INSTITUTO FEDERAL
São Paulo
Câmpus São Carlos

AP1S1 – Algoritmos e Programação

Recursividade

Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas

INSTITUTO FEDERAL
São Paulo
Câmpus São Carlos

Recursão

- **Recursão** é um método de resolução de problemas que envolve quebrar um problema em subproblemas menores e menores até chegar a um problema pequeno o suficiente para que ele possa ser resolvido trivialmente

- Normalmente recursão envolve uma função que chama a si mesma

- Embora não seja fácil perceber, a recursão nos permite escrever soluções elegantes para problemas que, de outra forma, podem ser muito difíceis de programar

Recursão – Entendendo com exemplo

- Suponha que você deseja calcular a soma de uma lista de números, tais como: [1,3,5,7,9]

- Implementação conhecida:

```
def listsum(numList):
    theSum = 0
    for i in numList:
        theSum = theSum + i
    return theSum

print(listsum([1,3,5,7,9]))
```

Recursão – Entendendo com exemplo

- Como você calcularia a soma de uma lista de números se não tivesse os laços while ou for?
- Se você fosse um matemático poderia começar recordando que a adição é uma função definida para dois parâmetros, um par de números
- Para redefinir o problema da adição de uma lista para a adição de pares de números, podemos reescrever a lista como uma expressão totalmente entre parênteses. Tal expressão poderia ser algo como:

(((((1+3)+5)+7)+9)

- Poderíamos também colocar os parênteses na ordem reversa,

(1+(3+(5+(7+9))))

Recursão – Entendendo com exemplo

- Como podemos usar essa ideia e transformá-la em um programa Python?

- Em primeiro lugar, vamos reformular o problema soma em termos de listas de Python

- Poderíamos dizer que a soma da lista numList é a soma do primeiro elemento da lista (numList [0]), com a soma dos números no resto da lista (numList [1:])

- De forma funcional podemos escrever:

listSum(numList)=first(numList)+listSum(rest(numList))

Recursão – Entendendo com exemplo

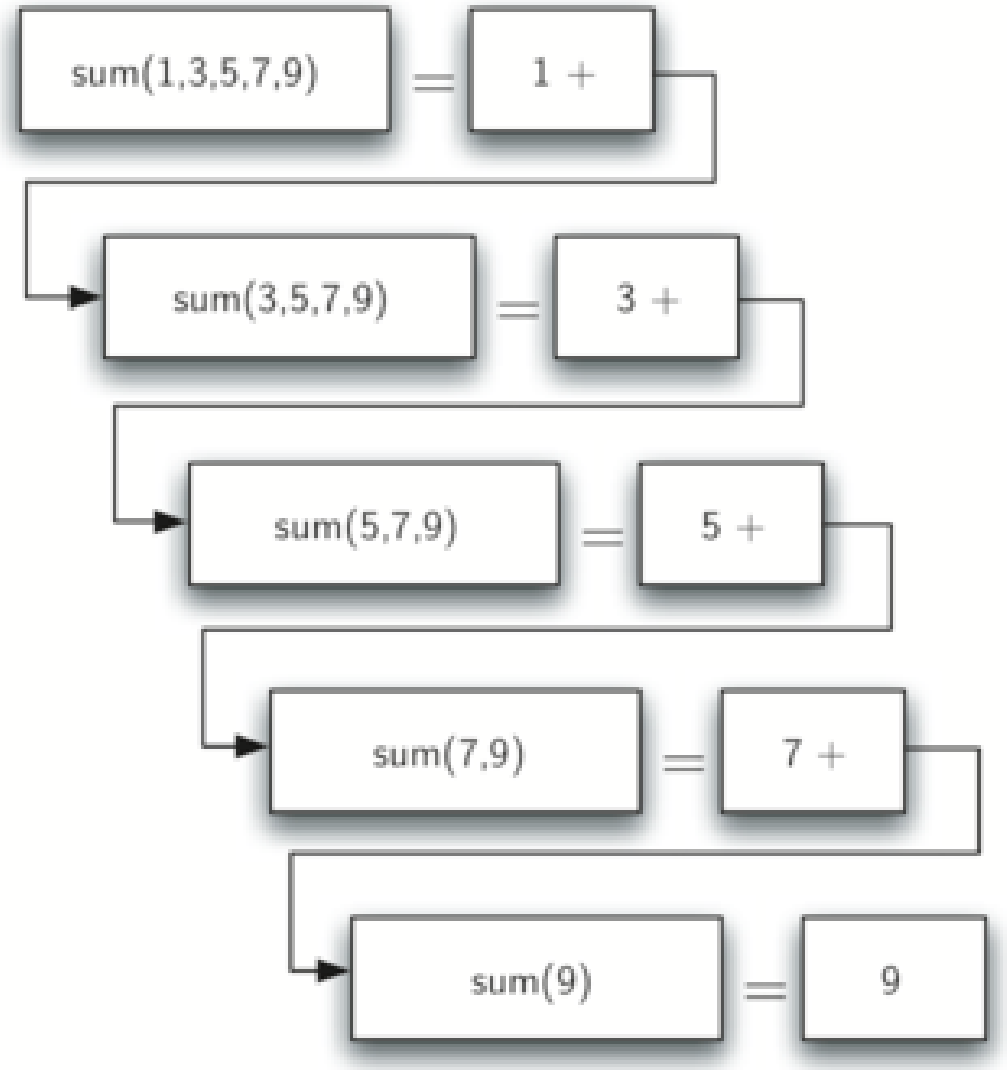
- A implementação ficaria assim:

```
def listsum(numList):
    if len(numList) == 1:
        return numList[0]
    else:
        return numList[0] + listsum(numList[1:])

print(listsum([1,3,5,7,9]))
```

