ESTRUTURA DE DADOS

Curso de Licenciatura em Ciências da Computação

Lucas Sampaio Leite

Professor

INSTITUTO FEDERAL
Baiano

Lucas Sampaio Leite

lucas.leite@ifbaiano.edu.br

Objetivo Geral

INSTITUTO FEDERAL
Baiano

- 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

INSTITUTO FEDERAL
Baiano

- 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

INSTITUTO FEDERAL
Baiano

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

INSTITUTO FEDERAL
Baiano

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

INSTITUTO FEDERAL
Baiano

- 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

INSTITUTO FEDERAL
Baiano

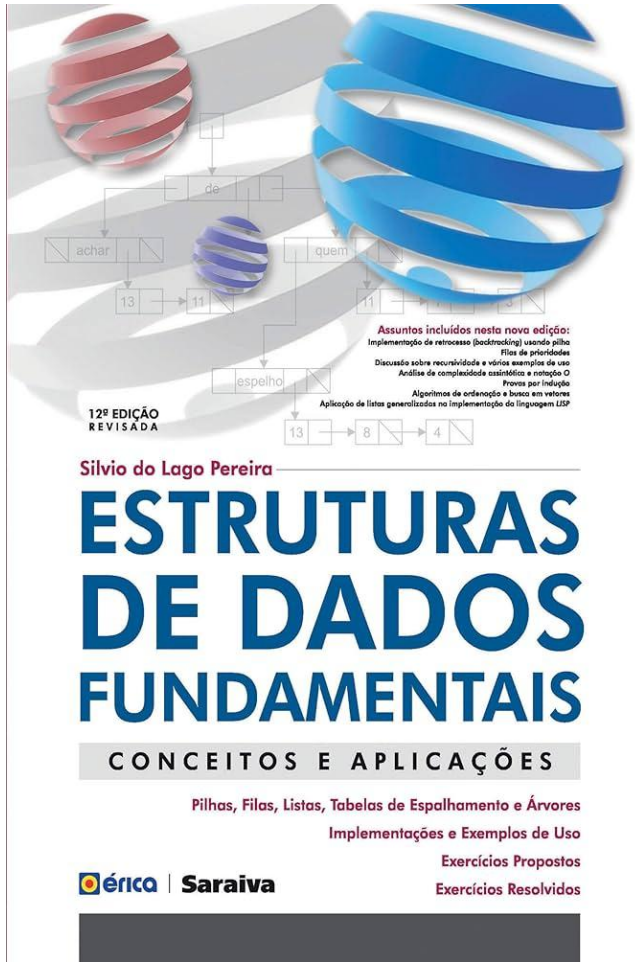
- Aulas síncronas e presenciais
  - Aulas expositivas e dialogadas
  - Abordagem prática
  - Aulas de resolução de exercícios
- **Com chamada**

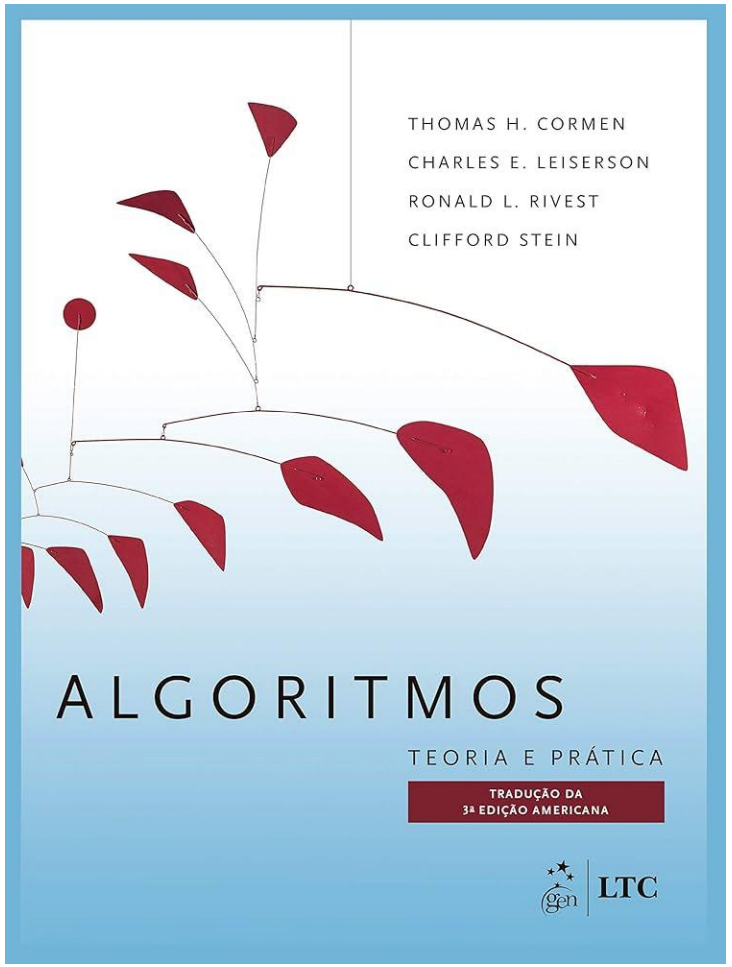
Avaliação

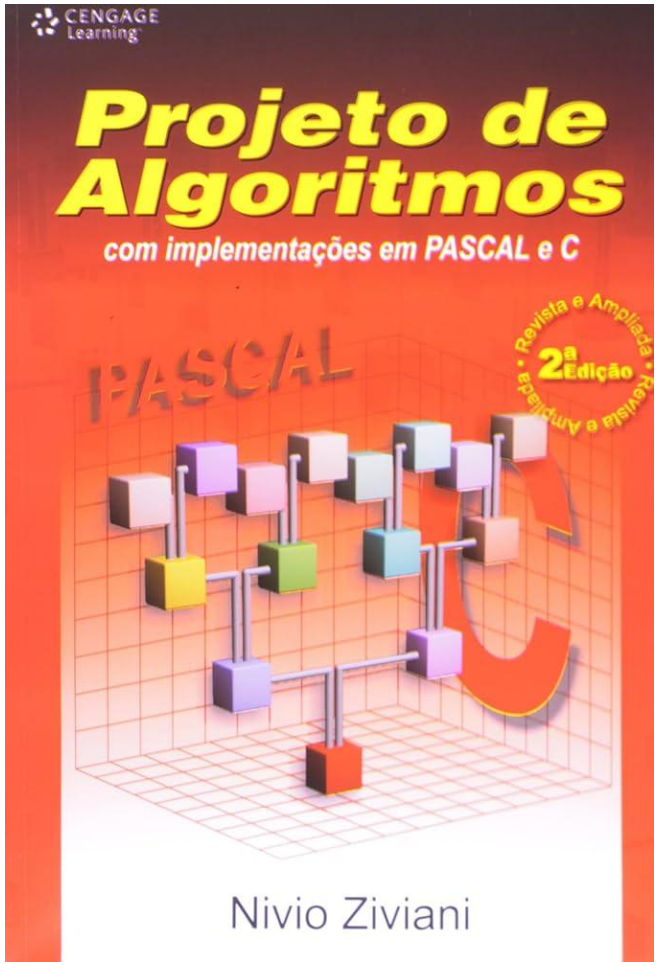
- Avaliação 1: Listas de exercício 30% + Prova 70%
- Avaliação 2: Listas de exercício 30% + Prova 70%
- Prova final

