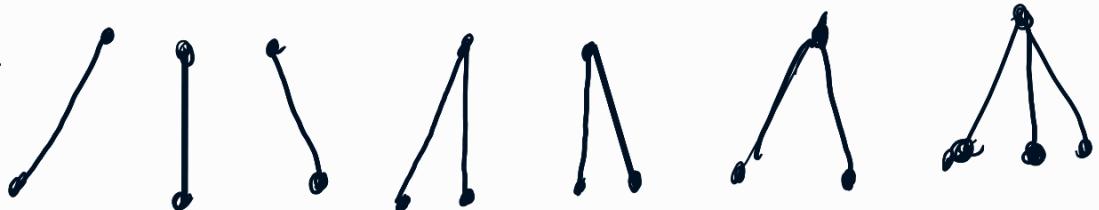


Questão 01)

1)

Base:

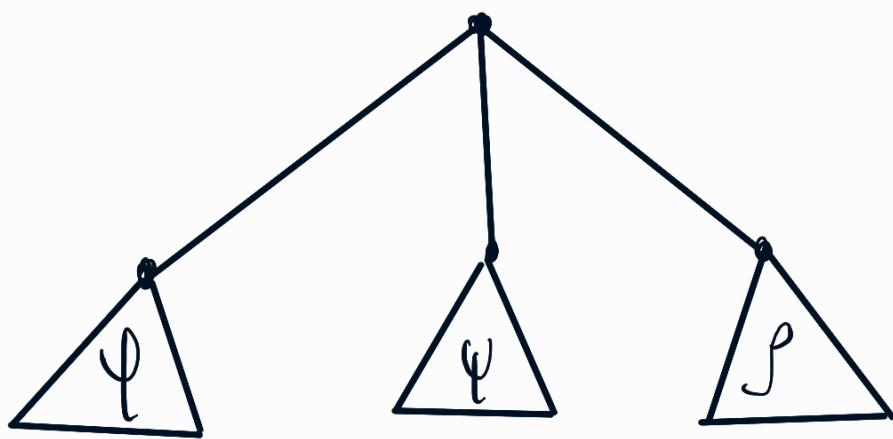
Recursão:



2)

Base:

Recursão: Seja  $\varphi$ ,  $\psi$  e  $\rho$  árvores estritamente ternárias



Para o caso da árvore estritamente terária sabemos que o número de vértices no caso base é 1, o que gera resto 1 quando dividido por 3.

Agora para fora do caso base teremos  $1 + \# \Psi + \# \Psi + \# \mathcal{P}$ . Por definição cada uma dessas árvores terá 1 no base que terá nenhum ou 3 filhos.

Analisemos primeiro o caso onde as 3 árvores possuem só a base:

$$1 + 1 + 1 + 1 = 4 \div 3 \Rightarrow 1$$

Agora analisemos o caso onde todas possuem filhos, mas não possuem netos:

$$1 + \underbrace{(1+3)}_{\#\Psi} + \underbrace{(1+3)}_{\#\Psi} + \underbrace{(1+3)}_{\#\mathcal{P}}$$

$$1 + 12 = 13 \div 3 \Rightarrow 1$$

Para além disso, podemos sempre criar uma árvore estritamente ternária como  $\varphi, \psi$  e  $\rho$  da definição.

Assim, para o caso geral, teremos

$$1 + \varphi + \psi + \rho$$

que se necessário expansão, será da forma  $\varphi = 1 + \varphi' + \varphi'' + \varphi'''$ . todas são árvores estritamente ternárias.