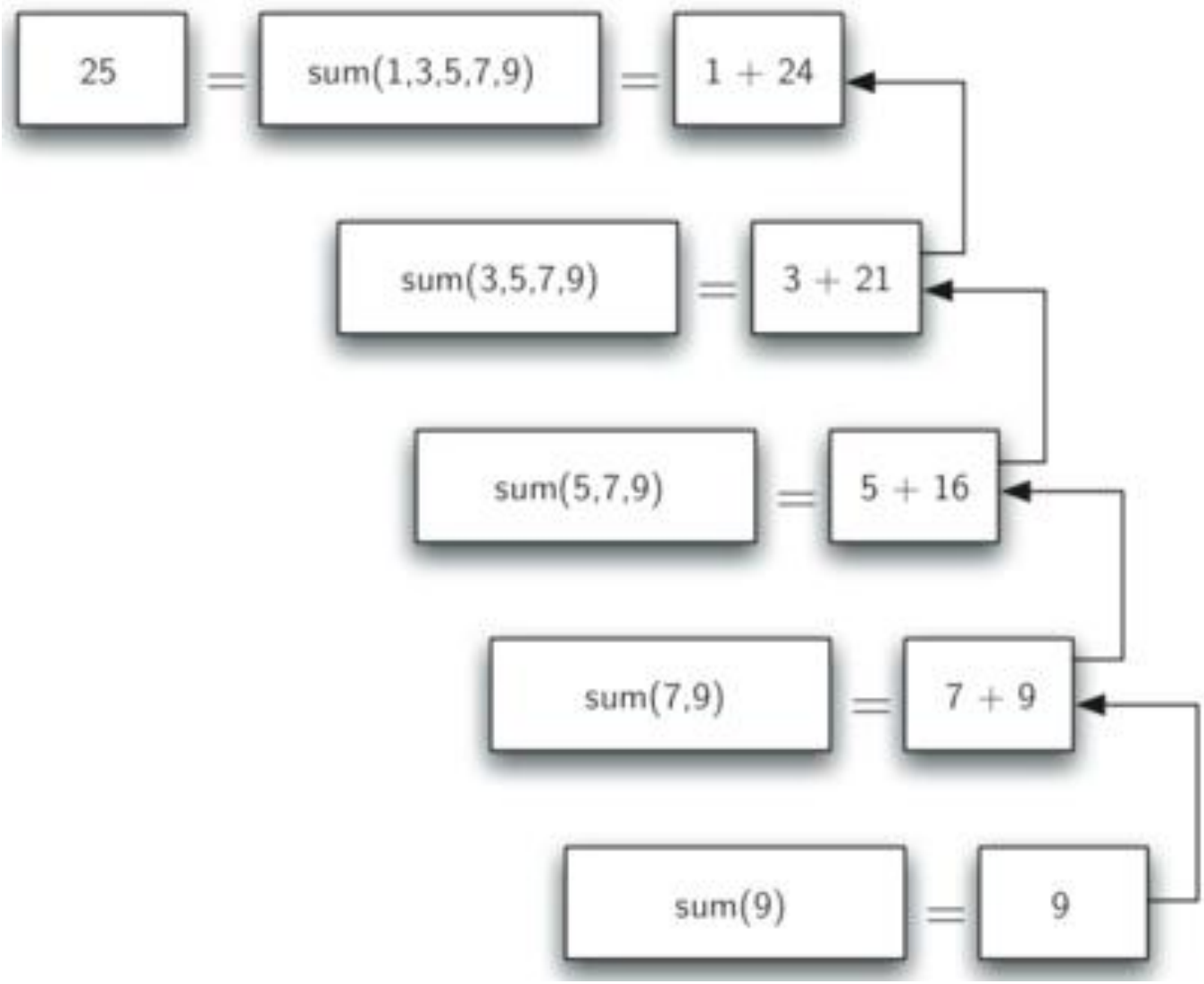
Recursão – Entendendo com exemplo

- A execução ficaria assim:



Recursão – Entendendo com exemplo

- Retornos:



Recursão – As três leis

- Todos os algoritmos recursivos devem obedecer a três leis importantes:
  - Um algoritmo recursivo deve ter um *caso básico*
  - Um algoritmo recursivo deve mudar o seu estado e se aproximar do caso básico
  - Um algoritmo recursivo deve chamar a si mesmo, recursivamente

Recursão – As três leis

- Exemplo: Encontre a definição recursiva de:

1. x * y
2. 3 * 4 = 3 + (3*3)
3. 3 * 3 = 3 + (3*2)
4. 3 * 2 = 3 + (3*1)
5. 3 * 1 = 3 + (3*0)
6. 3 * 0 = 0

Caso Trivial? x*0 = 0

Passo Recursivo? x* y = x + x*(y-1)

$$x * y = \begin{cases} 0, y = 0 \\ x + x * (y-1), y > 0 \end{cases}$$

Exemplo – Sequência de Fibonacci

- Sequência de Fibonacci: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34...
- Os elementos iniciais são: 0 e 1
- Os demais elementos são definidos em função da soma dos dois elementos imediatamente anteriores

Exemplo – Sequência de Fibonacci

```
def termo_fibonacci(n):
    if n <= 1:
        return n
    else:
        return termo_fibonacci(n-1) + termo_fibonacci(n-2)