Bibliografia Básica

INSTITUTO FEDERAL
Baiano







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

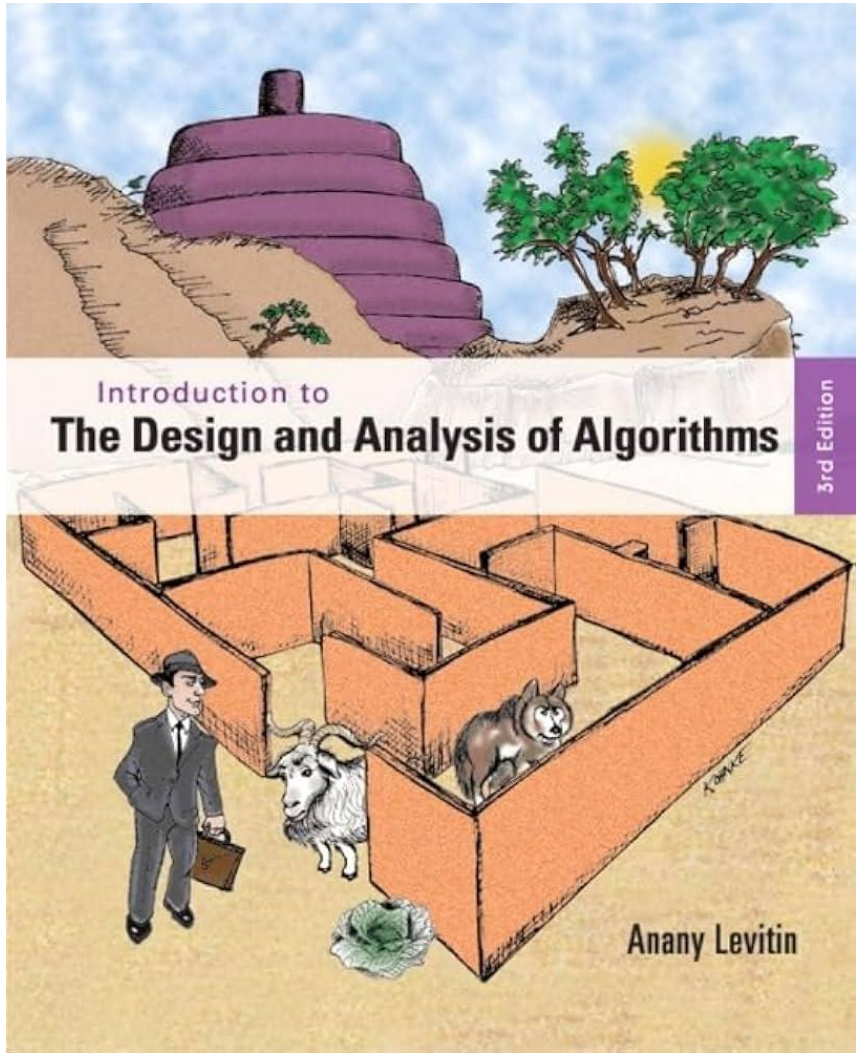
INSTITUTO FEDERAL
Baiano

- 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.

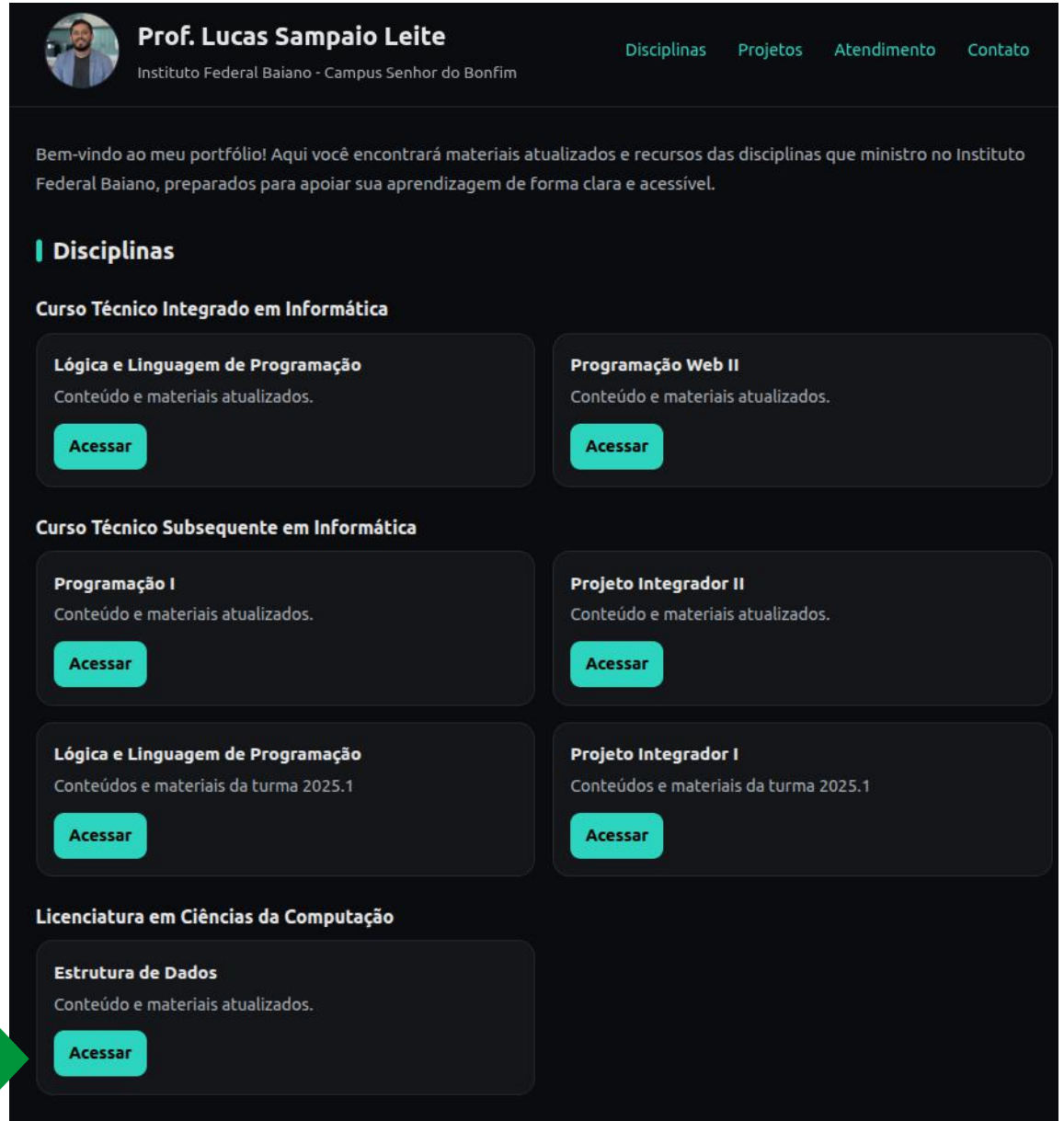
- DEITEL, Harvey M.; DEITEL, Paul J. C ++ como programar. 5. ed. Porto Alegre: Pearson, 2006. ISBN: 9788576050568.

