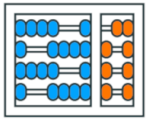


Instituto de Computação

UNIVERSIDADE ESTADUAL DE CAMPINAS

UNICAMP

MC102 - Aula 25

Exemplos sobre Recursão (parte 4)

Algoritmos e Programação de Computadores

Turmas

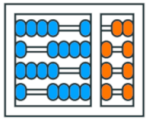
OVXZ

Prof. Lise R. R. Navarrete

lrommel@ic.unicamp.br

Terça-feira, 28 de junho de 2022

21:00h - 23:00h (CB06)



Instituto de Computação

UNIVERSIDADE ESTADUAL DE CAMPINAS

UNICAMP

MC102 – Algoritmos e Programação de Computadores

Turmas

OVXZ

https://ic.unicamp.br/~mc102/

Site da Coordenação de MC102

Aulas teóricas:

Terça-feira, 21:00h - 23:00h (CB06)

Quinta-feira, 19:00h - 21:00h (CB06)

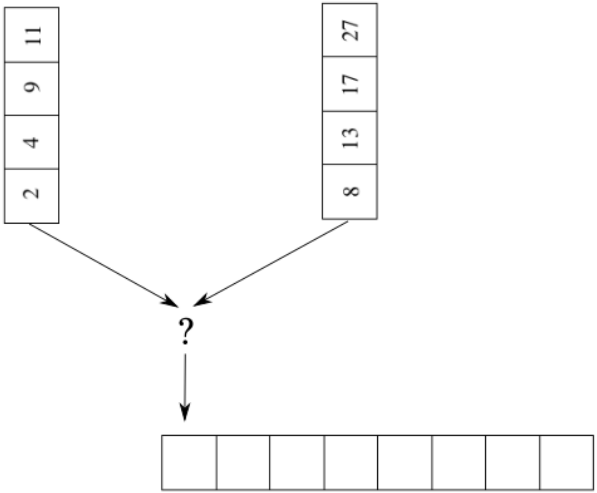
Conteúdo

- Exemplo 22
- Exemplo 23
- Exemplo 24
- Exemplo 25
- Exemplo 26
- Exemplo 27

Exemplo 22

https://ic.unicamp.br/~mc102/aulas/aula13.pdf

Implemente uma versão recursiva da função *merge*.


11
9
4
2
27
17
13
8
?

https://ic.unicamp.br/~mc102/aulas/aula13.pdf

```
1  def merge(lista1, lista2):
2    i = j = 0
3    aux = []
4
5    while (i < len(lista1)) and (j < len(lista2)):
6      if lista1[i] < lista2[j]:
7        aux.append(lista1[i])
8        i = i + 1
9      else:
10       aux.append(lista2[j])
11       j = j + 1
12
13   while i < len(lista1):
14     aux.append(lista1[i])
15     i = i + 1
16
17   while j < len(lista2):
18     aux.append(lista2[j])
19     j = j + 1
20
21   return aux
```

https://colab.research.google.com/

```
def merge(lista1, lista2):
   # casos base
   if len(lista1)==0:
      return lista2

   if len(lista2)==0:
      return lista1

   # divisão




   # conquista


   # combinação




   return solução
```

https://colab.research.google.com/

```
def merge(lista1, lista2):
   # casos base
   if len(lista1)==0:
      return lista2

   if len(lista2)==0:
      return lista1

   # divisão
   if lista1[0] < lista2[0]:


   else:


   # conquista


   # combinação




   return solução
```

https://colab.research.google.com/

```python
def merge(lista1, lista2):
    # casos base
    if len(lista1)==0:
        return lista2

    if len(lista2)==0:
        return lista1

    # divisão
    if lista1[0] < lista2[0]:
        lista1_subproblema = lista1[1:] # sem o primeiro elemento
        lista2_subproblema = lista2
    else:
        lista1_subproblema = lista1
        lista2_subproblema = lista2[1:] # sem o primeiro elemento

    # conquista


    # combinação




    return solução
```