nro=int(input('Digite o termo da sequência de Fibonacci: '))
print('O ',nro,'° termo de Fibonacci é: ',
termo_fibonacci(nro))
```

Exemplo – Sequência de Fibonacci

```
def sequencia_fibonacci(a, b, n):
    if a < n:
        print(a," ",end="")
    if b < n:
        sequencia_fibonacci(b, a + b, n)


nro=int(input('Digite o valor de parada da sequência de Fibonacci: '))
sequencia_fibonacci(0,1,nro)
```

Exercícios

1. Escreva um algoritmo recursivo para calcular x^y

2. Escreva um algoritmo recursivo para calcular o quociente da divisão inteira
3. Escreva um algoritmo recursivo para calcular o resto da divisão inteira
4. Escreva um algoritmo recursivo para calcular o fatorial de um número
5. Escreva um algoritmo recursivo para retornar a quantidade de caracteres de uma string
6. Crie uma função recursiva que exiba verticalmente uma string

Exercícios

7. Crie uma função recursiva para exibir os algarismos de um número positivo de forma invertida. Exemplo: 326

   6

   2

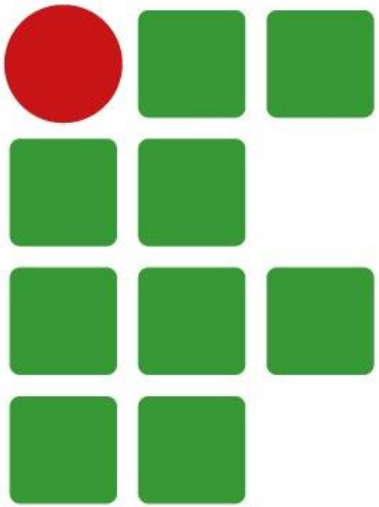
   3

8. Crie uma função recursiva para exibir os algarismos de um número positivo na ordem correta. Exemplo: 326

   3

   2

   6



INSTITUTO FEDERAL
São Paulo
Câmpus São Carlos