- TENENBAUM, Aaron. Estruturas de dados usando C. São Paulo: Makron, 1995. ISBN: 8534603480.

Outras Referências



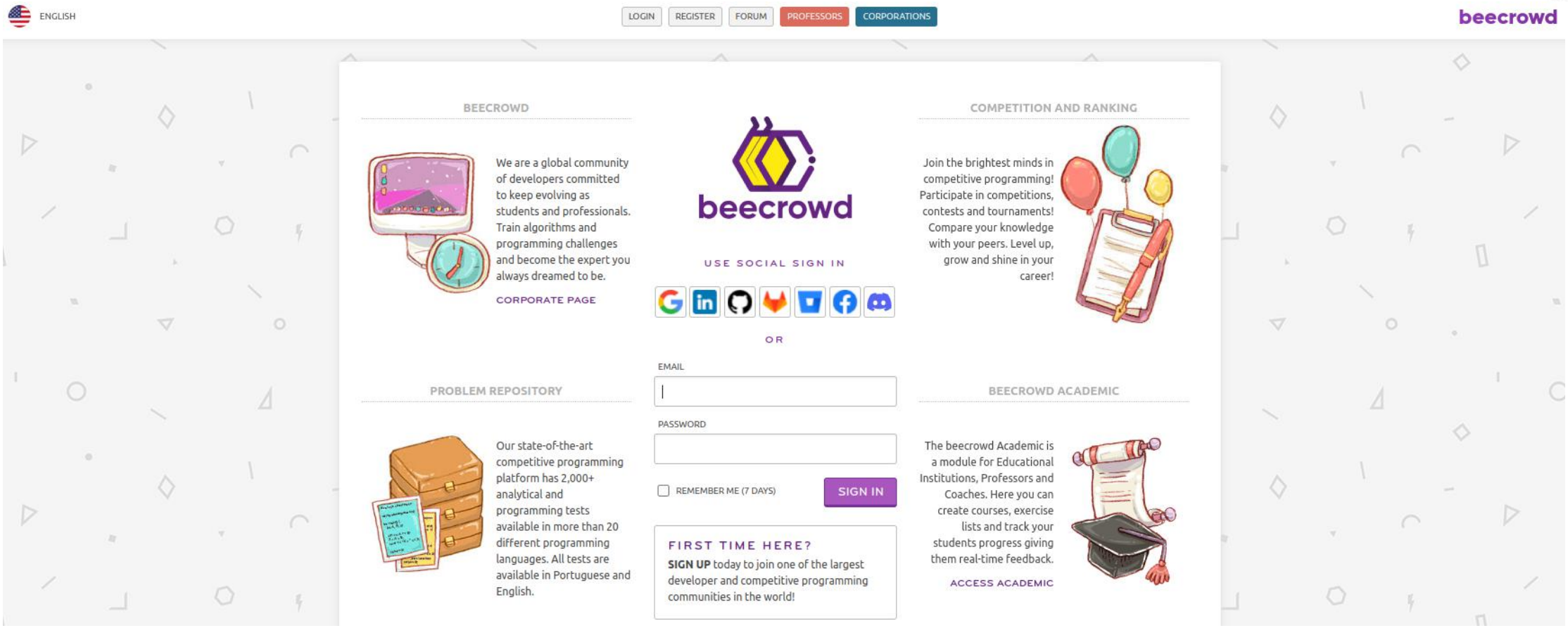
Material de aula e informações sobre a disciplina



http://lucassampaio.tech/

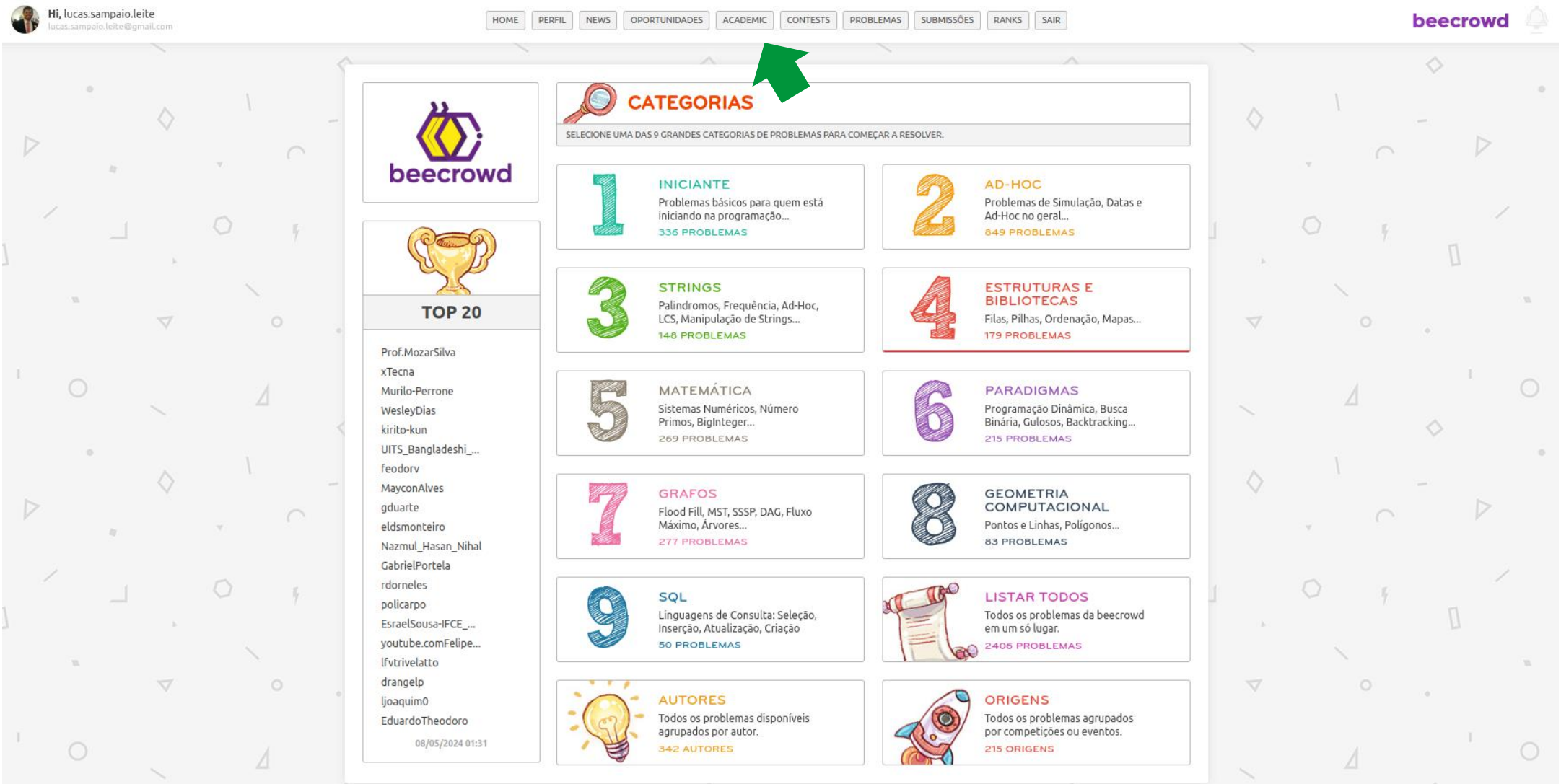
Referência de exercícios Python (com árbitro digital)

