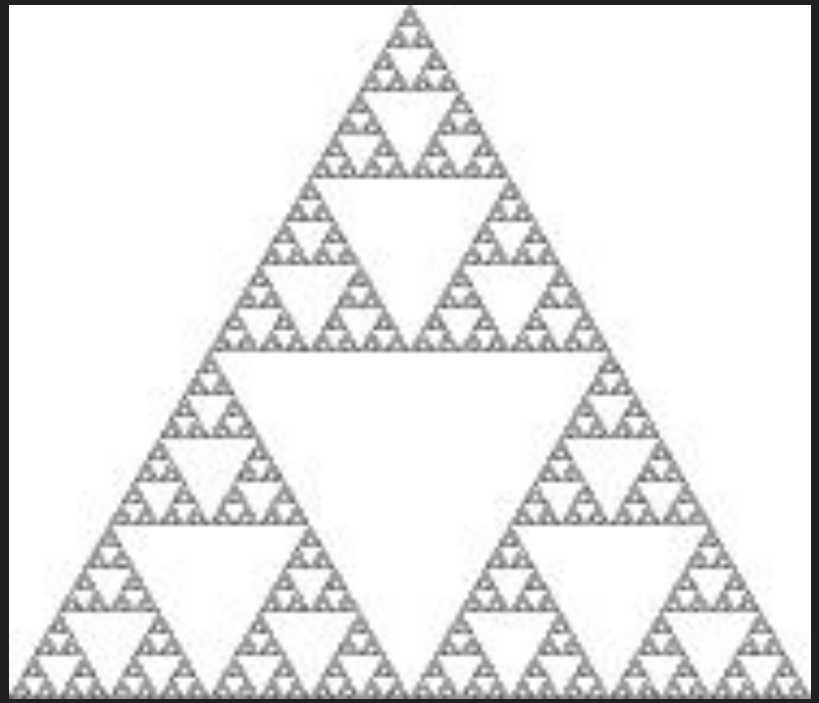
SSC0800 - Introdução à Ciência de Computação I

Recursão

Prof.: Leonardo Tórtoro Pereira
leonardop@usp.br

O que vamos aprender hoje?

Recursão



Recursão [1]

- É quando uma função chama a ela mesma
  - Essa chamada pode ser direta ou indireta
- Pode resolver com facilidade alguns problemas
  - Mas, para outros, um laço de iteração é melhor
- Normalmente é dividida em um caso base, que retorna um valor para a solução, e o caso maior, que é expressado em função de problemas menores

Recursão [1]

```
def recursiveFunction(input):
    print(input)
    recursiveFunction(input-1)


recursiveFunction(5)
```

Recursão [1]

- Quando o caso base não é alcançado, pode retornar *stack overflow*

Recursão [1]

```
def recursiveFunction(input):
  if input <= 0:
    return
  print(input)
  recursiveFunction(input-1)


recursiveFunction(5)
```

Recursão [1]

- O método de recursão costuma ser usado quando a solução depende da solução de instâncias menores do mesmo problema

Recursão [1]

- Alguns problemas bem conhecidos que podemos usar recursão:
  - Fatorial
  - Sequência de Fibonacci (e outras sequências no geral)
  - Busca binária
  - Maior divisor comum
  - Torre de Hanoi
  - ...

Fatorial

Recursão [1]

```
def fact(n):
    if n <= 1:
        return 1
    else:
        answer = n*fact(n-1)
        return answer

print(fact(5))
```

Fibonacci

Recursão [1]

```
def fibonacci(n):
    if n == 0:
        return 0
    if n == 1 or n == 2:
        return 1
    else:
        return fibonacci(n - 1) + fibonacci(n - 2)
NUMBER = 10
for i in range(NUMBER):
  print(fibonacci(i))
```

Maior Divisor Comum

Recursão [1]

```
def mdc(x, y):
    if y == 0:
        return x
    else:
        return mdc(y, x % y)

print(mdc(4, 6))
```

Busca Binária

Recursão [1]

```
def binary_search(begin, end, key, vector):
    middle = (begin + end) // 2
    if begin > end:
        return NOT_FOUND
    if key == vector[middle]:
        return middle;
    elif key < vector[middle]:
        return binary_search(begin, middle-1, key, vector);
    elif key > vector[middle]:
        return binary_search(middle+1, end, key, vector)
    return NOT_FOUND

NOT_FOUND = -1
vetor = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
print(binary_search(0, 9, 10, vetor))
```

Exemplos

Referências

1. https://www.geeksforgeeks.org/introduction-to-recursion-data-structure-and-algorithm-tutorials/
2. https://www.learnpython.org/
3. https://www.w3schools.com/python/
4. https://panda.ime.usp.br/cc110/static/cc110/index.html
5. https://www.youtube.com/playlist?list=PLcoJJSvnDgcKpOi_UeneTNTIVOigRQwcn