https://colab.research.google.com/

```python
def merge(lista1, lista2):
    # casos base
    if len(lista1)==0:
        return lista2

    if len(lista2)==0:
        return lista1

    # divisão
    if lista1[0] < lista2[0]:
        lista1_subproblema = lista1[1:] # sem o primeiro elemento
        lista2_subproblema = lista2
    else:
        lista1_subproblema = lista1
        lista2_subproblema = lista2[1:] # sem o primeiro elemento

    # conquista
    aux = merge(lista1_subproblema, lista2_subproblema)

    # combinação




    return solução
```

https://colab.research.google.com/

```
[1] def merge(lista1, lista2):
        # casos base
        if len(lista1)==0:
            return lista2

        if len(lista2)==0:
            return lista1

        # divisão
        if lista1[0] < lista2[0]:
            lista1_subproblema = lista1[1:] # sem o primeiro elemento
            lista2_subproblema = lista2
        else:
            lista1_subproblema = lista1
            lista2_subproblema = lista2[1:] # sem o primeiro elemento

        # conquista
        aux = merge(lista1_subproblema, lista2_subproblema)

        # combinação
        if lista1[0] < lista2[0]:
            solução = [lista1[0]] + aux
        else:
            solução = [lista2[0]] + aux

        return solução
```

https://colab.research.google.com/

```
[1] def merge(lista1, lista2):
        # casos base
        if len(lista1)==0:
            return lista2

        if len(lista2)==0:
            return lista1

        # divisão
        if lista1[0] < lista2[0]:
            lista1_subproblema = lista1[1:] # sem o primeiro elemento
            lista2_subproblema = lista2
        else:
            lista1_subproblema = lista1
            lista2_subproblema = lista2[1:] # sem o primeiro elemento

        # conquista
        aux = merge(lista1_subproblema, lista2_subproblema)

        # combinação
        if lista1[0] < lista2[0]:
            solução = [lista1[0]] + aux
        else:
            solução = [lista2[0]] + aux

        return solução
```

```
merge([2, 4, 9, 11], [8, 13, 17, 27])
```

```
[2, 4, 8, 9, 11, 13, 17, 27]
```

https://colab.research.google.com/

```
[4] def merge(lista1, lista2):
        # casos base
        if len(lista1)==0:
            return lista2

        if len(lista2)==0:
            return lista1

        # divisão, conquista e combinação
        if lista1[0] < lista2[0]:
            return [lista1[0]] + merge(lista1[1:], lista2)
        else:
            return [lista2[0]] + merge(lista1, lista2[1:])
```

```
merge([2, 4, 9, 11], [8, 13, 17, 27])
```

```
[2, 4, 8, 9, 11, 13, 17, 27]
```

https://colab.research.google.com/

```
[10] def merge(lista1, lista2):
       # casos base
       if len(lista1)==0:
          return lista2

       if len(lista2)==0:
          return lista1

       # divisão, conquista e combinação
       if lista1[-1] > lista2[-1]:
          return merge(lista1[:-1], lista2) + [lista1[-1]]
       else:
          return merge(lista1, lista2[:-1]) + [lista2[-1]]
```

```
merge([2, 4, 9, 11], [8, 13, 17, 27])
```

```
[2, 4, 8, 9, 11, 13, 17, 27]
```

https://ic.unicamp.br/~mc102/aulas/aula13.pdf

```
1  def merge(lista1, lista2):
2    if lista1 == []:
3      return lista2
4
5    if lista2 == []:
6      return lista1
7
8    if lista1[0] < lista2[0]:
9      aux = merge(lista1[1:], lista2)
10     aux = [lista1[0]] + aux
11   else:
12     aux = merge(lista1, lista2[1:])
13     aux = [lista2[0]] + aux
14
15   return aux
```

https://ic.unicamp.br/~mc102/aulas/aula13.pdf

- Uma implementação mais eficiente:

```
1  def merge(lista1, lista2):
2    if lista1 == []:
3      return lista2
4
5    if lista2 == []:
6      return lista1
7
8    if lista1[-1] > lista2[-1]:
9      aux = merge(lista1[:-1], lista2)
10     aux = aux + [lista1[-1]]
11   else:
12     aux = merge(lista1, lista2[:-1])
13     aux = aux + [lista2[-1]]
14
15   return aux
```

Exemplo 23

https://ic.unicamp.br/~mc102/listas/lista6.pdf

Faça uma função recursiva que calcule a média dos elementos de uma lista de números.