INSTITUTO FEDERAL
Baiano



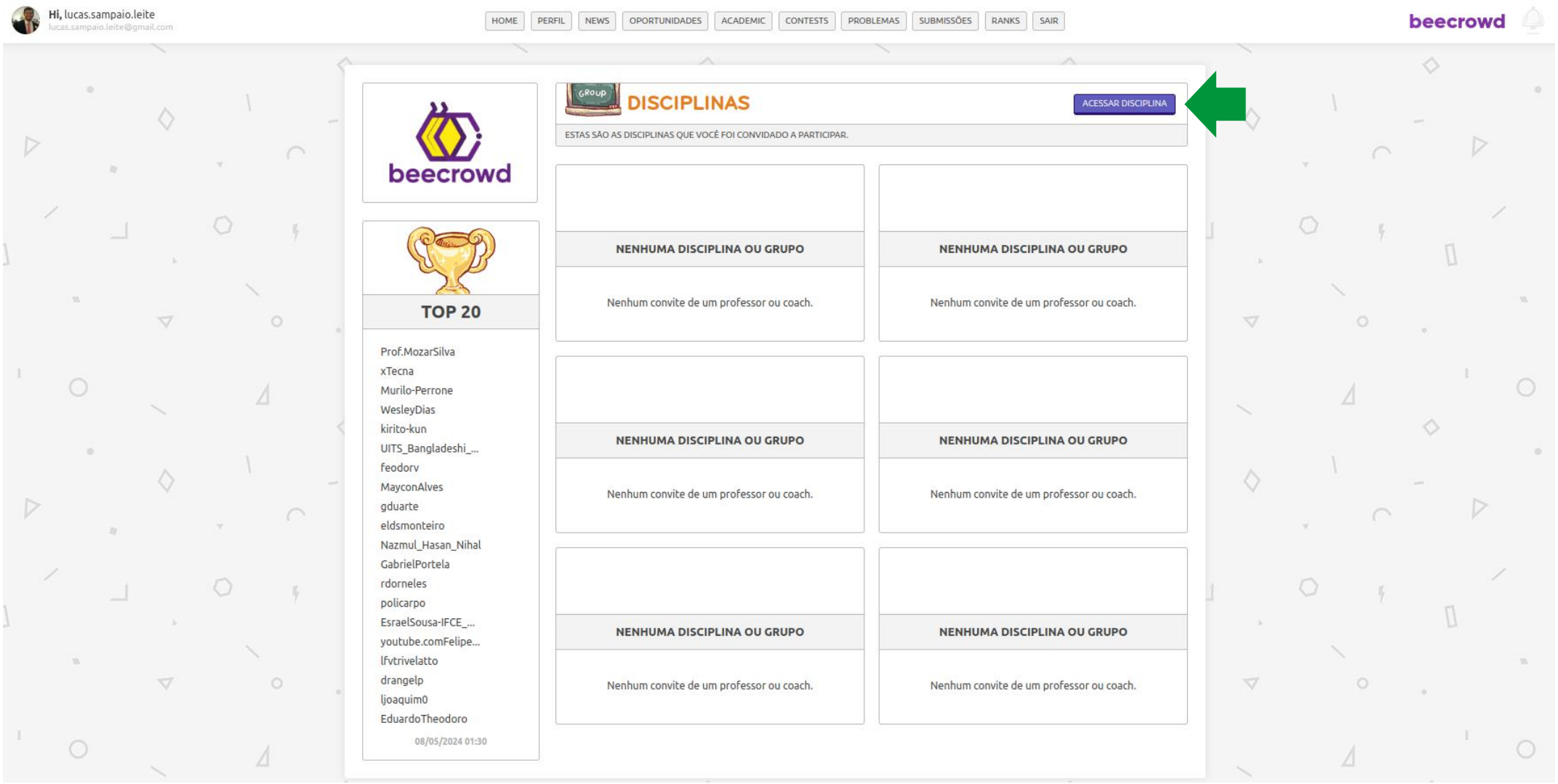
https://judge.beecrowd.com/

Acesso da turma para listas de exercícios



https://judge.beecrowd.com/

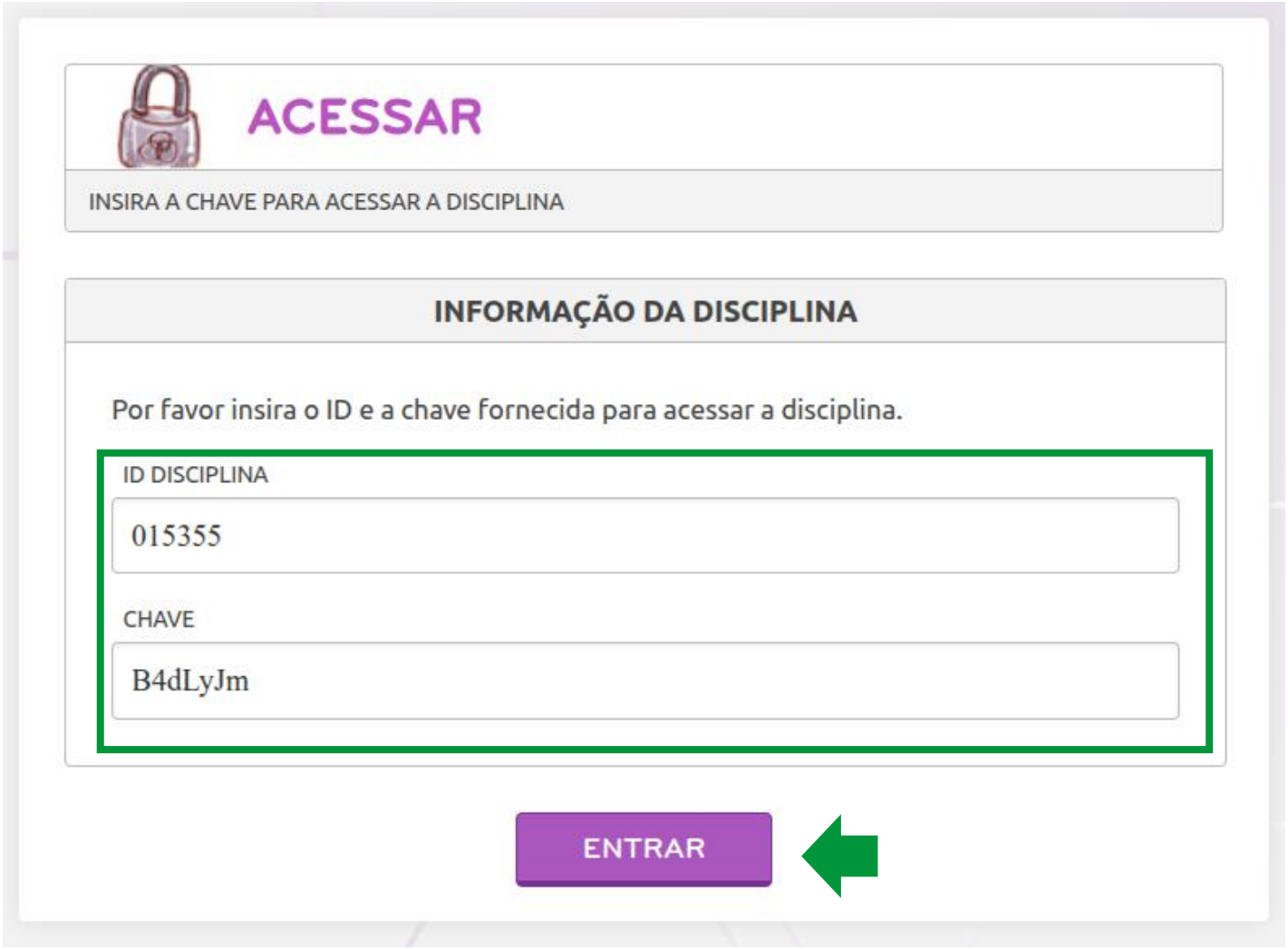
Acesso da turma para listas de exercícios



https://judge.beecrowd.com/

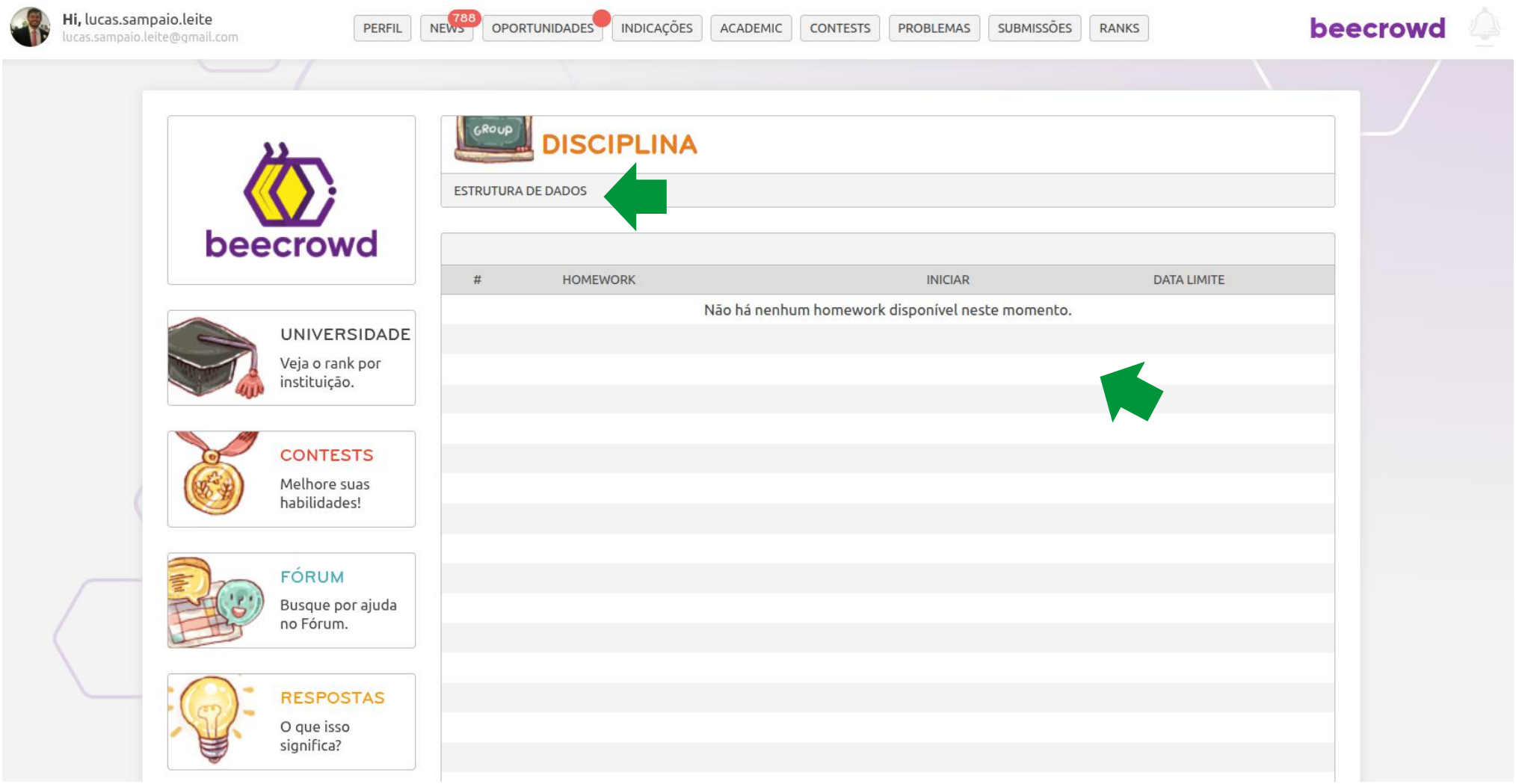
Acesso da turma para listas de exercícios

INSTITUTO FEDERAL
Baiano



https://judge.beecrowd.com/

Acesso da turma para listas de exercícios



https://judge.beecrowd.com/

Boas práticas

INSTITUTO FEDERAL
Baiano

- Para a dinâmica de aprendizagem da disciplina funcionar, é importante realizar as práticas e exercícios passados em sala (não deixem acumular atividades).

