ITI1120
Labo10: Recursion

1

Objectives

- Recursion:
  - Examples of recursive algorithms and their conversion in Python
  - Exercises

2

Recursion – Simple Example

- A recursive algorithm that counts the number of *digits* in a non-negative integer N.
  - Example: if N = 34567, then the result is 5.
  - If = 1234567890, then the result is 10.

3

Recursive – Algorithm

DATA: N *(a non-negative integer)*

INTERMEDIARY: DigitsLeft *(digits left)*

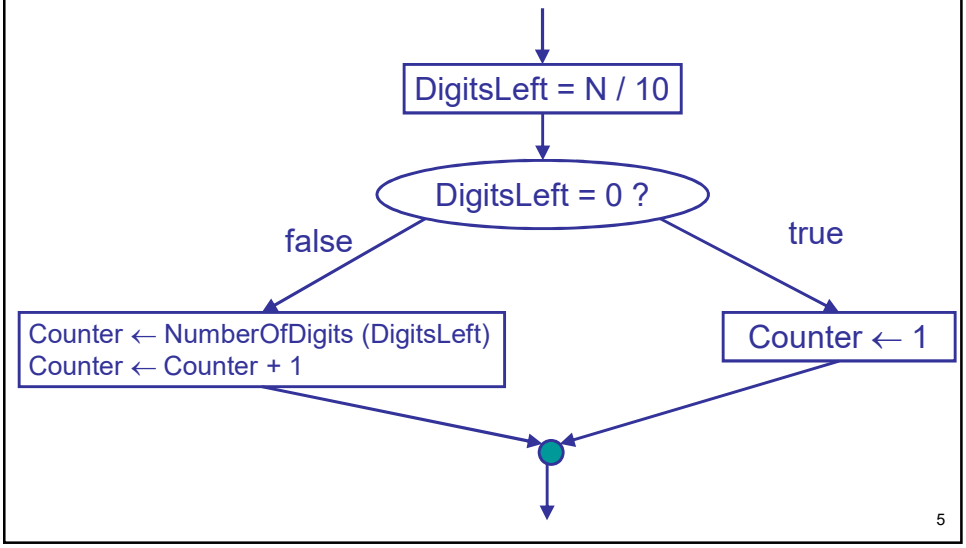
RESULT: Counter *(number of digits in N)*

HEADER: Counter ← NumberOfDigits(N)

4

Recursive – Algorithm - Suite

MODULE:



5

Trace for N = 254

6

Trace, page 2

Counter ← NumberOfDigits (DigitsLeft)

↓ 25

Counter ← NumberOfDigits (N)

7

Trace , page 3

Counter ← NumberOfDigits (DigitsLeft)

↑ 1 ↓ 2

Counter ← NumberOfDigits (N)

Ligne	N	ResteDe Chiffres	Compteur
Valeurs initiales	2	?	?
(1) ResteDeChiffres = N/10		0	
(2) ResteDeChiffres = 0 ? vrai			
(5) Compteur ← 1			1

8

Trace , page 2

Counter ← NumberOfDigits (DigitsLeft)
↑ 2 ↓ 25
Counter ← NumberOfDigits (N)

Line	N	DigitsLeft	Coun-teur
Initial values	25	?	?
(1) DigitsLeft = N / 10		2	
(2) DigitsLeft = 0 ? false			
(3) Call Counter ← NumberOfDigits (DigitsLeft)			1
(4) Compteur ← Compteur + 1			2

9

Trace, page 1

Line	N	DigitsLeft	Counter
Initial values	254	?	?
(1) DigitsLef = N / 10		25	
(2) DigitsLef = 0 ? false			
(3) Call Counter ← NumberOfDigits (DigitsLeft)			
(4) Compteur ← Compteur + 1			3

10

Explain how the provided code functions

- See NumberOfDigits.py
- We have added to that algorithm implementation display instructions to indicate where we enter or exit an invocation, and to trace the local values of `n`):
  - Immediately after the déclarations of the local variable
    ```
    1: Inputting the function with n = …
    ```
  - Just before the call to the recursive function:
    ```
    2: recursive call cominf from n = …
    ```
  - Just after the call of the recursive function:
    ```
    3: Coming back from a recursive  with n = …
    ```
  - Just after the command "return"
    ```
    4: Coming back from the function with n = …,
       counter = …
    ```
  - In the base case
    ```
    5: Base case with n = …
    ```

11

2nd simple example

- Derive a recursive algorithm that verifies if a list of integers A is sorted in increasing/decreasing order.
  - Note: *increasing is* different from strictly increasing (where two elements cannot be equal)
- The size of A is larger or equal to 2.
- Examples:
  - A = [3, 6, 8, 5, 9]: False
  - A = [4, 5, 6, 6, 9, 14]: True

