香港中文大學（深圳）
The Chinese University of Hong Kong, Shenzhen

INTRODUCTION TO COMPUTER SCIENCE: PROGRAMMING METHODOLOGY

TUTORIAL 11

RECURSIVE ALGORITHM

Frederick Khasanto
122040014

16 November 2023

Structure of a recursive algorithm

Computing factorials:

$0! = 1;$

$n! = n \times (n - 1)!; \; n > 0$

Base case or stopping condition

Original problem **Subproblem** **Simplest case**

```
# Return the factorial for the specified number
def factorial(n):
    if n == 0: # Base case
        return 1
    else:
        return n * factorial(n - 1) # Recursive call
```

Recursive call tree-I

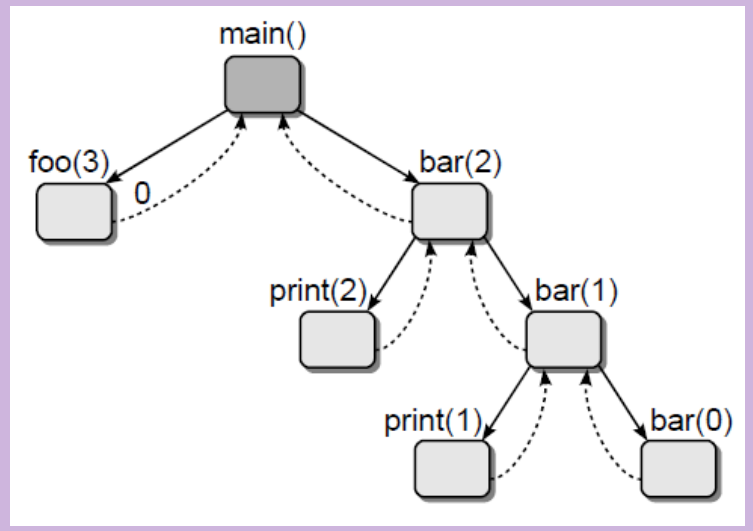
- The recursive call tree consists of small boxes and directed edges between boxes. Each box represents a function call and is labeled with the name of the function and the actual arguments passed to the function when it was invoked.
- The **directed edges** between the boxes indicate the flow of execution. The **solid edges** indicate the function from which a call originated. The **dashed edges** indicate function returns and are labeled with the return value if a value is returned to the caller.
- The edges are listed **left to right** in the order the calls are made.

```
# A sample program containing three functions.
def main():
  y = foo( 3 )
  bar( 2 )

def foo( x ):
  if x % 2 != 0 :
    return 0
  else :
    return x + foo( x-1 )

def bar( n ):
  if n > 0 :
    print( n )
    bar( n-1 )

main()
```



Questions:

1. What are the outputs?

2. What is the value of y?

3. What is y if you pass an even number to foo()?

Recursive call tree-II

- **More examples:**

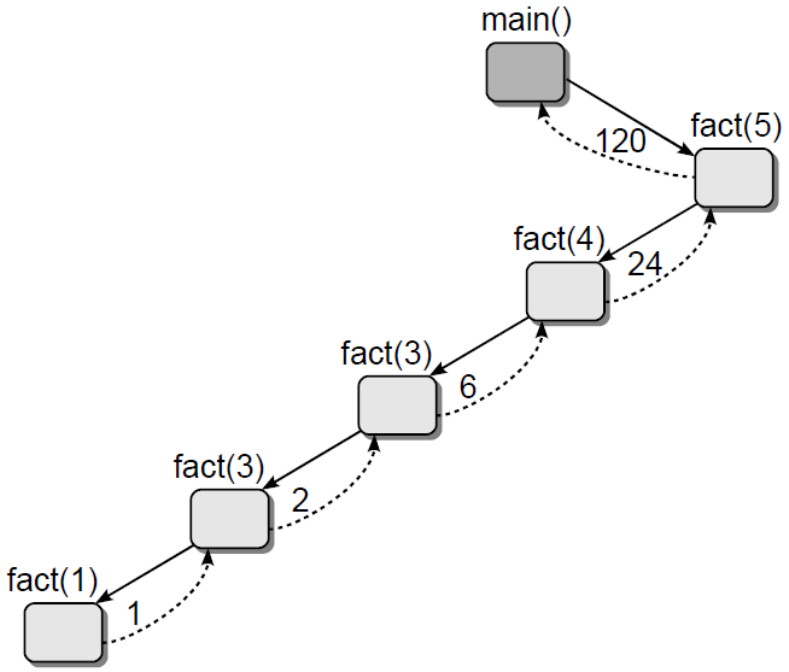


Figure 10.2: Recursive call tree for `fact(5)`.

