação de computadores: algoritmos, Pascal e C/C++ e Java. 3. ed. São Paulo: Pearson Prentice Hall, 2012. ISBN: 9788564574168.
- DEITEL, Harvey M.; DEITEL, Paul J. C ++ como programar. 5. ed. Porto Alegre: Pearson, 2006. ISBN: 9788576050568.
- TENENBAUM, Aaron. Estruturas de dados usando C. São Paulo: Makron, 1995. ISBN: 8534603480.

Outras Referências



Material de aula e informações sobre a disciplina



Prof. Lucas Sampaio Leite
Instituto Federal Baiano - Campus Senhor do Bonfim

Disciplinas Projetos Atendimento Contato

Bem-vindo ao meu portfólio! Aqui você encontrará materiais atualizados e recursos das disciplinas que ministro no Instituto Federal Baiano, preparados para apoiar sua aprendizagem de forma clara e acessível.

Disciplinas

Curso Técnico Integrado em Informática

- Lógica e Linguagem de Programação**
Conteúdo e materiais atualizados.
[Acessar](#)
- Programação Web II**
Conteúdo e materiais atualizados.
[Acessar](#)

Curso Técnico Subsequente em Informática

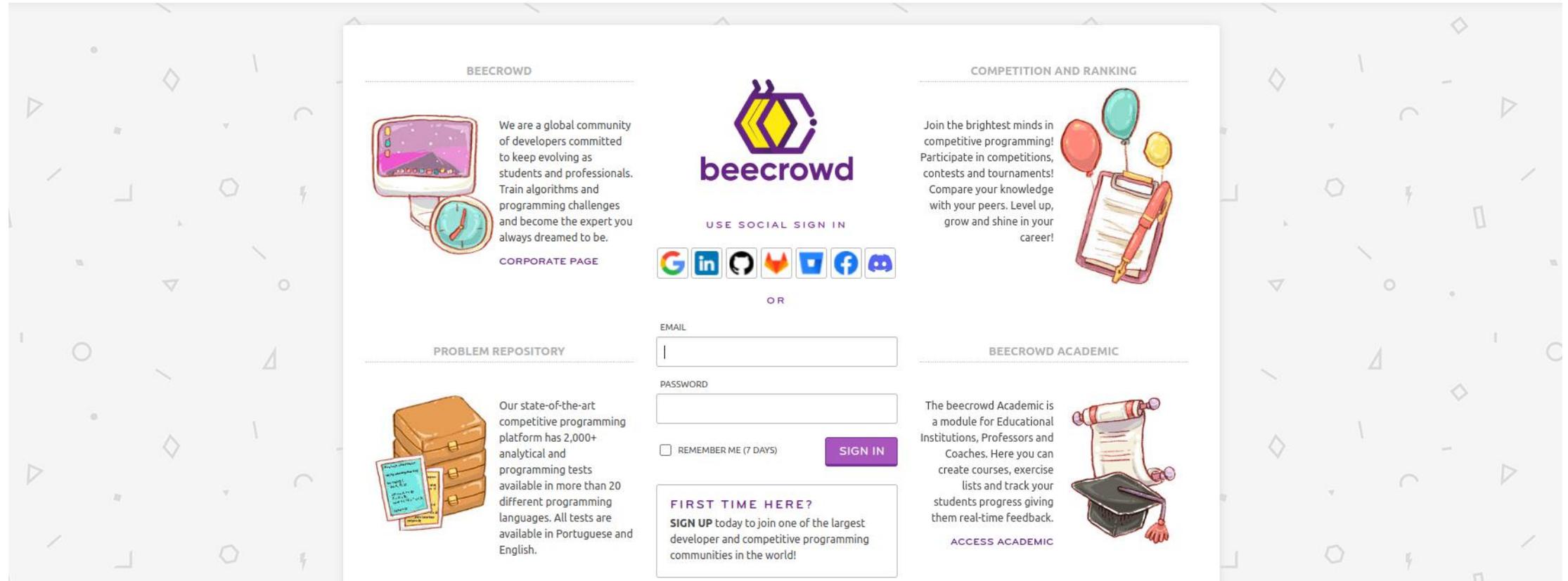
- Programação I**
Conteúdo e materiais atualizados.
[Acessar](#)
- Projeto Integrador II**
Conteúdo e materiais atualizados.
[Acessar](#)
- Lógica e Linguagem de Programação**
Conteúdos e materiais da turma 2025.1
[Acessar](#)
- Projeto Integrador I**
Conteúdos e materiais da turma 2025.1
[Acessar](#)

Licenciatura em Ciências da Computação

- Estrutura de Dados**
Conteúdo e materiais atualizados.
[Acessar](#)



Referência de exercícios Python (com árbitro digital)



BEECROWD

We are a global community of developers committed to keep evolving as students and professionals. Train algorithms and programming challenges and become the expert you always dreamed to be.

[CORPORATE PAGE](#)

PROBLEM REPOSITORY

Our state-of-the-art competitive programming platform has 2,000+ analytical and programming tests available in more than 20 different programming languages. All tests are available in Portuguese and English.

COMPETITION AND RANKING

Join the brightest minds in competitive programming! Participate in competitions, contests and tournaments! Compare your knowledge with your peers. Level up, grow and shine in your career!

BEECROWD ACADEMIC

The beecrowd Academic is a module for Educational Institutions, Professors and Coaches. Here you can create courses, exercise lists and track your students progress giving them real-time feedback.

[ACCESS ACADEMIC](#)

USE SOCIAL SIGN IN

[G](#) [in](#) [GitHub](#) [Python](#) [Twitter](#) [Facebook](#) [Discord](#)

OR

EMAIL

PASSWORD

REMEMBER ME (7 DAYS) [SIGN IN](#)

FIRST TIME HERE?