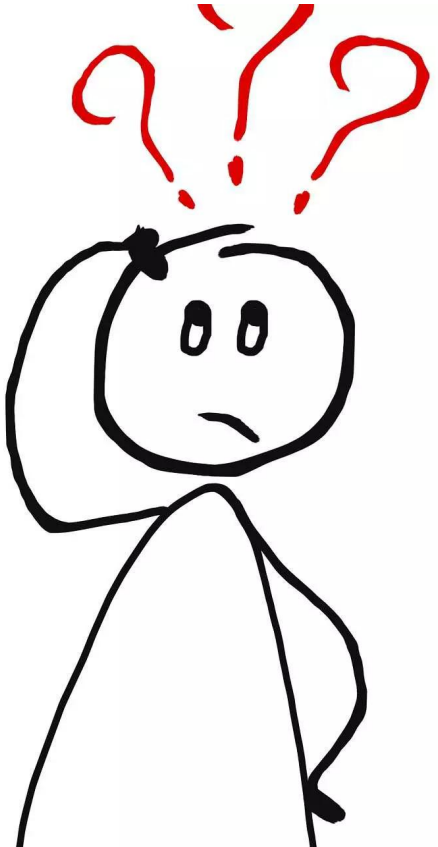
- Quem deixa acumular as atividades, tende a ter um desempenho inferior.

- Não deixem as listas de exercício para última hora.

- Organizem seu tempo entre as disciplinas.

O que é um Algoritmo?

INSTITUTO FEDERAL
Baiano



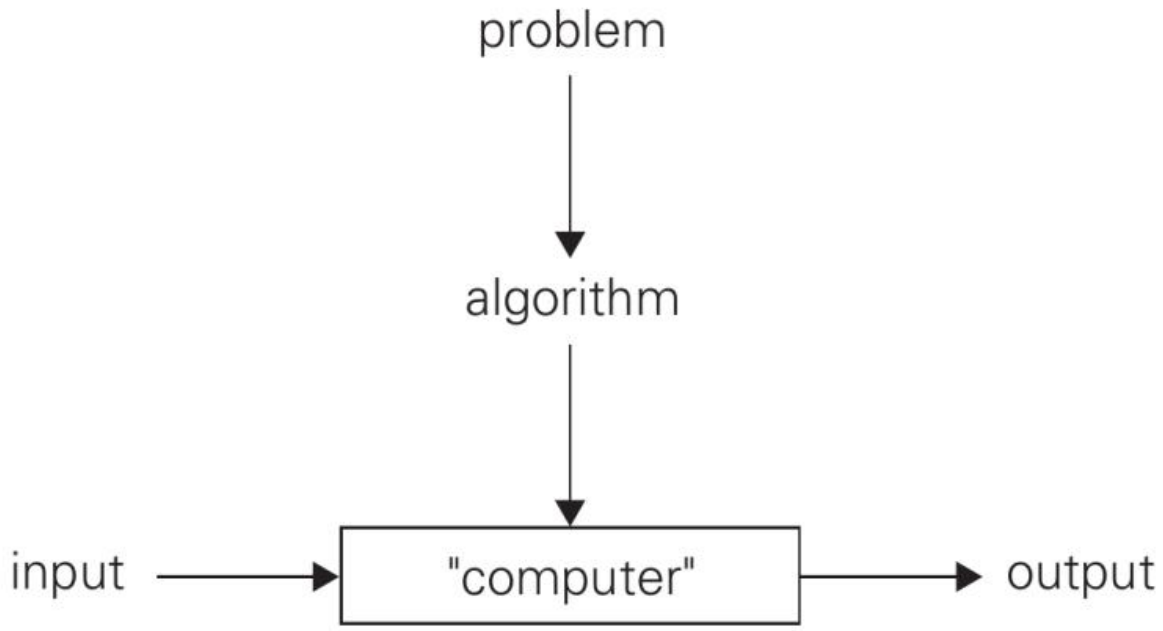
Definição informal (Cormen et al.)

INSTITUTO FEDERAL
Baiano

- 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...

INSTITUTO FEDERAL
Baiano

- 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)

INSTITUTO FEDERAL
Baiano

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

- Sequência de entrada (instância): < 31, 41, 59, 26, 41, 58 >
- Saída esperada: < 26, 31, 41, 41, 48, 49 >

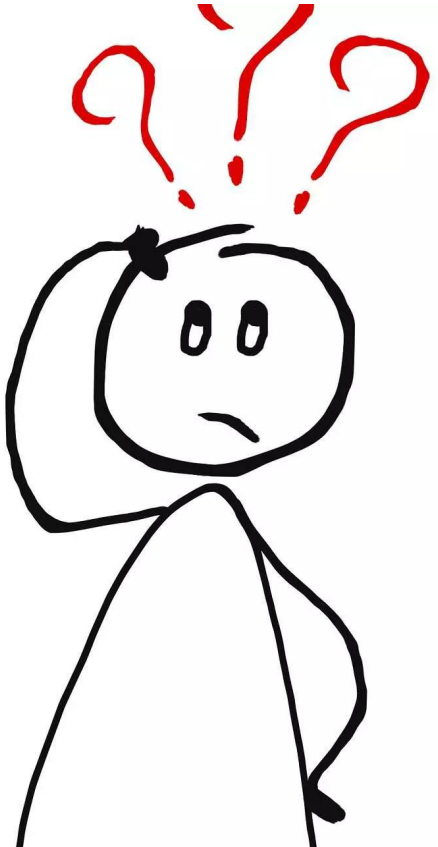
Exemplo (problema da ordenação)

INSTITUTO FEDERAL
Baiano

- 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?