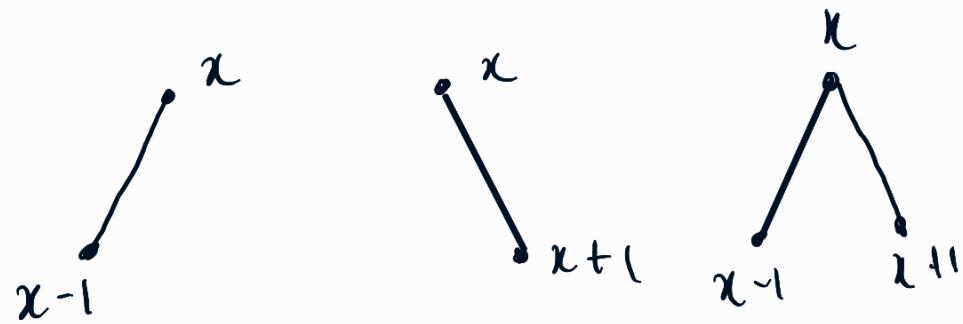
Logo, a prova se da por indução via recursão no caso base.

(Questão 02)

Base

$x$

Recursão



Busca em 'arvore binária'

BaB( $x$ : int, root: Node)  $\rightarrow$  bool

If  $\text{get\_nodes}(\text{root}) \neq 2$ :

    left, right  $\leftarrow \text{get\_nodes}$

    If not right and  $x > \text{left.value}$ :  
        return false

    elif not left and  $x < \text{right.value}$ :  
        return false

    if  $x == \text{root.value}$ :  
        return true

    left, right  $\leftarrow \text{get\_nodes}(\text{root})$

    if ( $x > \text{root.value}$ ):

        BaB( $x$ , right)

    if ( $x < \text{root.value}$ ):

        BaB( $x$ , left)

A busca funciona de forma recursiva.

Como existe a possibilidade de um nó não ter dois filhos e preciso validar se é possível que o valor que procuramos ainda pode ser encontrado, é o que acontece no primeiro if.

Já o segundo valida se encontramos o valor.

O terceiro e quarto compõem com a parte recursiva, passando para a função a direção de devemos procurar.

(Questão 03)



Chamemos  $v_0$ ,  $v_1$ , e  $v_2$  de  $g_0$ ,  $g_1$ , e  $g_2$ . São os co-  
grafos com apenas um vértice  
e nenhuma aresta, ou seja:

$g_i$  é o co grafo com o vértice  
 $v_i$  e nenhuma aresta.  $i = \{0, 1, 2\}$ .

$g_0$   $g_1$   $g_2$  Primeiro, como  $v_0$  e  $v_2$  não são conexos,  
façamos a união  
disjunta.

$\varphi$  é o co grafo da união  
entre  $g_0$  e  $g_2$  sem suas arestas.

Como  $g_1$  é conexo com  $g_0$  e  $g_2$  faremos o join entre  $g_1$  e  $\varphi$ , on-

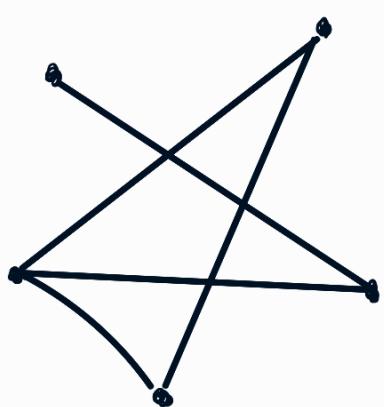
de o join é a união disjunta entre os  
cografos adicionando todas as arestas en-  
tre os vértices.

Chamemos de  $P_3$  o join  
de  $g_1$  com  $\varnothing$ . Logo,  $P_3$  tem  
a topologia