SIGN UP today to join one of the largest developer and competitive programming communities in the world!

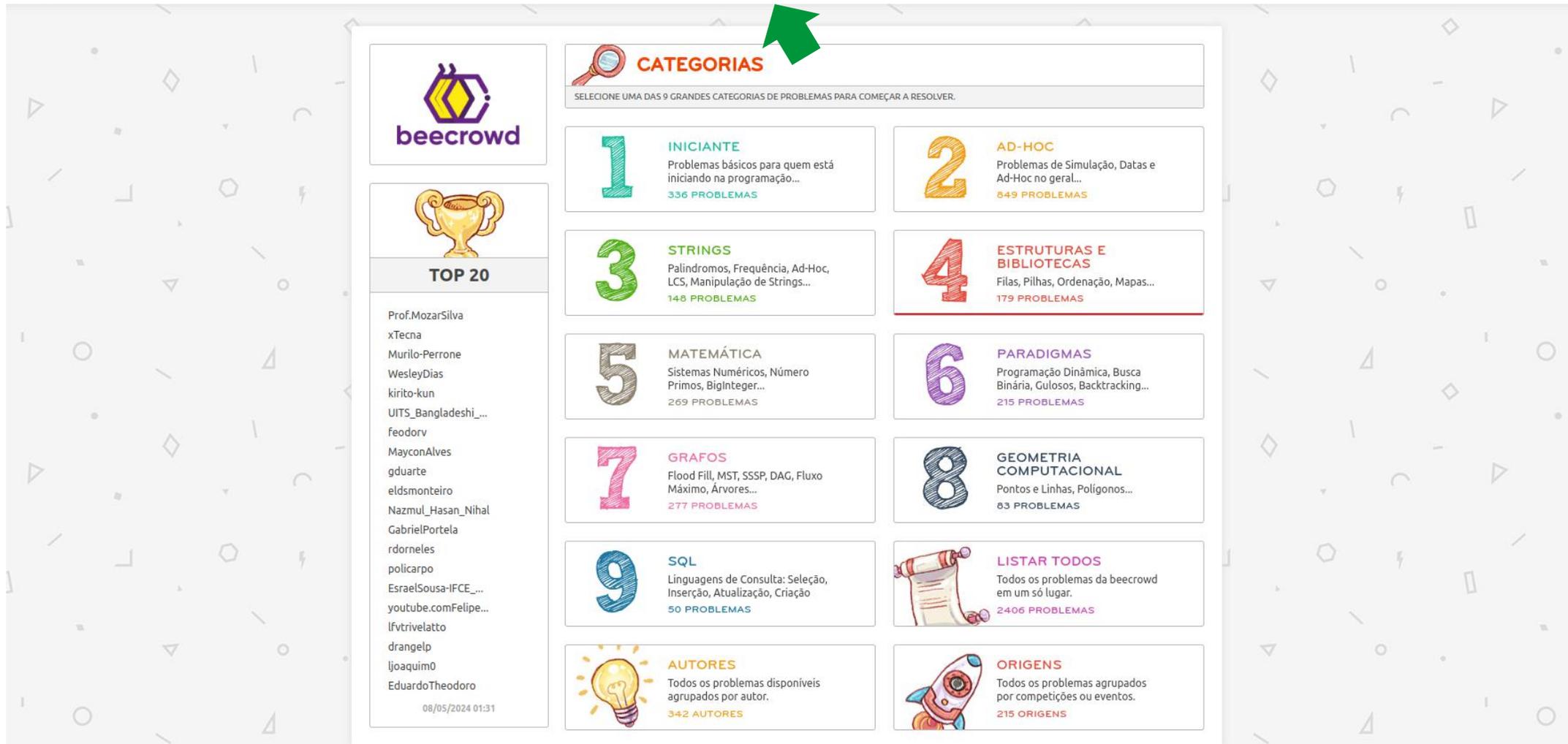
<https://judge.beecrowd.com/>

Acesso da turma para listas de exercícios

Hi, lucas.sampaio.leite
lucas.sampaio.leite@gmail.com

HOME PERFIL NEWS OPORTUNIDADES ACADEMIC CONTESTS PROBLEMAS SUBMISSÕES RANKS SAIR

beecrowd



beecrowd

TOP 20

- Prof.MozarSilva
- xTecn
- Murilo-Perrone
- WesleyDias
- kirito-kun
- UITS_Bangladeshi_...
- feodorv
- MayconAlves
- gduarte
- eldsmonteiro
- Nazmul_Hasan_Nihal
- GabrielPortela
- rdorneles
- policarpo
- EsraelSousa-IFCE_...
- youtube.comFelipe...
- lfvtrivelatto
- drangelp
- ljoaquim0
- EduardoTheodoro

08/05/2024 01:31

CATEGORIAS

SELECIONE UMA DAS 9 GRANDES CATEGORIAS DE PROBLEMAS PARA COMEÇAR A RESOLVER.