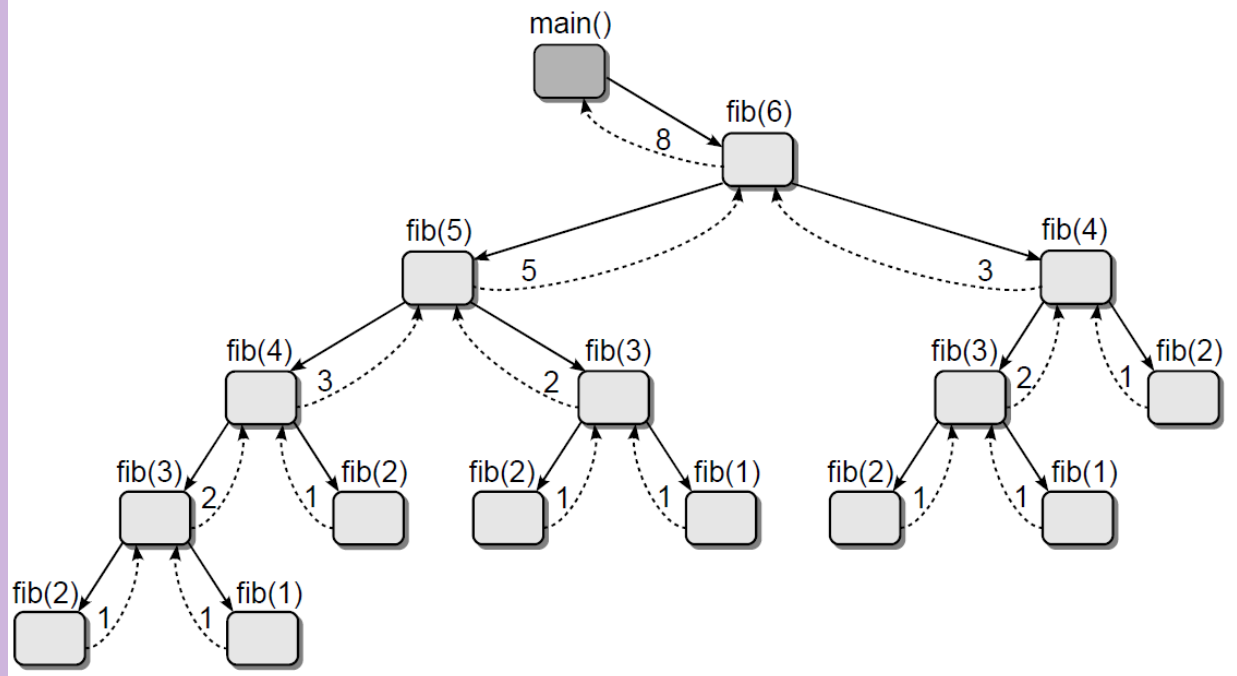


Figure 10.4: Recursive call tree for `fib(6)`.

- **Reference book: «Data Structures and Algorithms Using Python» by Rance D.Necaise, Chapter 10**

Q1: Number system conversion

- i) Write a program that define a recursive function def conv(n, base) that can print out the binary(base=2) or octal(base=8) version of the decimal number n.
- ii) Modify the program a little bit to provide a way for solving problem Q2 in Assignment 1. What about the reverse order?
- iii) Try to draw the recursive call trees of the three programs respectively if the arguments are n=12,base=2 for (i) and n=3125 for (ii).

Q2: All Permutations of a string

- Write a recursive program to print out all the possible permutations given a string with distinct characters. A sample run is as following. Draw the recursive call tree for this example.

```
Enter a string:abc
abc
acb
bac
bca
cab
cba
```

Q3: Judge palindrome

- Write a recursive program to define a function to judge whether it is a palindrome or not. If a string is equal to its reverse, then it is a palindrome. For example, "*level*" and "*madam*" are palindromes.

Q4: Rearrange a list

- Write a recursive algorithm to rearrange a list of integer values so that all the even values appear before all the odd values. A sample run is as following. Draw the recursive tree given lst=[1,4,3,2] to be the input.

```
def main():
    lst=[1,4,3,2]
    print(rearrange(lst))

main()
```

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

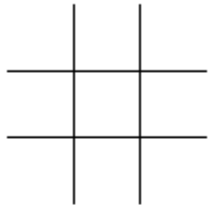
Q5: Find all the possible subsets

- Write a recursive algorithm to find all the possible subsets of a list(excluding itself). A sample run is as following. In this example the list [1,2,3] is passed to the function and a list of all subsets is returned. Draw the recursive call tree given the list [1,2,3] as an example. You should be able to rediscover following output by looking at the recursive call tree.

```
[[], [3], [2], [2, 3], [1], [1, 3], [1, 2]]
```