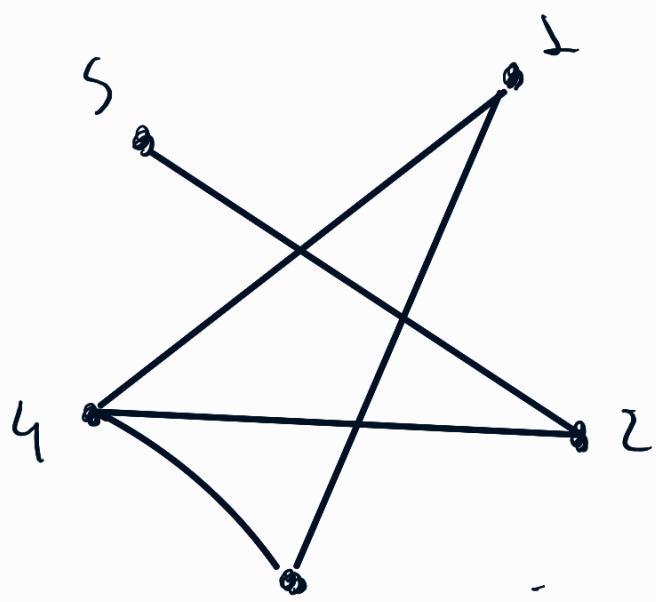
e é um cografo.

d)



Prove que não é  
um cografo.

Vamos começar enumerando os vértices:



A única forma de 4 e 5 não serem conexos é se considerarmos cada vértice como um cografio e fizermos a união

disjunta entre eles, assim, ambos são cografios, a união disjunta com cografio sem arestas.

Peguemos 2 como exemplo.

Como 2 é conexo com 4 e 5 basta fazermos um join e

tudo funcionaria.

Esqueçamos o 2 por um momento e focemos no 1, no caso de fazermos um join entre 1 e {4,5} teremos uma aresta o mais e no caso de fazermos join(1,4) visto que agora teríamos problemas com o 2, que não é conexo com 1.

Logo, não é possível construir esse grafo apenas com as regras dos cografos.

Podemos concluir, enfim, que não é um cografo.

(Questão 04)

Base:  $\epsilon, (, )$

Recursivo: Se  $($  e  $)$  são palavras já definidas em  $\Sigma$  a concatenação delas é palavra se e somente se a pilha  $\Psi$  tiver formada 0 quando o último símbolo for descoberto.

Seja  $\Psi$  uma pilha clássica LIFO onde para cada  $($  adicionamos um elemento e para cada  $)$  removemos um elemento.

Uma função recursiva pode ser vista da seguinte forma:

$\varphi = 0$

def Pilha (palavra) :

If len(palavra) == 0 :  
    return  $\varphi$

If palavra.pop() == ' ' :

$\varphi++$

função do python  
para pegar o último  
item de uma  
lista.

else :

$\varphi--$

"Palavra" é uma  
lista de caracteres  
"(", ")" .

Pilha (palavra[0 : len(palavra)-1])

Chama Pilha sem o último char.

## Prova

A palavra vazia é balanceada  
pois a  $\varphi$  tem tamanho zero e se  
comercarmos a soma do enunciado  
terminaríamos em 0.

A palavra de tamanho  $m$ , quando  
 $\varphi$  tem tamanho diferente de  
0 pode ter dois cenários, assim como  
no exemplo:

Se  $\varphi > 0$ , desbalanceada  
com mais "" que "".

Se  $\varphi < 0$ ,        "        menos

"" que "".

Porém, por com brincas, aceita-se apenas palavras onde ao final da recursão  $f=0$ .