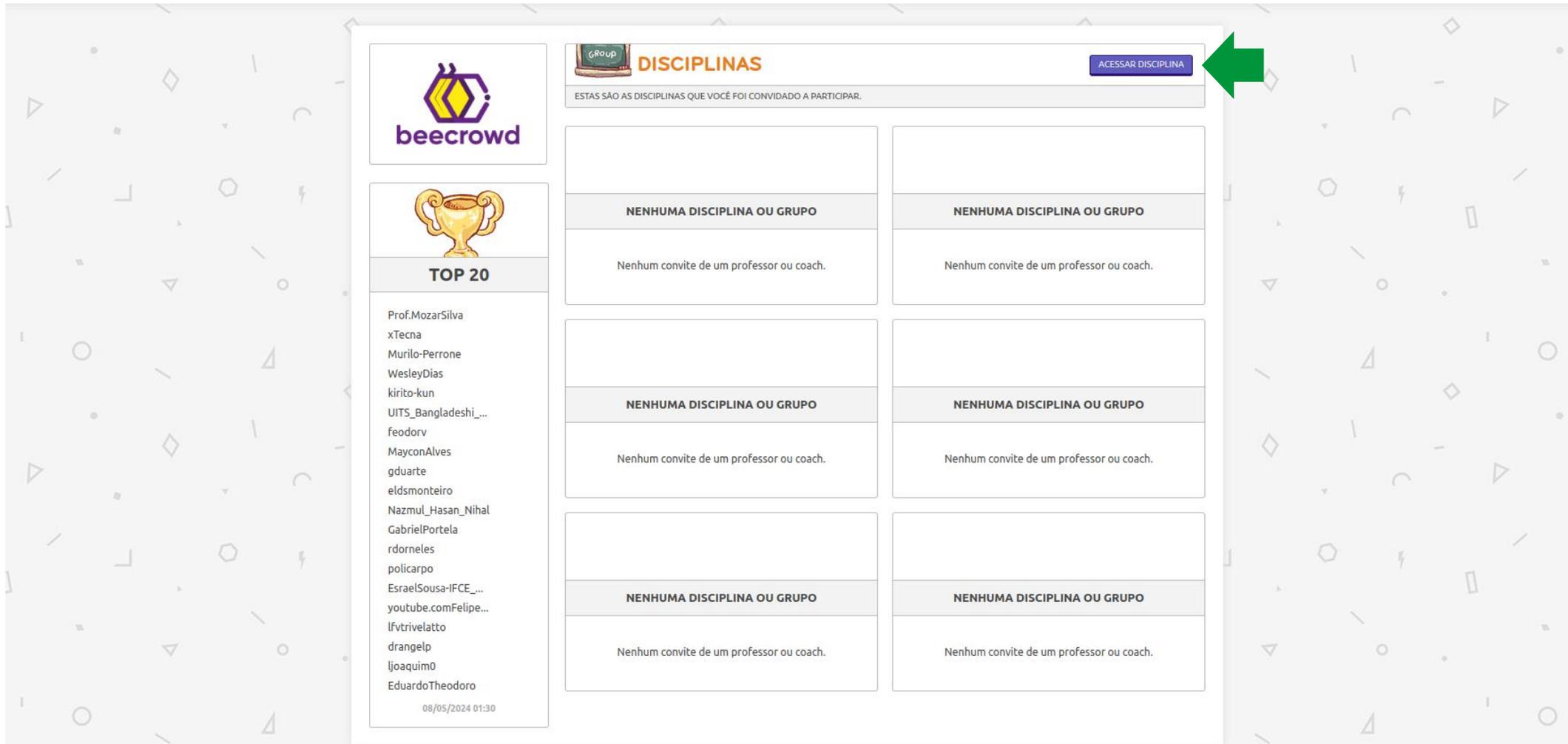
- 1 INICIANTE**
Problemas básicos para quem está iniciando na programação...
336 PROBLEMAS
- 2 AD-HOC**
Problemas de Simulação, Datas e Ad-Hoc no geral...
849 PROBLEMAS
- 3 STRINGS**
Palindromos, Frequência, Ad-Hoc, LCS, Manipulação de Strings...
148 PROBLEMAS
- 4 ESTRUTURAS E BIBLIOTECAS**
Filas, Pilhas, Ordenação, Mapas...
179 PROBLEMAS
- 5 MATEMÁTICA**
Sistemas Numéricos, Número Primos, BigInteger...
269 PROBLEMAS
- 6 PARADIGMAS**
Programação Dinâmica, Busca Binária, Gulosos, Backtracking...
215 PROBLEMAS
- 7 GRAFOS**
Flood Fill, MST, SSSP, DAG, Fluxo Máximo, Árvores...
277 PROBLEMAS
- 8 GEOMETRIA COMPUTACIONAL**
Pontos e Linhas, Polígonos...
83 PROBLEMAS
- 9 SQL**
Linguagens de Consulta: Seleção, Inserção, Atualização, Criação
50 PROBLEMAS

LISTAR TODOS
Todos os problemas da beecrowd em um só lugar.
2406 PROBLEMAS

AUTORES
Todos os problemas disponíveis agrupados por autor.
342 AUTORES

ORIGENS
Todos os problemas agrupados por competições ou eventos.
215 ORIGENS

Acesso da turma para listas de exercícios



The screenshot shows the Beecrowd interface. On the left, there is a sidebar with the Beecrowd logo, a trophy icon, and a 'TOP 20' list of users. The main content area is titled 'DISCIPLINAS' and contains a grid of discipline cards. Each card has a header 'NENHUMA DISCIPLINA OU GRUPO' and a body with the text 'Nenhum convite de um professor ou coach.' A green arrow points to a blue button labeled 'ACESSAR DISCIPLINA' in the top right corner of the main content area.

Acesso da turma para listas de exercícios



ACESSAR

INSIRA A CHAVE PARA ACESSAR A DISCIPLINA

INFORMAÇÃO DA DISCIPLINA

Por favor insira o ID e a chave fornecida para acessar a disciplina.