https://colab.research.google.com/

```
[15] def media(lista):
       # casos base
       if len(lista)==1:
          return lista[0]

       # divisão
       meio = len(lista) // 2
       lista1 = lista[:meio]
       lista2 = lista[meio:]

       # conquista
       aux1 = media(lista1)
       aux2 = media(lista2)

       # combinação
       solução = (aux1*len(lista1) + aux2*len(lista2))/len(lista)

       return solução
```

```
[16] media([1,2])
```

```
1.5
```

```
media([1,1,1,1,1,7])
```

```
2.0
```

Exemplo 24

https://ic.unicamp.br/~mc102/listas/lista6.pdf

Escreva uma função recursiva que calcule o somatório dos n primeiros valores da série harmônica. A série harmônica é definida como:

$$\sum_{k=1}^{\infty} \frac{1}{k} = 1 + \frac{1}{2} + \frac{1}{3} + \ldots$$

https://colab.research.google.com/

```
[20] def harmonica(n):
       # caso base
       if n == 1:
          return 1

       # divisão
       sub =  n-1

       # conquista
       aux = harmonica(sub)

       # combinação
       solução = aux + 1/n

       return solução
```

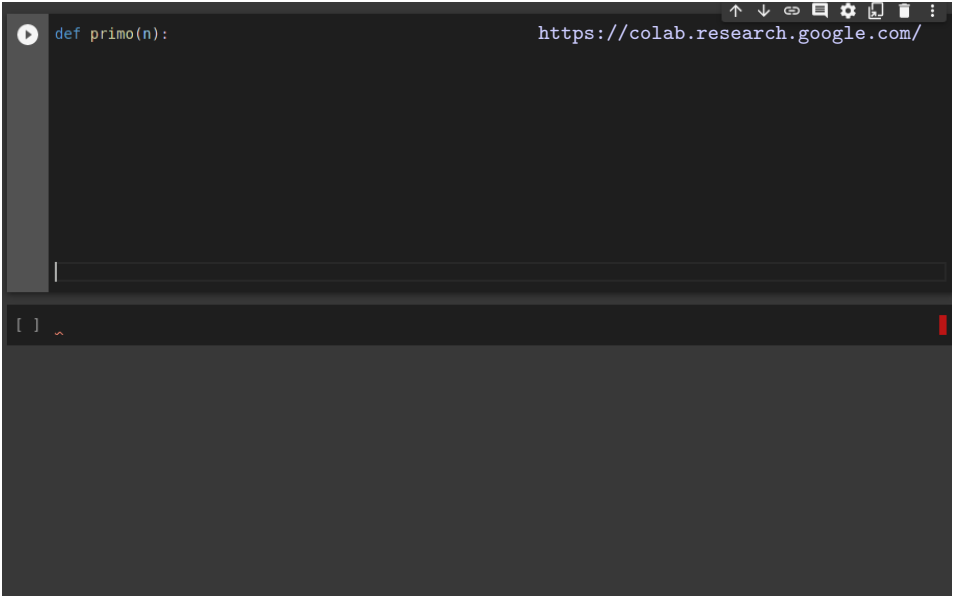
```
harmonica(100)
```

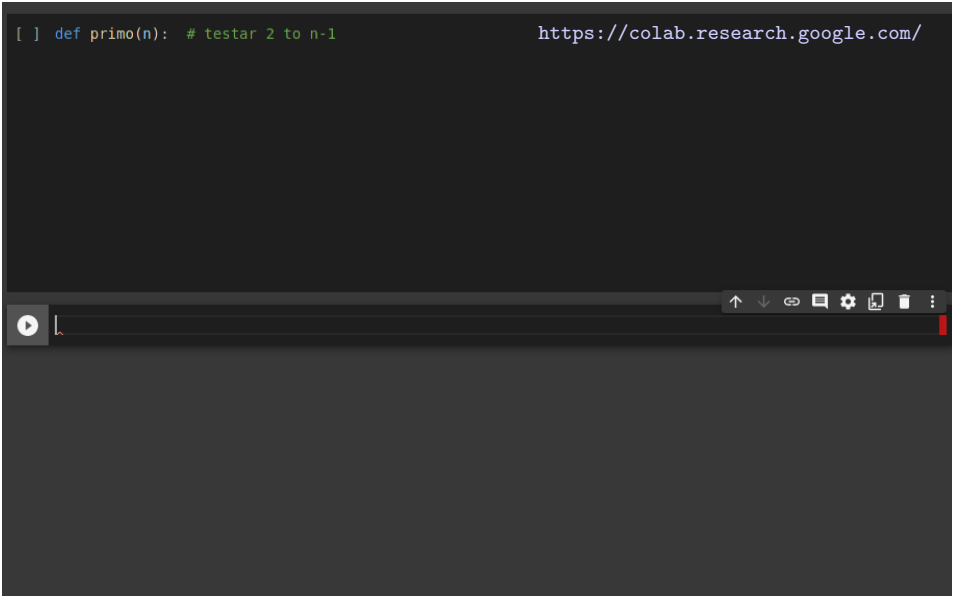
```
5.187377517639621
```

Exemplo 25

https://ic.unicamp.br/~mc102/listas/lista6.pdf

Escreva uma função recursiva que determine se um número inteiro n é primo.


def primo(n):
https://colab.research.google.com/
[ ]


[ ] def primo(n): # testar 2 to n-1
https://colab.research.google.com/

https://colab.research.google.com/

```
[52] def primo(n, testador=2):  # testar 2 to n-1
       # casos base
       if n < 2:
         return 0
       if n == 2:
         return 1
       if testador == n:
         return 1
       if n % testador == 0:
         return 0

       # Divisão
       sub = testador + 1

       # Conquista
       aux = primo(n, sub)

       # Combinação: nada a fazer
       return aux
```

```
[53] print(primo(3))
```

```
1
```

```
print([x for x in range(100) if primo(x)==1 ])