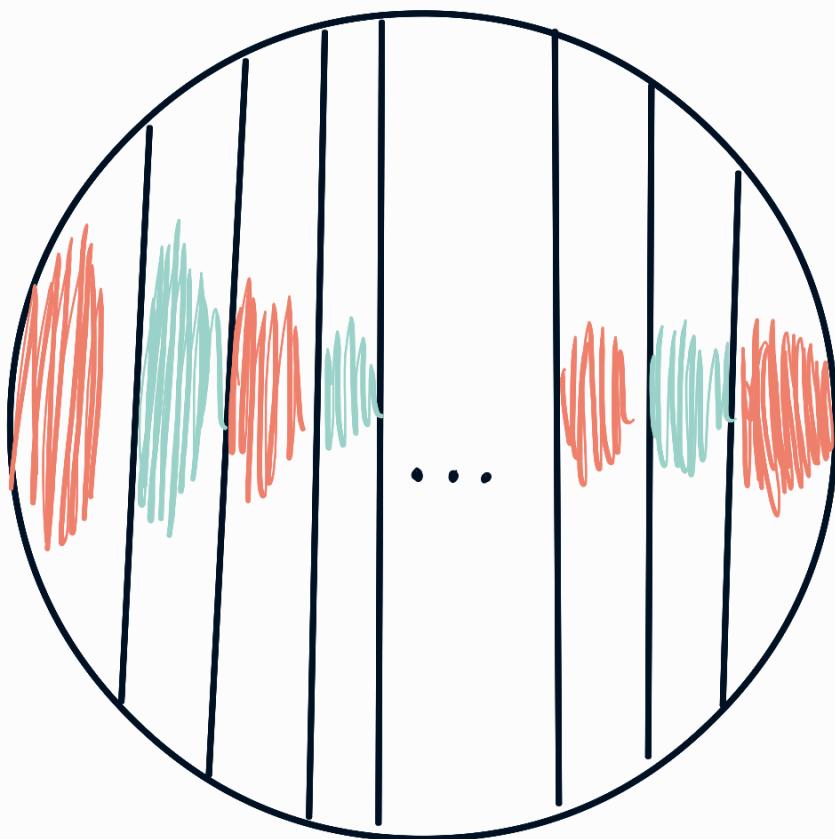
obs: A definição do enunciado e a minha são a mesma.

(Questão 05)

O caso base só precisa de uma cor para ser colorido, logo, é bicolorável.

Toda adição de corda que não cruza outra corda separa o mapa em  $I + k$  sendo  $k$  o número de cordas. Como não há cruzamento podemos combinar dois a dois e pintar o mapa haja vista que, em essências

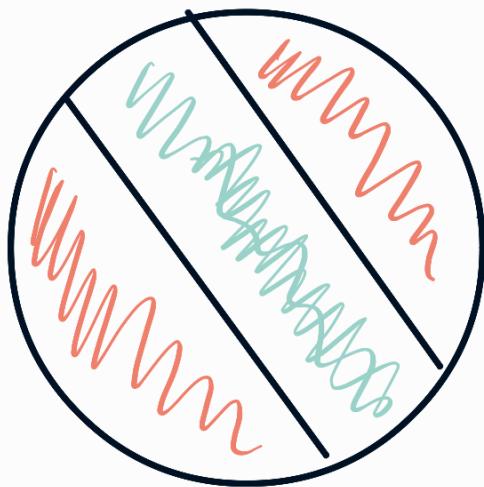
da primeira e da ultima partição, todas as outras fazem fronteira com exatamente duas.



Como sempre podemos fazer essa combinação, esse cenário também é bi-colorável.

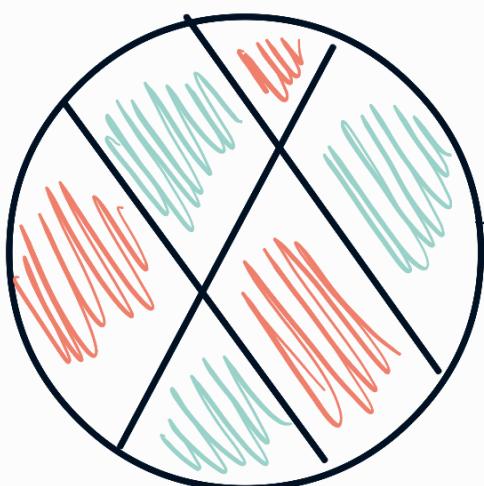
Agora, caso alguma corda cruze a outra iremos repartir em seções e reversivamente inverter o padrão de pintura escolhido na etapa anterior entre as seções.

Pintemos da seguinte forma  
recursiva:



- Caso trivial desconta acima onde podemos escolher duas das cores.