ID DISCIPLINA

CHAVE

ENTRAR 

Acesso da turma para listas de exercícios



 **DISCIPLINA**

ESTRUTURA DE DADOS 

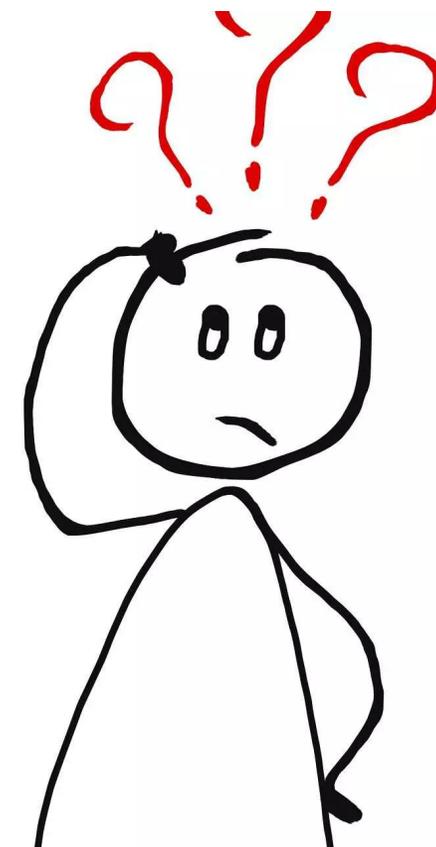
#	HOMEWORK	INICIAR	DATA LIMITE
Não há nenhum homework disponível neste momento.			



Boas práticas

- Para a dinâmica de aprendizagem da disciplina funcionar, é importante realizar as práticas e exercícios passados em sala (não deixem acumular atividades).
- Quem deixa acumular as atividades, tende a ter um desempenho inferior.
- Não deixem as listas de exercício para última hora.
- Organizem seu tempo entre as disciplinas.

O que é um Algoritmo?

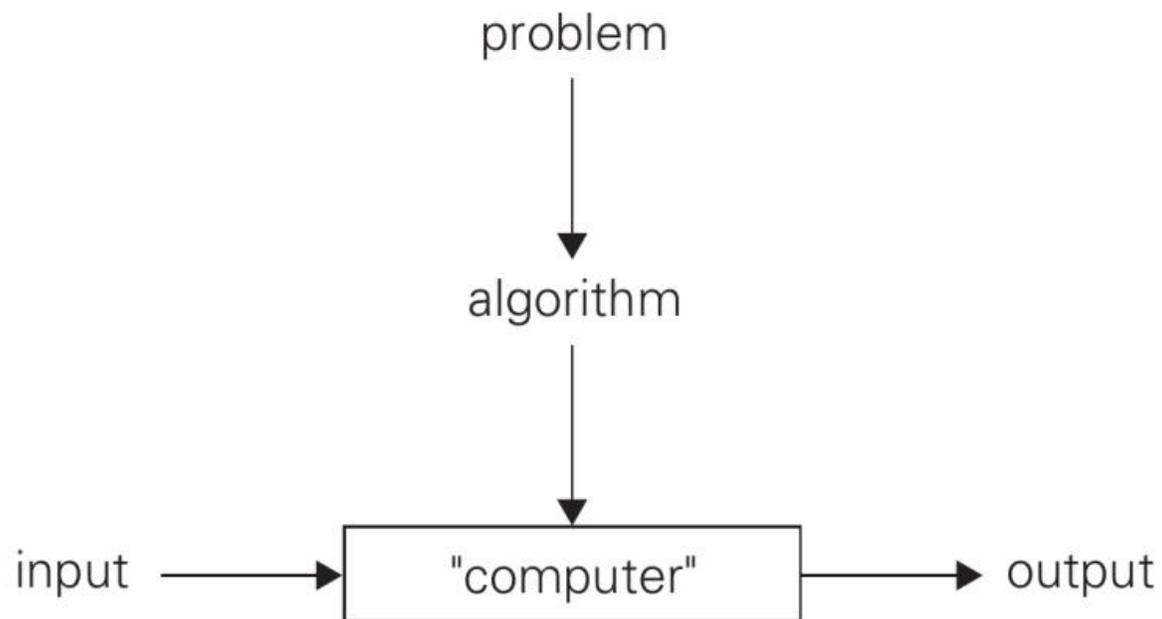


Definição informal (Cormen et al.)

- Informalmente, um algoritmo é qualquer procedimento computacional bem definido que toma algum valor ou conjunto de valores como entrada e produz algum valor ou conjunto de valores com saída.
- Portanto, um algoritmo é uma sequência de etapas computacionais que transformam a entrada em saída.

Definição (Levitin)

- Um algoritmo é uma sequência de instruções inequívocas para resolver um problema, ou seja, para obter uma saída necessária para qualquer entrada legítima em um período de tempo finito.



Algumas ponderações...

- O requisito de não ambiguidade não pode ser comprometido.
- O intervalo de entradas válidas deve ser especificado cuidadosamente.
- O mesmo algoritmo pode ser representado de diversas maneiras diferentes.
 - Podem existir vários algoritmos para resolver o mesmo problema.
- Algoritmos podem resolver o mesmo problema com velocidades muito diferentes.

Exemplo (problema da ordenação)

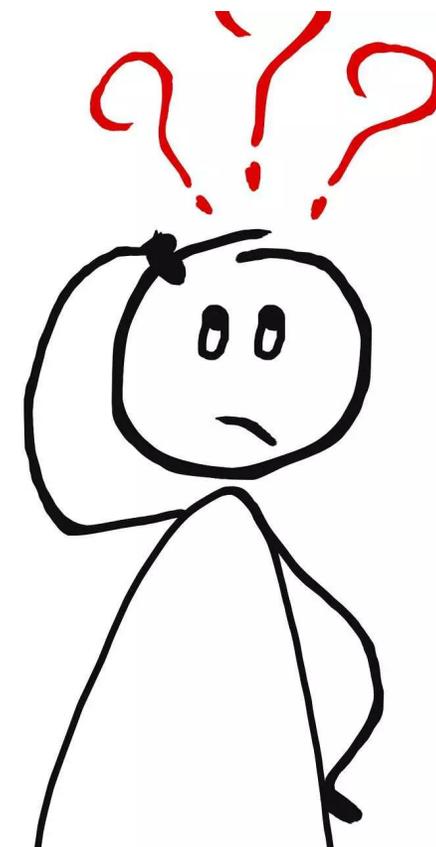
- Entrada: Uma sequência de números inteiros $\langle a_1, a_2, \dots, a_n \rangle$
- Saída: Uma permutação (reordenação) $\langle a_1', a_2', \dots, a_n' \rangle$. da sequência de entrada, tal que $\langle a_1' \leq a_2' \leq \dots \leq a_n' \rangle$