```

```
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97]
```

https://colab.research.google.com/

```
[64] def primo(n, testador=2):  # testar 2 to n-1
       # casos base
       if n < 2:
         return False
       if n == 2:
         return True
       if testador*testador > n:
         return 1
       if n % testador == 0:
         return False

       # Divisão, conquista e combinação
       return primo(n, testador+1)
```

```
print([x for x in range(100) if primo(x) ])
```

```
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97]
```

Exemplo 26

https://ic.unicamp.br/~mc102/listas/lista6.pdf

Escreva uma função recursiva que calcule $\lfloor \lg n \rfloor$, ou seja, o piso do logaritmo de n na base 2. Por exemplo, $\lfloor \lg 100 \rfloor = 6$.

https://colab.research.google.com/

```
[72] def piso_log(n):
       # caso base
       if n < 2:
         return 0

       # divisão, conquista e combinação
       return piso_log( n // 2 ) + 1
```

```
[73] piso_log(100)
```

```
6
```

```
print([piso_log(x) for x in range(1,20)])
```

```
[0, 1, 1, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 4, 4, 4, 4]
```

Exemplo 27

https://ic.unicamp.br/~mc102/listas/lista6.pdf

8. Suponha que uma matriz binária quadrada `M` represente a ligação entre um conjunto de n cidades. Desta forma, `M[i][j] = 1` indica que existe uma estrada da cidade `i` para a cidade `j`, e `M[i][j] = 0`, caso contrário. Por exemplo, na matriz abaixo temos que a cidade 0 possui estradas para as cidades 1 e 2, já a cidade 1 possui estrada apenas para a cidade 2. Note que existe uma estrada saindo da cidade 0 em direção à cidade 1, mas não há estrada saindo da cidade 1 em direção à cidade 0, isso porque nem sempre uma estrada que liga duas cidades possui vias de ida e volta.

```
0 1 1 0
0 0 1 0
1 1 0 1
1 0 1 0
```

Escreva uma função recursiva que, dada uma matriz `M` e uma cidade `i`, determine todas as cidades que podem ser alcançadas a partir de `i`.

