**discursividade**

Aula de hoje foi abordado sobre discursividade sobre árvores como grau, se ela é binária ou não, ela é binária caso houver apenas 2 caminhos a seguir na árvore, o grau da árvore será dado com base no número de maior filhos dessa árvore

[**https://dontpad.com/estruturas2025**](https://dontpad.com/estruturas2025)

**LINK DA AULA 1505/2025 - novo**

**https://meet.google.com/uej-uspt-bhv**

**# CONCEITO DE RECURSIVIDADE**

**class Counter:**

**count = 0**

**def positionOfFibonacciNaoRecursivo(position):**

**Counter.count += 1**

**if position == 1 or position == 2:**

**return 1**

**a, b = 1, 1**

**for \_ in range(3, position + 1):**

**Counter.count += 1**

**a, b = b, a + b**

**return b**

**def positionOfFibonacci(position):**

**Counter.count = Counter.count +1**

**if position == 1 or position == 2:**

**return 1**

**#calcular recursivamente o N ézimo termo?**

**return positionOfFibonacci(position - 1) + positionOfFibonacci(position - 2)**

**def main():**

**result = positionOfFibonacciNaoRecursivo(36)**

**print("numero de chamadas")**

**print(Counter.count)**

**print(result)**

**main()**

**#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**# 1 - A lista esta vazia? então pode disparar um erro**

**# 2 - A lista contém somente 1 elemento?**

**# 3 - vou em busca do penutlimo elemento**

**# 4 - faço uma copia do ultimo elemento**

**# 5 - anulo a variavel de referencia next do penultimo**

**# 6 - retornar a variavel de referencia do elemento removido**

**# Online Python compiler (interpreter) to run Python online.**

**# Write Python 3 code in this online editor and run it.**

**class Node:**

**def \_\_init\_\_(self, content: str):**

**self.content = content**

**self.next = None**

**class LinkedList:**

**#Inicializador**

**def \_\_init\_\_(self):**

**self.first = None**

**self.qtdElements = 0**

**#Funcao para insercao de um novo elemento**

**def insert(self, content: str):**

**self.qtdElements += 1**

**if self.first == None:**

**self.first = Node(content)**

**return**

**copyFirst = self.first**

**while(copyFirst.next != None):**

**copyFirst = copyFirst.next**

**copyFirst.next = Node(content)**

**#Funcao para impressao de todos os elementos da lista**

**def printList(self):**

**copyFirst = self.first**

**while (copyFirst != None):**

**print(copyFirst.content)**

**copyFirst = copyFirst.next**

**#Funcao que retorna verdadeiro ou falso para lista**

**def isEmpty(self):**

**return self.first == None**

**#Funcao que retorna o numero de elementos da lista**

**def size(self):**

**return self.qtdElements**

**#Funcao que insere um elemento em qq posicao**

**def insertAt(self, position: int, content: str):**

**#Nao sera permitido inserir em qualquer posicao**

**#Lista vazia?**

**#Lista somente 1 elemento?**

**#Demais casos**

**print("Insere o conteudo na posicao desejada")**

**def removeAt(self, position: int):**

**print("remove e retorna o conteudo da posicao desejada")**

**#Funcao que remove e retorna o ultimo elemento da lista**

**def remove(self):**

**#Caso em que lista está vazia**

**if (self.isEmpty()):**

**raise IndexError("Index out of bounds")**

**#Caso em que tenho somente 1 elemento na lista**

**if self.qtdElements == 1 or self.first.next == None:**

**copy = self.first**

**self.first = None**

**self.qtdElements = 0**

**return copy**

**#Tenho no minimo 2 elementos na lista**

**prev = self.first**

**while (prev.next.next != None):**

**prev = prev.next**

**copy = prev.next**

**prev.next = None**

**self.qtdElements -= 1**

**return copy**

**list = LinkedList()**

**print(list.isEmpty())**

**list.insert("maça")**

**list.insert("pera")**

**list.insert("abacaxi")**

**list.insert("laranja")**

**print(list.remove().content)**

**print(list.remove().content)**

**print(list.remove().content)**

**print(list.size())**

**list.printList()**

**===========================================================================================================================================================================**

**LINK DA AULA 1505/2025 - novo**

**https://meet.google.com/uej-uspt-bhv**

**class Node:**

**def \_\_init\_\_(self, content):**

**self.content = content**

**self.left = None**

**self.right = None**

**def \_\_str\_\_(self):**

**return self.content**

**class BinaryTree:**

**def \_\_init\_\_(self):**

**self.root = None**

**def getRoot(self):**

**return self.root**

**def inOrder(self, root):**

**if root is None:**

**return**

**self.inOrder(root.left)**

**print(root)**

**self.inOrder(root.right)**

**def preOrder(self, root):**

**if root is None:**

**return**

**print(root)**

**self.preOrder(root.left)**

**self.preOrder(root.right)**

**def postOrder(self, root):**

**if root is None:**

**return**

**self.postOrder(root.left)**

**self.postOrder(root.right)**

**print(root)**

**def insert(self, root, newContent):**

**if root is None:**

**return Node(newContent)**

**if newContent < root.content:**

**root.left = self.insert(root.left, newContent)**

**else:**

**root.right = self.insert(root.right, newContent)**

**return root # importante!**

**tree = BinaryTree()**

**tree.root = tree.insert(tree.getRoot(), "Laura")**

**tree.root = tree.insert(tree.getRoot(), "Bia")**

**tree.root = tree.insert(tree.getRoot(), "Carla")**

**tree.root = tree.insert(tree.getRoot(), "Douglas")**

**tree.root = tree.insert(tree.getRoot(), "Maria")**

**tree.root = tree.insert(tree.getRoot(), "Paulo")**

**tree.root = tree.insert(tree.getRoot(), "Nara")**

**tree.root = tree.insert(tree.getRoot(), "Otavio")**

**tree.preOrder(tree.getRoot())**

**Tarefa fazer uma função paar conta o tamanho da árvore**