- Sequência de entrada (instância): $\langle 31, 41, 59, 26, 41, 58 \rangle$
- Saída esperada: $\langle 26, 31, 41, 41, 48, 49 \rangle$

Exemplo (problema da ordenação)

- Em geral, uma **instância** de um problema consiste na entrada (que satisfaz quaisquer restrições impostas no enunciado do problema) necessária para calcular uma solução para o problema.
- Um algoritmo é **correto** se, para toda instância de entrada ele parar com a saída correta.
- Dizemos que um algoritmo correto **resolve** o problema computacional dado.

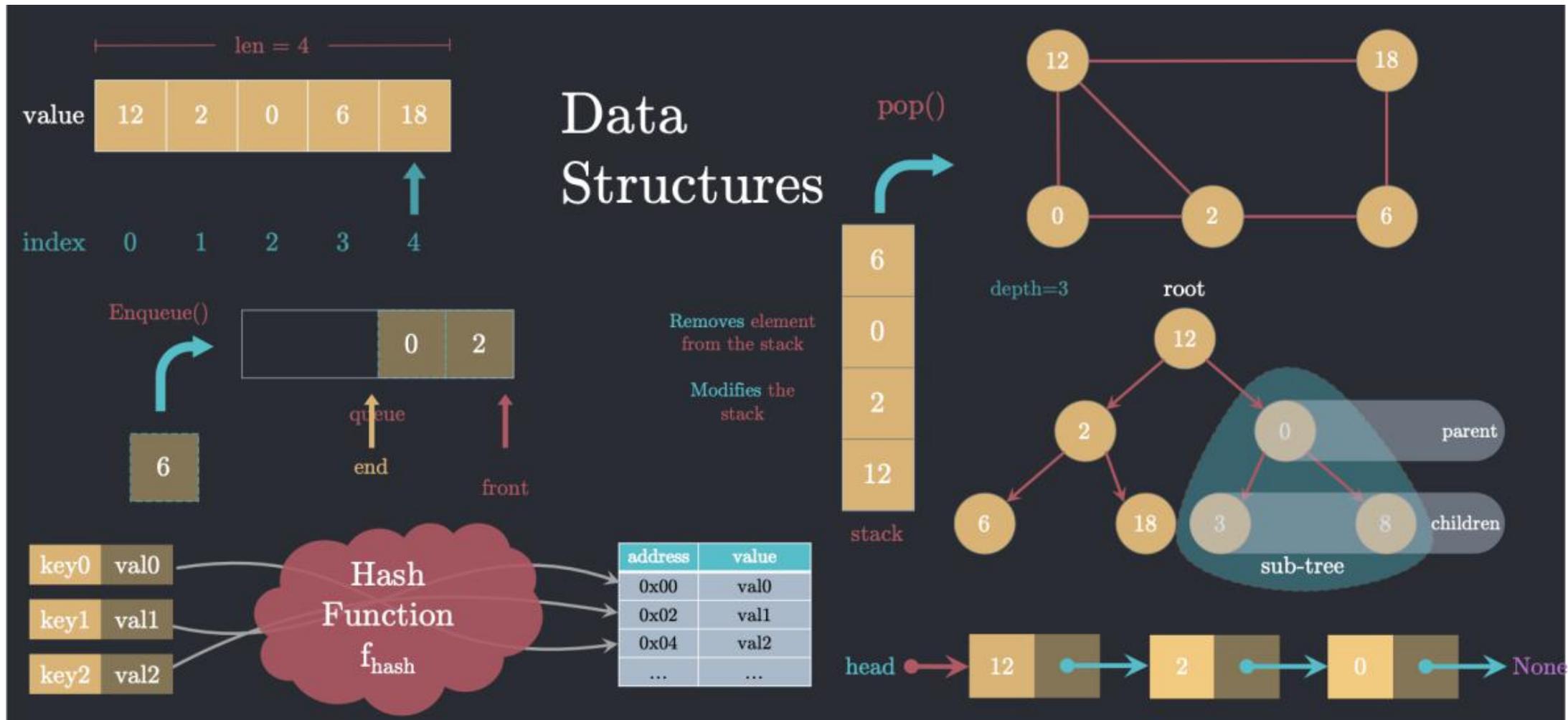
O que são estruturas de dados?



Definição informal (Cormen et al.)

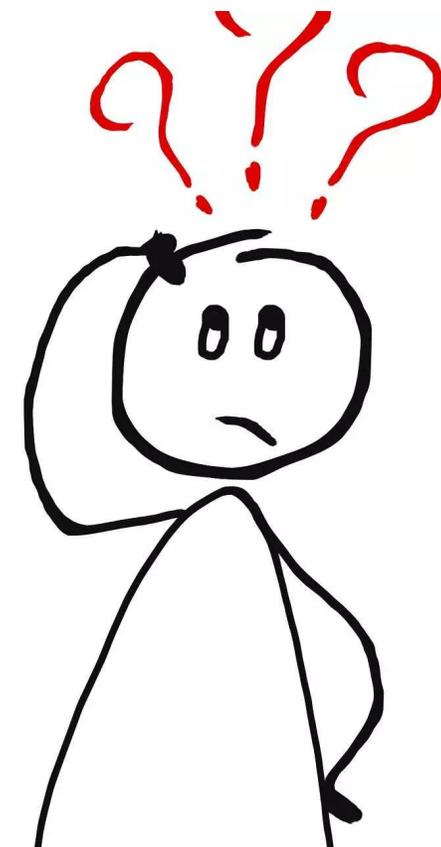
- Uma estrutura de dados é um modo de armazenar e organizar dados com o objetivo de facilitar acesso e modificações.
- Nenhuma estrutura de dados única funciona bem para todas as finalidades e, por isso, é importante conhecer os pontos fortes e limitações de várias delas.