INSTITUTO FEDERAL
Baiano

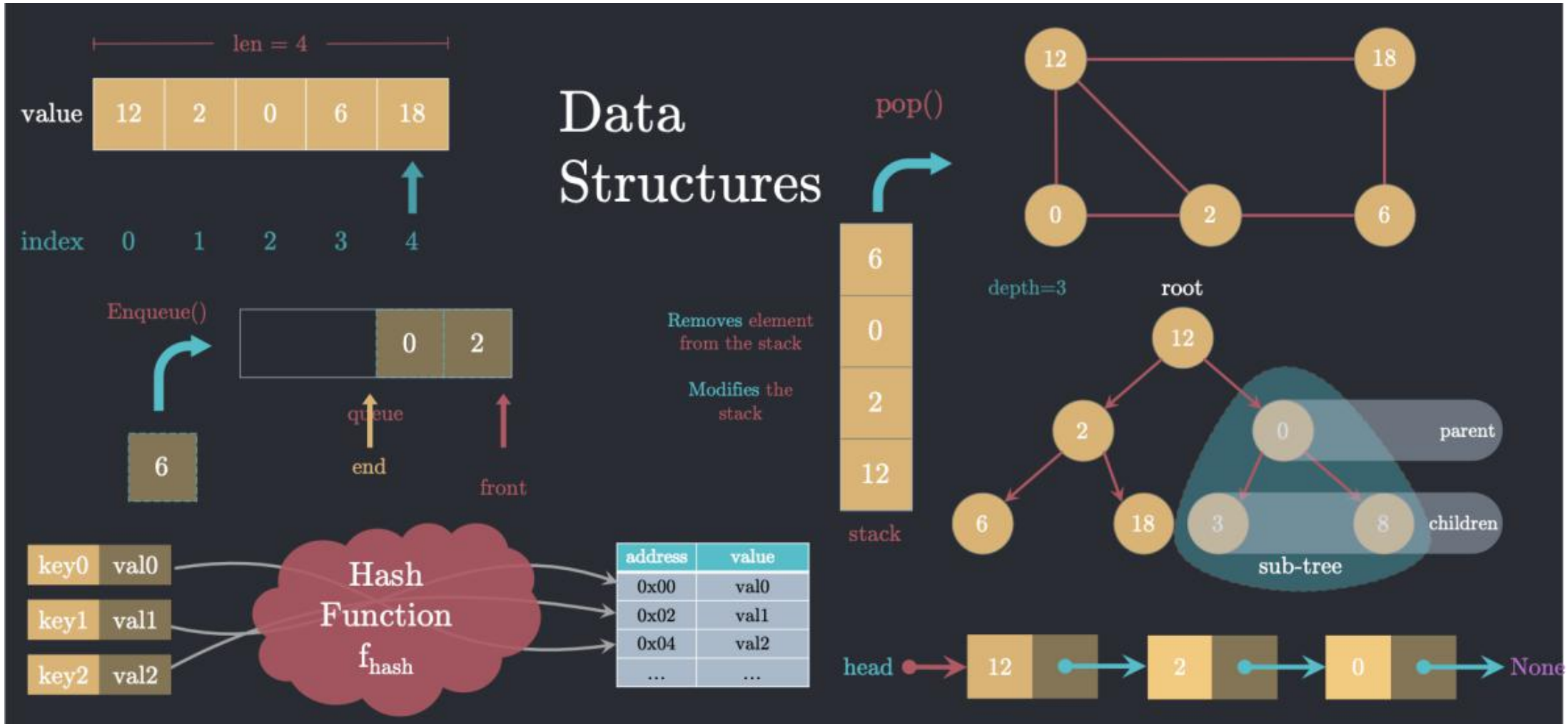


Definição informal (Cormen et al.)

INSTITUTO FEDERAL
Baiano

- 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.

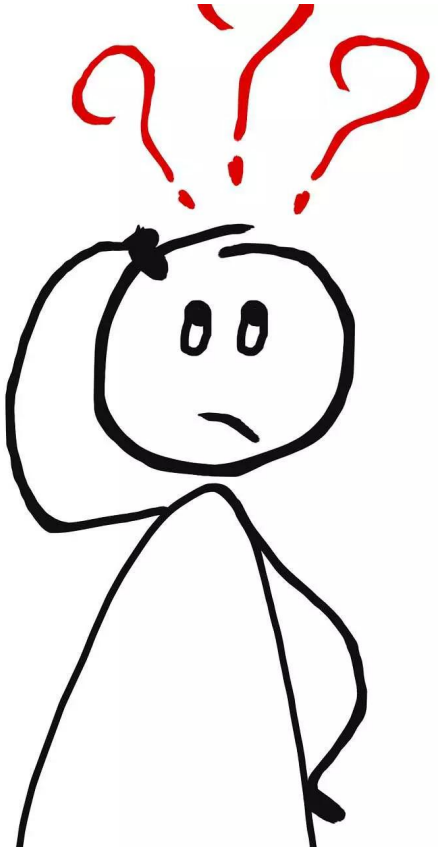
Definição informal (Cormen et al.)



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?

INSTITUTO FEDERAL
Baiano



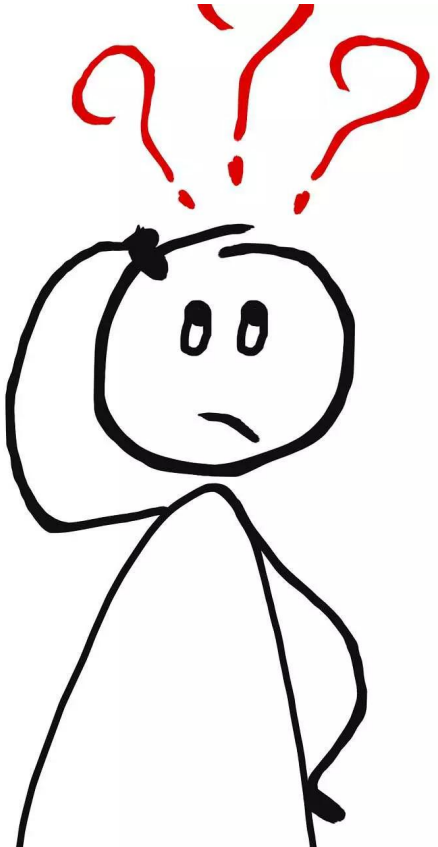
Qual a relação entre algoritmos e estruturas de dados?

INSTITUTO FEDERAL
Baiano

- 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?