• Adicionemos agora 2 cruzamentos



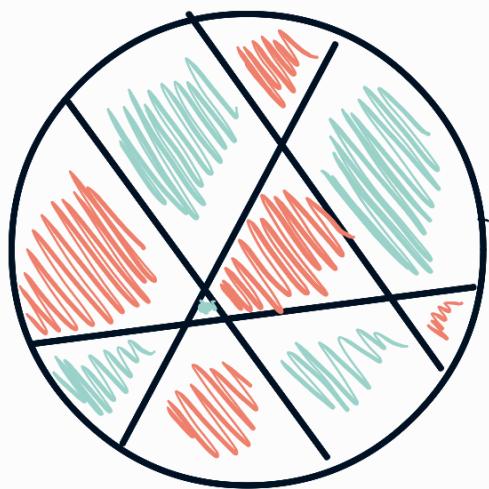
seção + e  
ra do passo  
2.

• Chamemos de seção 1 a seção superior e 2 a inferior.  
Repetiremos a pintura do passo anterior na

inverte-se a pintura anterior na seção

Adicionemos mais cruzamentos

- ① por definição, fronteira é entendida como um segmento de reta.



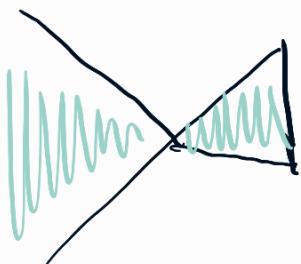
Novamente consideramos a seção 1 como a superior e a seção 3, agora, como o inferior.

A seção dois pode ser vista como a junção das componentes do meio do mapa.

Nova mente, recursivamente pintarmos a seção 1 com as cores do passo anterior, para o zíper termos as cores e nesse caso,

em especial, usamos a definição  
I para pintar o pequeno triângulo  
de azul, invertendo as cores  
somente entre as fronteiras.

Peda de fimeções do mapa.  
o cruzamento



Pode ser simplificado  
para uma estrutura  
sem encaimento pois  
a fronteira é definida por um  
segmento de reta e não  
um ponto, por isso podemos  
pintar essa estrutura de azul.

Questão 06)

Comemos com  $m = 3$ .



Calculando as diagonais, nota-se que

$$\frac{3(3-3)}{2} = 0, \text{ onde}$$

é verdade haja vista

a definição de diagonal

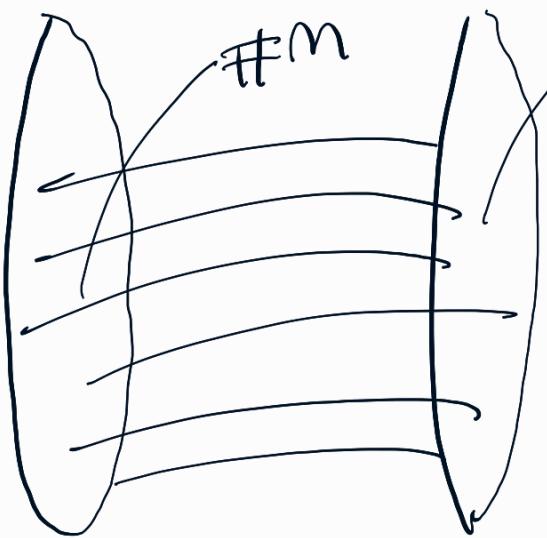
(ligar dois lados não consecutivos).

Para  $m > 3$ , temos:

Considere que o polígono em questão é convexo, logo, nenhum segmento de reta ligando lados distintos do polígono encontra outro lado,

considere ainda um lado qualquer dentro desse polígono.

Assim, no passo anterior, que todo triângulo tem 0 diagonais, logo, esse lado em específico tem  $m-3$  possíveis relações.



Devemos ainda considerar o simétrico desse lado, pois

a sua diagonal é a mesma do lado selecionado inicialmente, logo, deveremos remover a repetição dividindo tudo por 2.