Definição informal (Cormen et al.)



Fonte imagem: <https://brasap.com.br/8-estruturas-de-dados-que-todo-o-programador-precisa-conhecer/>

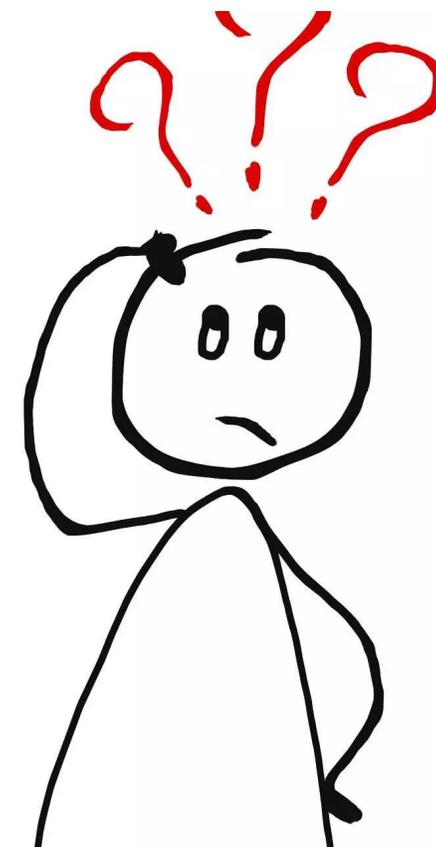
Qual a relação entre algoritmos e estruturas de dados?



Qual a relação entre algoritmos e estruturas de dados?

- Um algoritmo pode ser otimizado ou ineficiente dependendo da estrutura de dados utilizada para armazenar e manipular os dados envolvidos no problema.
- Exemplo:
 - Buscar um elemento em uma lista não ordenada.
 - Buscar um elemento em uma árvore de busca (BST) aplicando a propriedade de busca binária.

O que são vetores e matrizes?



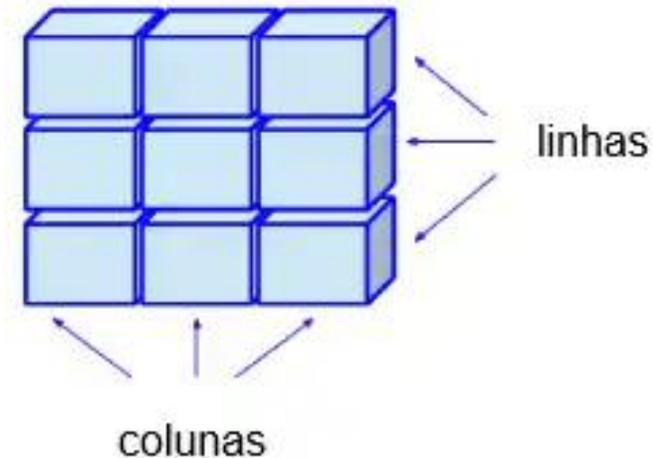
Vetores e Matrizes

- Um **vetor** (ou array unidimensional) é uma estrutura de dados que armazena uma **sequência de elementos do mesmo tipo**, organizados em uma **única dimensão** e acessados por índices.
- Uma **matriz** é uma extensão dos vetores: é um **array multidimensional**, podendo ter duas dimensões (linhas e colunas) ou mais, como matrizes tridimensionais e n-dimensionais.