12

2nd example - solution

DATA:
- A — *(liste of integers)*
- N — *(size of the list A)*

RESULT:
- Sorted — *(Boolean: true if A is sorted)*

INTERMEDIARY:
- SortedSmaller — *(Boolean: true if a smaller A is sorted)*

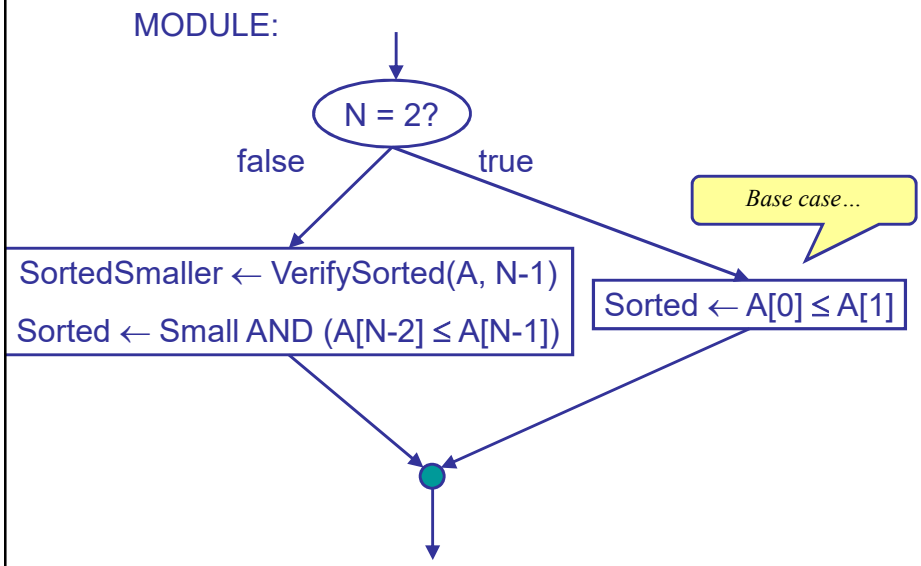
HEADER:
Sorted ← VerifySorted(A,N)

HYPOTHESYS: N ≥ 2

13

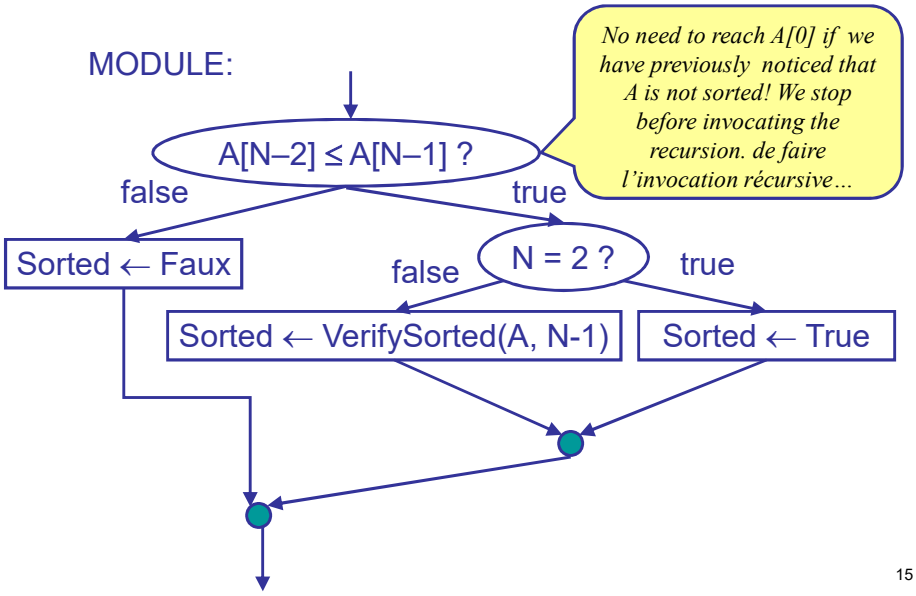
2nd example – simpe solution

MODULE:



14

2nd example – efficient solution



15

Example 2 – Python Program

- See verifySort.py
- Examine the conversion from the algorithm in Python, execute and test it in the main program. How to make the call? (You have to provide two parameters)
- Insert some print(…) calls to trace the execution of the recursive function.

16

Exercise #1

- Derive a Python recursive function to verify if all the elements in positions 0...N-1 of a list, A, of integers are *digits (between 0 and 9)*.
- The function takes two parameters, a reference to the list and an integer that is initially the size of the list.
- How to make the call in the main program?

17

Exercise #2

- Derive a recursive Python function to **create** a list containing the values ranging from 0 to N-1.
- In the main program we need to initialize a list (L = [])

After that we must call the recursive function by providing it that list to modify it.

- The second parameter of the recursive function is a value n. The main program should ask the user to key in its value from the keyboard.

18

Exercise #3 – Euclide Algorithm

- The *Biggest Commun Divisor* (BCD) of 2 integer numbers that divide the 2 numbers with a rest of zeo.

- The Euclid algorithm to find the BCD of x and y is:

  pcd(x,y) is …

y	if $x \geq y$ and x mod y is 0
bcd(y, x)	si $x < y$
bcd(y, x mod y)	otherwise

- if $x \geq y$, thus the algorithm becomes

  pcd(x, y) is …

y	if x mod y is 0
bcd(y, x mod y)	otherwise

  It works even if $x < y$. The first recursive call exchange the order to y, x

19

Exercise #3 - Euclide Algorithm

- Derive a recursive Python function to find the *biggest Commun Divisor* (BCD) of two numbers *x* and *y*.

- Test it

  ```
  bcd(1234,4321)
  bcd(8192,192)
  ```

  …

20