Logo, o número de diagonais é  $\frac{n(n-3)}{2}$

b) O caso base novamente é trivial, com  $n=3$ ,  $n-2=1$ , que é o próprio polígono.

Para  $n > 3$ , considere  $G$  e  $H$  partes de um polígono conexo

Onde  $H$  tem  $h$  lados  $\frac{h(h-3)}{2}$  diagonais e  $h-2$  triângulos

e que  $G$  tem  $g$  lados  $\frac{g(g-3)}{2}$  diagonais e  $g-2$  triângulos.

Por definição, nem he nem a diagonal de  $H$  pode encontrar um lado de  $G$  e vice-

versa, porem, todos os triângulos de P são aqueles em G, H e na união entre G e H removendo as ocorrências exclusivas de G e H. (inner join).

Assim, seja  $\psi$  a quantidade de triângulos da união de G e H, sem considerar os triângulos de G e H, teremos:

$$\psi = \underbrace{g+h-2}_{\text{todo}} - \underbrace{(g-2)}_{T.G} - \underbrace{(h-2)}_{T.H}$$

$$\begin{aligned}\psi &= g+h-2 - g+2 - h + 2 \\ &= 2.\end{aligned}$$

Logo,  $g+h=m \Rightarrow m-h=g$

O polígono tem

$$\underbrace{g-2}_{t.g} + \underbrace{h-2}_{t.h} + \varphi$$

$$= m - h - 2 + h - 2 + \varphi$$

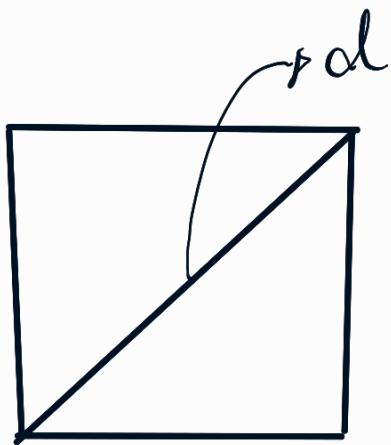
$$= m - 4 + \varphi$$

$$= m - 2 \text{ triângulos}.$$

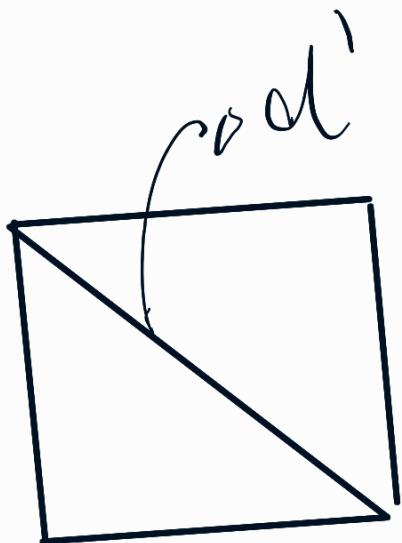
c) Para o caso base  $P=4$ ,  
teremos 2 triângulos e  $\frac{4(4-3)}{2}$   
= 2 diagonais.

Como  $P=4$  é um quadrado,

é trivial perceber que uma diagonal, d, é uma aresta dos dois triângulos.



ou



Agora, para todo  $P$  com  $n > 4$ ,  $P$  terá  $\frac{n(n-3)}{2}$  diagonais e  $n-2$  triângulos.