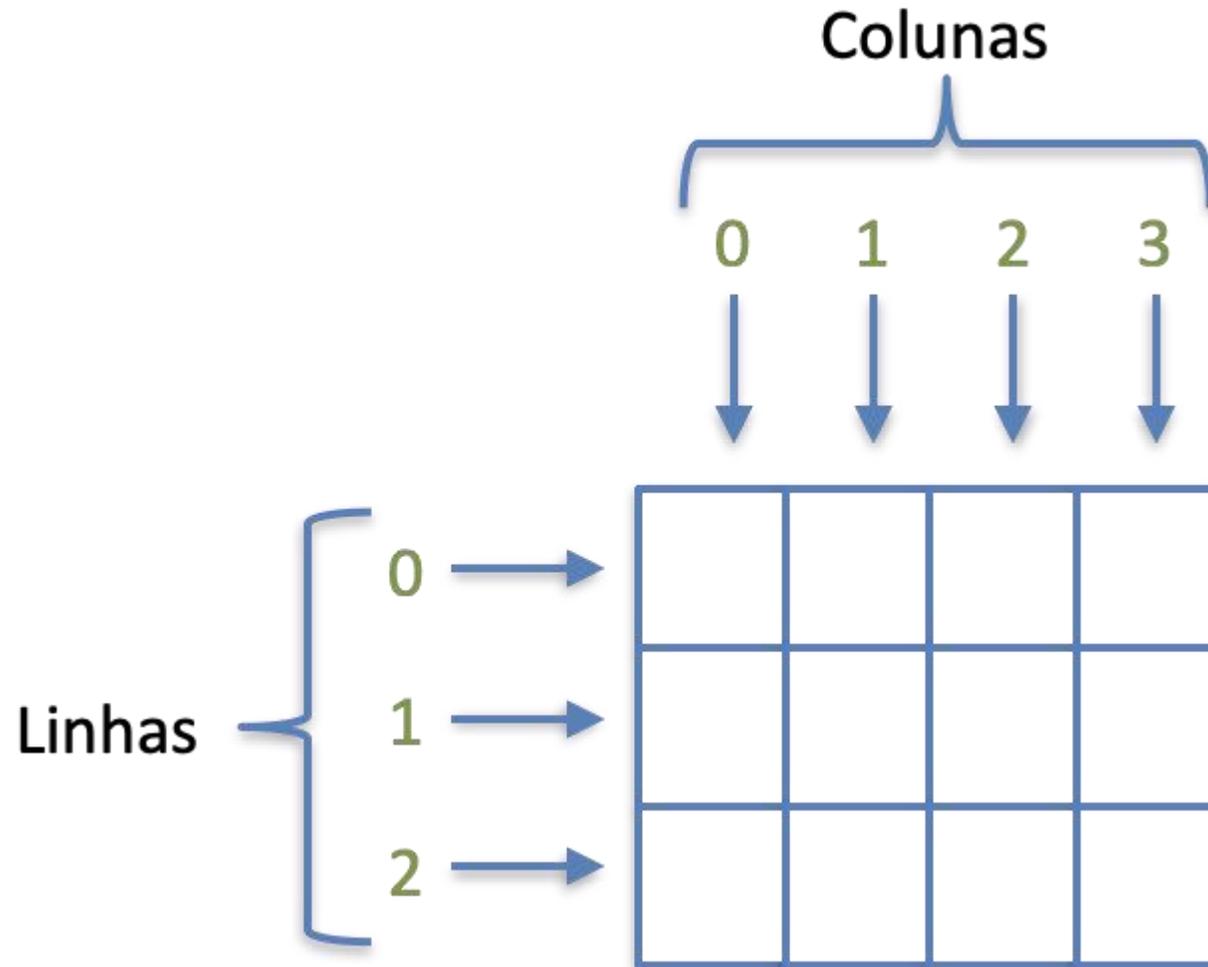
Vetor



Matriz



Vetores e Matrizes



Vetores

- Um vetor é uma estrutura de dados linear que armazena uma coleção ordenada de elementos do mesmo tipo.
- Cada elemento é identificado por um índice, permitindo acesso direto e eficiente.
- O tamanho do vetor é fixo, sendo definido no momento de sua declaração.
- Para acessar um elemento, utiliza-se seu índice correspondente, que geralmente começa em 0 e vai até tamanho - 1.

Vetores

- Declaração e inicialização de um vetor de inteiros com 5 elementos

```
vetor: array[0..4] of integer
```

- Inicialização dos elementos do vetor:

```
vetor[0] := 10
```

```
vetor[1] := 20
```

```
vetor[2] := 30
```

```
vetor[3] := 40
```

```
vetor[4] := 50
```

Vetores

- Declaração e inicialização de um vetor de inteiros com 5 elementos

```
vetor: array[0..4] of integer
```

- Inicialização dos elementos do vetor:

```
vetor[0] := 10
```

```
vetor[1] := 20
```

```
vetor[2] := 30
```

```
vetor[3] := 40
```

```
vetor[4] := 50
```



```
#include <stdio.h>

int main() {
    int vetor[5];

    vetor[0] = 10;
    vetor[1] = 20;
    vetor[2] = 30;
    vetor[3] = 40;
    vetor[4] = 50;

    return 0;
}
```

Exercício rápido

- Sendo a, b e o vetor x iguais a:

$$a = \boxed{2}$$

$$b = \boxed{4}$$

	0	1	2	3	4	5	6	7	8	9
x =	2	6	8	3	10	9	1	21	33	14

- Escreva o valor correspondente de:

a) $x[a+1] =$

b) $x[a+2] =$

c) $x[a+3] =$

d) $x[a*4] =$

e) $x[a*1] =$

f) $x[a*2] =$

g) $x[a*3] =$

h) $x[x[a+b]] =$

i) $x[a+b] =$

j) $x[8-x[2]] =$

k) $x[x[4]] =$

l) $x[x[x[7]]] =$

m) $x[x[1]*x[4]] =$

n) $x[x[a+4]] =$

Exercício rápido

- Sendo a, b e o vetor x iguais a:

$$a = \boxed{2}$$

$$b = \boxed{4}$$

	0	1	2	3	4	5	6	7	8	9
x =	2	6	8	3	10	9	1	21	33	14

- Escreva o valor correspondente de:

a) $x[a+1] = x[2+1] = x[3] = 3$

b) $x[a+2] = x[2+2] = x[4] = 10$

c) $x[a+3] = x[2+3] = x[5] = 9$

d) $x[a*4] = x[2*4] = x[8] = 33$

e) $x[a*1] = x[2*1] = x[2] = 8$

f) $x[a*2] = x[2*2] = x[4] = 10$

g) $x[a*3] = x[2*3] = x[6] = 1$

h) $x[x[a+b]] = x[x[2+4]] = x[x[6]] = x[1] = 6$

i) $x[a+b] = x[2+4] = x[6] = 1$

j) $x[8-x[2]] = x[8-8] = x[0] = 2$

k) $x[x[4]] = x[10] = \text{Inválido}$

l) $x[x[x[7]]] = x[x[21]] = \text{Inválido}$

m) $x[x[1]*x[4]] = x[6*10] = x[60] = \text{Inválido}$

n) $x[x[a+4]] = x[x[2+4]] = x[x[6]] = x[1] = 6$

Vetores

- Declaração de um vetor em C:

- <tipo> identificador [tamanho];

```
int w[4];
```

- <tipo> identificador [] = { valor1, valor2, ... };

```
int v[] = {1, 2, 3, 4};
```

Vetores

- Declaração de um vetor em C:

- <tipo> identificador [tamanho];

```
int w[4];
```

- <tipo> identificador [] = { valor1, valor2, ... };

```
int v[] = {1, 2, 3, 4};
```



Quando o tamanho não é especificado, o compilador o determina automaticamente com base na quantidade de elementos fornecidos no inicializador.