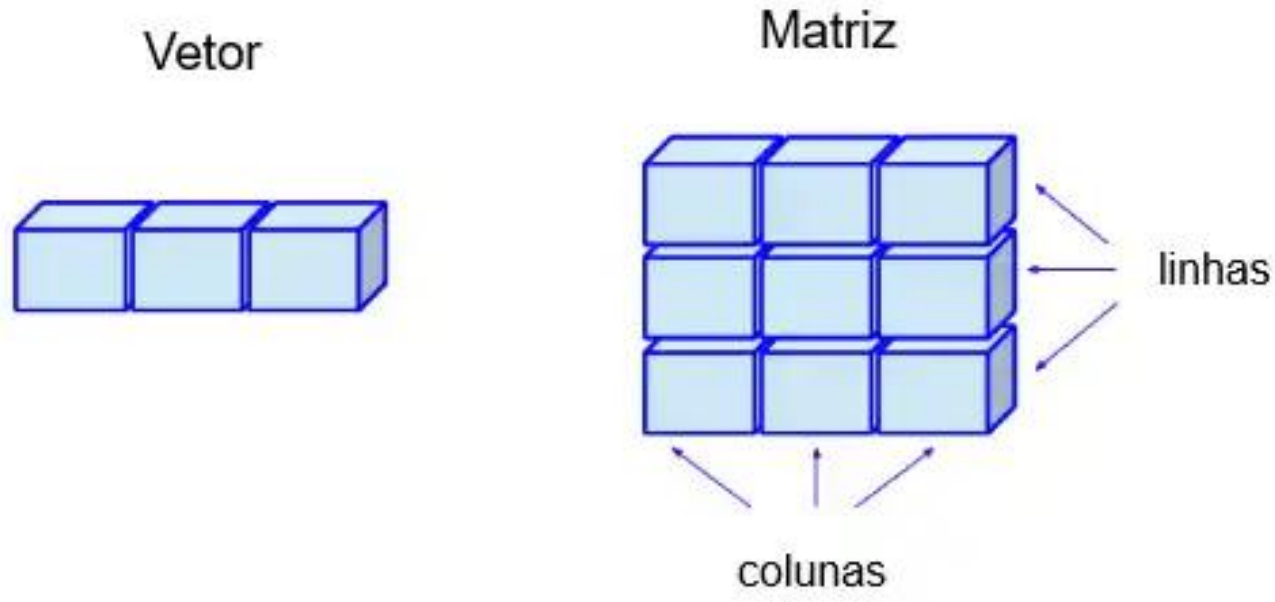
INSTITUTO FEDERAL
Baiano



Vetores e Matrizes

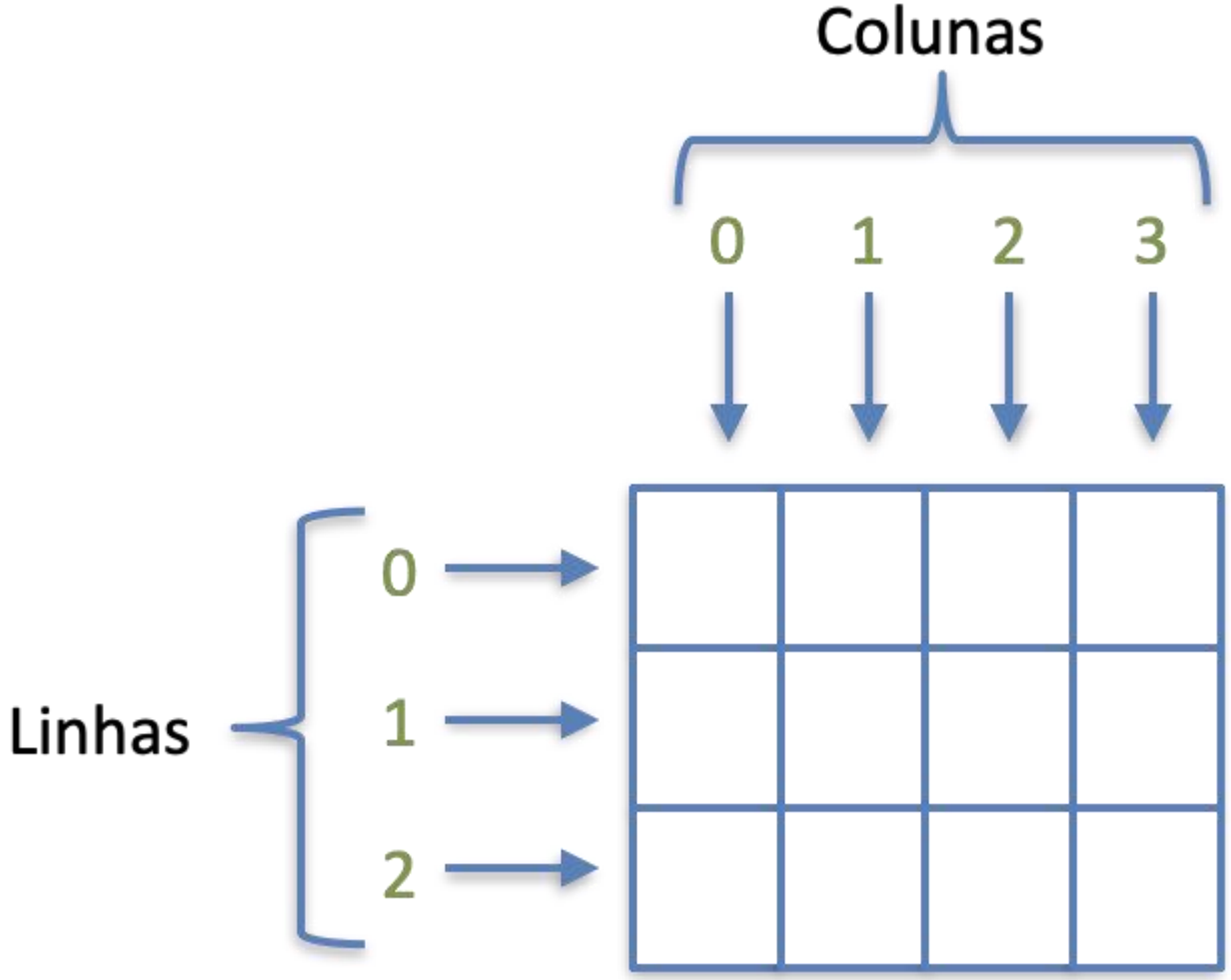
INSTITUTO FEDERAL
Baiano

- 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.



Vetores e Matrizes

INSTITUTO FEDERAL
Baiano



Vetores

INSTITUTO FEDERAL
Baiano

- 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

INSTITUTO FEDERAL
Baiano

- 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

INSTITUTO FEDERAL
Baiano

- 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

```c
#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 = | 2 |

b = | 4 |

x =

0	1	2	3	4	5	6	7	8	9
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

INSTITUTO FEDERAL
Baiano

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

a = 2 **b =** 4 **x =**

0	1	2	3	4	5	6	7	8	9
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] = Inválido
l) x[x[x[7]]] = x[x[21]] = Inválido
m) x[x[1]*x[4]] = x[6*10] = x[60] = Inválido
n) x[x[a+4]] = x[x[2+4]] = x[x[6]] = x[1] = 6

Vetores

INSTITUTO FEDERAL
Baiano

- Declaração de um vetor em C:
  - <tipo> identificador [tamanho];

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

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

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

Vetores

INSTITUTO FEDERAL
Baiano

- 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

INSTITUTO FEDERAL
Baiano

- 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

INSTITUTO FEDERAL
Baiano

```c
#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

INSTITUTO FEDERAL
Baiano

```
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

INSTITUTO FEDERAL
Baiano

- 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

INSTITUTO FEDERAL
Baiano

- 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

INSTITUTO FEDERAL
Baiano

- 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
```

```c
#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

INSTITUTO FEDERAL
Baiano

- 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

INSTITUTO FEDERAL
Baiano

- 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

```c
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;
}
```

INSTITUTO FEDERAL
Baiano

Matrizes

```c
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;
}
```

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

INSTITUTO FEDERAL
Baiano

Exercícios

INSTITUTO FEDERAL
Baiano

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