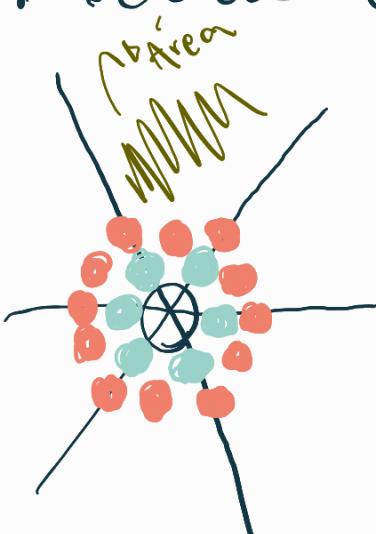
Como os triângulos cobrem o polígono todo, sem sobrar ou faltar, assim como as arestas dos triângulos não se cruzam,

para toda diagonal de  $P$  haver de existir pelo menos um triangulo o qual d seja aresta, do contrario, a reta de limitada por d criaria um a secas onde nã existiria triangulizacã, o que é um absurdo.

Questão 07)

Para criarmos uma definição recursiva para esse problema, analisemos a sua estrutura geométrica.

Uma moeda pode "receber" outras moedas da seguinte forma:



Se pegarmos dois segmentos consecutivos dessa construção e analisarmos, nota-se que:

A quantidade de moedas nessa área cresce a medida que aumentamos o valor de  $m$ , na ordem de  $(m-1) + f(m-1)$ . Ou seja, para definir o valor da função no ponto  $m$ , soma-se o valor de  $(m-1) +$  o valor da função definida em  $f(m-1)$ .

Nota-se que essa decimal tende a convergir para um valor base, nesse caso, definiremos  $f(0) = 0$ .

Olhando no desenho quando temos apenas uma moeda, a área entre dois segmentos de reta não possui nenhuma moeda -



Cria-se então, para a área entre os segmentos a seguinte recursão

$$f(0) = 0$$

$$f(1) = f(0) + (1-1) = 0 \quad (\text{Nas tem moedas mas área apenas nas bordas})$$

$$f(2) = f(1) + (2-1) = 1$$

$$f(n) = f(n-1) + (n-1)$$

↑ tem uma moeda na área.

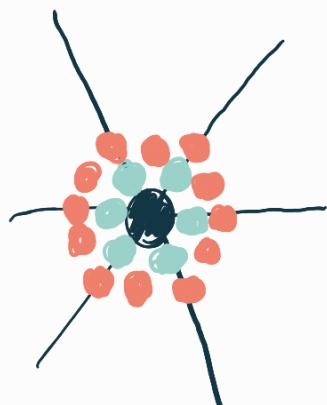
Agora para a recursão da fronteira, ou seja, para as moedas em cima dos segmentos, teremos:

$$g(0) = 1$$

$$g(1) = g(0) + b = 7$$

$$g(2) = g(1) + b = 13$$

$$g(n) = g(n-1) + b$$



Podemos definir uma outra função que seja a composição das duas, o que descreveria bem a recursão do conjunto numérico compacto, como:

$$\psi(n) = b f(n) + g(n)$$

$$\psi(0) = b(0) + g(0) = 0 + 1 = 1$$

$$\varphi(1) = b(0) + g(1) = 0 + 7 = 7$$

$$\varphi(2) = b(f(2)) + g(2)$$

$$= b(r + f(1)) + (g(1) + b)$$

$$= b + (b + 7) = 19$$

Apesar de fum ciamar podemos simplificar a definição para apenas uma fum ção.

Nota-se que a fumçāo  $g(n)$  assim como a  $f(n)$  usam o termo  $n-1$  em suas recursões, porém, a  $g(n)$  soma um valor constante em cada iteração.

Podemos simplificar a definição recursiva fazendo as características de  $g(n)$  para  $f(n)$ , como a soma do termo constante e seu caso base.

Assim,

$$f(0) = 1$$

$$f(1) = f(0) + b + b(1-1) = 7$$

$$f(2) = f(1) + b + b(2-1) =$$

$$7 + b + b = 19$$

$$f(n) = b(n-1) + b + f(n-1)$$

$$= bn - b + b + f(n-1)$$

$$= bn + f(n-1)$$

$$f(1) = b + 1 = 7$$

$$f(2) = 12 + 7 = 19$$

$$f(3) = 18 + 19 = 37$$