Vetores

- vetor: array[0..4] of integer = [10, 20, 30, 40, 50]

Para cada elemento em vetor:

Faça algo com o elemento

- Percorrendo o vetor e imprimindo cada elemento:

Para cada elemento em vetor:

Imprimir elemento

Vetores

```
#include <stdio.h>

int main() {

    int vetor[5] = {10, 20, 30, 40, 50};

    printf("Percorrendo o vetor e dobrando cada elemento:\n");
    for (int i = 0; i < 5; i++) {
        int resultado = vetor[i] * 2;
        printf("Elemento %d dobrado = %d\n", vetor[i], resultado);
    }

    printf("\nPercorrendo o vetor e imprimindo cada elemento:\n");
    for (int i = 0; i < 5; i++) {
        printf("vetor[%d] = %d\n", i, vetor[i]);
    }

    return 0;
}
```

Vetores

```
Percorrendo o vetor e dobrando cada elemento:
```

```
Elemento 10 dobrado = 20
```

```
Elemento 20 dobrado = 40
```

```
Elemento 30 dobrado = 60
```

```
Elemento 40 dobrado = 80
```

```
Elemento 50 dobrado = 100
```

```
Percorrendo o vetor e imprimindo cada elemento:
```

```
vetor[0] = 10
```

```
vetor[1] = 20
```

```
vetor[2] = 30
```

```
vetor[3] = 40
```

```
vetor[4] = 50
```

Matrizes

- Enquanto um vetor possui apenas uma dimensão, uma matriz pode armazenar informações em múltiplas dimensões, como as matrizes bidimensionais, tridimensionais e até estruturas com mais dimensões.
- Em uma matriz bidimensional, cada elemento é acessado por dois índices, que representam sua linha e coluna.
- O tamanho da matriz é definido no momento de sua declaração, indicando o número de linhas e colunas (ou dimensões adicionais, se houver).
- De maneira conceitual, uma matriz pode ser vista como um vetor de vetores, em que cada vetor interno corresponde a uma linha da matriz.

Matrizes

- Declaração e inicialização de uma matriz 2x3 de inteiros:

```
matriz: array[0..1, 0..2] of integer
```

- Inicialização dos elementos da matriz:

```
matriz[0][0] := 1
```

```
matriz[0][1] := 2
```

```
matriz[0][2] := 3
```

```
matriz[1][0] := 4
```

```
matriz[1][1] := 5
```

```
matriz[1][2] := 6
```

Matrizes

- Declaração e inicialização de uma matriz 2x3 de inteiros:

matriz: array[0..1, 0..2] of integer

- Inicialização dos elementos da matriz:

matriz[0][0] := 1

matriz[0][1] := 2

matriz[0][2] := 3

matriz[1][0] := 4

matriz[1][1] := 5

matriz[1][2] := 6



```
#include <stdio.h>

int main() {
    int matriz[2][3];

    matriz[0][0] = 1;
    matriz[0][1] = 2;
    matriz[0][2] = 3;

    matriz[1][0] = 4;
    matriz[1][1] = 5;
    matriz[1][2] = 6;

    return 0;
}
```

Vetores

- Declaração de uma matriz em C:

- <tipo> identificador[linhas][colunas];

```
int m[3][4];
```

- <tipo> identificador[linhas][colunas] = { valores, valores, ... };

```
int matriz[2][3] = {1, 2, 3, 4, 5, 6};
```

- <tipo> identificador[][colunas] = { valores, valores, ... };;

```
int matriz[][3] = {1, 2, 3, 4, 5, 6};
```

Vetores

- Declaração de uma matriz em C:

- `<tipo> identificador[linhas][colunas] = { { valores }, { valores }, ... };`

```
int matriz[2][3] = {{1, 2, 3}, {4, 5, 6}};
```

- `<tipo> identificador[][colunas] = { { valores }, { valores }, ... };`

```
int matriz[][3] = {{1, 2, 3}, {4, 5, 6}};
```

Matrizes

- matriz: `array[0..2][0..2]` of integer

Para cada linha em matriz:

Para cada coluna em linha:

Faça algo com o elemento

Matrizes

```
int main() {
    int matriz[2][3];

    matriz[0][0] = 1;
    matriz[0][1] = 2;
    matriz[0][2] = 3;

    matriz[1][0] = 4;
    matriz[1][1] = 5;
    matriz[1][2] = 6;

    printf("Matriz 2x3:\n");
    for (int i = 0; i < 2; i++) {
        for (int j = 0; j < 3; j++) {
            printf("%d ", matriz[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```

Matrizes

```
Matriz 2x3:  
1 2 3  
4 5 6
```



```
int main() {  
    int matriz[2][3];  
  
    matriz[0][0] = 1;  
    matriz[0][1] = 2;  
    matriz[0][2] = 3;  
  
    matriz[1][0] = 4;  
    matriz[1][1] = 5;  
    matriz[1][2] = 6;  
  
    printf("Matriz 2x3:\n");  
    for (int i = 0; i < 2; i++) {  
        for (int j = 0; j < 3; j++) {  
            printf("%d ", matriz[i][j]);  
        }  
        printf("\n");  
    }  
    return 0;  
}
```

Exercícios

1. Implemente um programa para ordenar um vetor de 10 elementos. Adote sua estratégia.
2. Crie um algoritmo que some duas matrizes. Verifique se elas são compatíveis.
3. Crie um algoritmo que receba uma matriz[5][5] e verifique se esta matriz é identidade.
4. Crie um algoritmo que receba uma matriz[5][5] e verifique se esta matriz é triangular superior.

ESTRUTURA DE DADOS

Curso de Licenciatura em Ciências da Computação

Lucas Sampaio Leite



**INSTITUTO
FEDERAL**

Baiano