Q6: Playing Tic-Tac-Toe-I

Consider the game of tic-tac-toe in which two players use a board containing nine squares organized into three rows of three columns:



The two players take turns placing tokens of `X`s and `O`s in the squares. One player is assigned the `X`s while the other is assigned the `O`s. Play continues until all of the squares are filled, resulting in a draw, or one of the players wins by aligning three identical pieces vertically, diagonally, or horizontally. The following diagrams show three different game boards, two resulting in wins and one resulting in a draw:

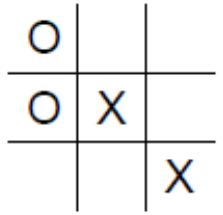
O		X
O	X	
X		

O	X	X
	O	X
		O

X	O	O
O	X	X
X	X	O

Q6: Playing Tic-Tac-Toe-II

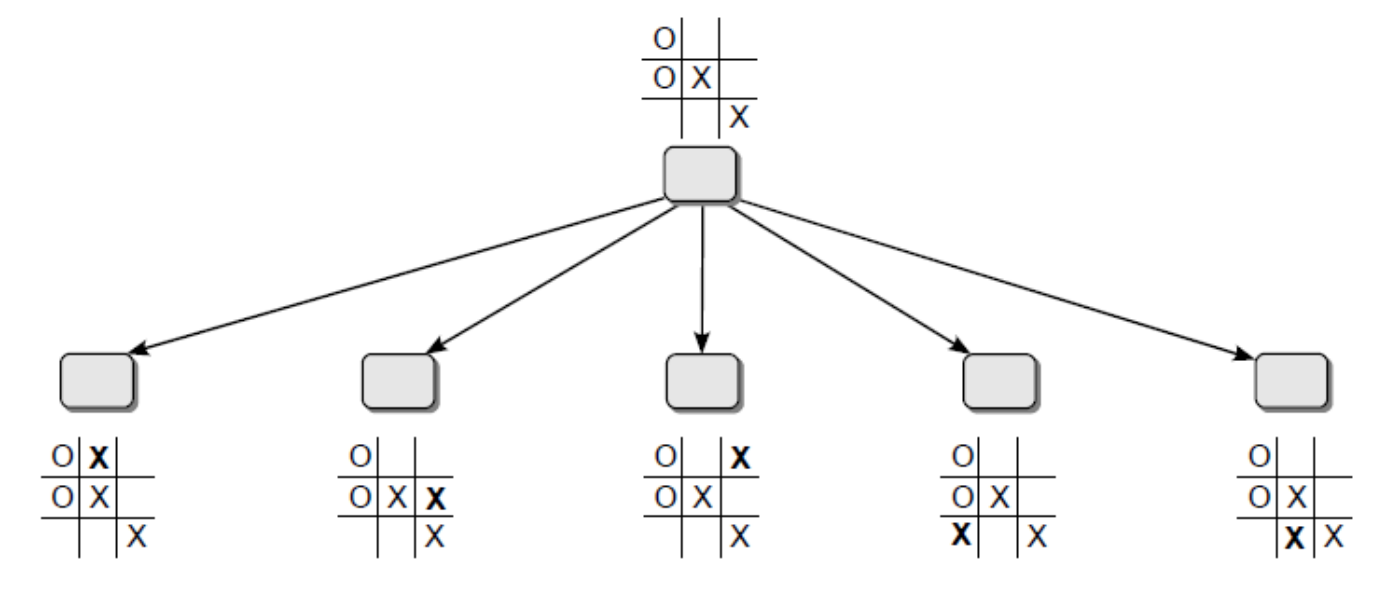
Suppose you are playing a game of tic-tac-toe in which four moves have been played as follows:



and now it's X's turn to play, which happens to be the computer. The computer needs to evaluate all of its possible moves to determine the best move to make, which it can do by evaluating the game tree starting from this point in the game. It can use recursion and build a recursive call tree to represent all possible moves in the game for both itself and its opponent. During the recursion, the tokens are placed on the board for both the computer and its opponent as if they were both actually playing. As the recursion unwinds, the tokens are picked to return the game to its current state. This game tree shows the five possible moves the computer can make at this point:

Q6: Playing Tic-Tac-Toe-III

"The game tree"



- **Try to design a recursive algorithm for computer to determine where it should move. And write a program to allow you and computer to compete and to see whether you could win or not. Either ◯(moves first) or × could be chosen.**

The computer would need to evaluate all of these moves to determine which would be the best. The decision would be based on which move would allow it to win before its opponent. The next figure shows the part of the game tree that is constructed while evaluating the placement of an X in the upper-right square.