https://colab.research.google.com/

```
[29] M=[[1,0,1,0],
       [0,0,1,0],
       [1,0,0,1],
       [1,0,0,0]]
```

```
[31] i = 2
     E = [x for x in range(len(M)) if M[i][x]==1]
     print(E)
```

```
     [0, 3]
```

```
[32] for i in range(4):
       E = [x for x in range(len(M)) if M[i][x]==1]
       print(E)
```

```
     [0, 2]
     [2]
     [0, 3]
     [0]
```

https://colab.research.google.com/

```
[29] M=[[1,0,1,0],
       [0,0,1,0],
       [1,0,0,1],
       [1,0,0,0]]
```

```
[31] i = 2
     E = [x for x in range(len(M)) if M[i][x]==1]
     print(E)
```

```
[0, 3]
```

```
for i in range(4):
  A=4*[False]
  E = [x for x in range(len(M)) if M[i][x]==1 and not A[x]]
  print(E)
```

```
[0, 2]
[2]
[0, 3]
[0]
```

```
[ ]
```

https://colab.research.google.com/

```
[20] def cidades(M, A, i):
       E = [x for x in range(len(M)) if M[i][x]==1 and not A[x] ]

       # caso não base
       if len(E) > 0:
         for k in E:
            A[k] = True
            cidades(M, A, k)
```

```
[21] M=[[0,1,0,0],
        [1,0,0,0],
        [1,0,0,1],
        [1,0,0,0]]
```

```
for i in range(4):
  A=4*[False]
  cidades(M, A, i)
  print(A)
```

```
[True, True, False, False]
[True, True, False, False]
[True, True, False, True]
[True, True, False, False]
```

https://colab.research.google.com/

```
[20] def cidades(M, A, i):
       E = [x for x in range(len(M)) if M[i][x]==1 and not A[x] ]

       # caso não base
       if len(E) > 0:
         for k in E:
            A[k] = True
            cidades(M, A, k)
```

```
[23] M=[[0,1,1,0],
        [0,0,1,0],
        [1,1,0,1],
        [1,0,1,0]]
```

```
for i in range(4):
  A=4*[False]
  cidades(M, A, i)
  print(A)
```

```
[True, True, True, True]
[True, True, True, True]
[True, True, True, True]
[True, True, True, True]
```

https://colab.research.google.com/

```
[20] def cidades(M, A, i):
       E = [x for x in range(len(M)) if M[i][x]==1 and not A[x] ]

       # caso não base
       if len(E) > 0:
         for k in E:
            A[k] = True
            cidades(M, A, k)
```

```
[27] M=[[0,0,1,0],
        [0,0,1,0],
        [1,0,0,1],
        [1,0,0,0]]
```

```
for i in range(4):
  A=4*[False]
  cidades(M, A, i)
  print(A)
```

```
[True, False, True, True]
[True, False, True, True]
[True, False, True, True]
[True, False, True, True]
```

Perguntas ....

Referências

- Zanoni Dias, MC102, Algoritmos e Programação de Computadores, IC/UNICAMP, 2021. https://ic.unicamp.br/~mc102/

  Aula Introdutória [slides] [vídeo]
  Primeira Aula de Laboratório [slides] [vídeo]
  Python Básico: Tipos, Variáveis, Operadores, Entrada e Saída [slides] [vídeo]
  Comandos Condicionais [slides] [vídeo]
  Comandos de Repetição [slides] [vídeo]
  Listas e Tuplas [slides] [vídeo]
  Strings [slides] [vídeo]
  Dicionários [slides] [vídeo]
  Funções [slides] [vídeo]
  Objetos Multidimensionais [slides] [vídeo]
  Algoritmos de Ordenação [slides] [vídeo]
  Algoritmos de Busca [slides] [vídeo]
  Recursão [slides] [vídeo]
  Algoritmos de Ordenação Recursivos [slides] [vídeo]
  Arquivos [slides] [vídeo]
  Expressões Regulares [slides] [vídeo]
  Execução de Testes no Google Cloud Shell [slides] [vídeo]
  Numpy [slides] [vídeo]
  Pandas [slides] [vídeo]

- Panda - Cursos de Computação em Python (IME -USP) https://panda.ime.usp.br/

  Como Pensar Como um Cientista da Computação https://panda.ime.usp.br/pensepy/static/pensepy/
  Aulas de Introdução à Computação em Python https://panda.ime.usp.br/aulasPython/static/aulasPython/

- Fabio Kon, Introdução à Ciência da Computação com Python http://bit.ly/FabioKon/
- Socratica, Python Programming Tutorials http://bit.ly/SocraticaPython/
- Google - online editor for cloud-native applications (Python programming) https://shell.cloud.google.com/
- w3schools - Python Tutorial https://www.w3schools.com/python/
- Outros, citados nos Slides.