SOMANATHAN SUBRAMANIYAN

STUDENT ID: G00364742

Computational Thinking with Algorithms

Problem Sheet - Answers

2019

Table of Contents

[Q1: Question 1](#_Toc3661342)

[Answer 2](#_Toc3661343)

[Q2: Question 2 3](#_Toc3661344)

[Answer 3](#_Toc3661345)

[Q2 (b) Question 3](#_Toc3661346)

[Answer 3](#_Toc3661347)

[Q2 (c) Question 4](#_Toc3661348)

[Answer 4](#_Toc3661349)

[Q2 (d) Question 5](#_Toc3661350)

[Answer 5](#_Toc3661351)

[Q3 (a) Question 6](#_Toc3661352)

[Answer: 6](#_Toc3661353)

[Q3 (b) Question 6](#_Toc3661354)

[Answer: 6](#_Toc3661355)

[Q3 (c) Question 6](#_Toc3661356)

[Answer 6](#_Toc3661357)

[Q3 (d) Question 7](#_Toc3661358)

[Answer 7](#_Toc3661359)

[Q3 (e) Question 8](#_Toc3661360)

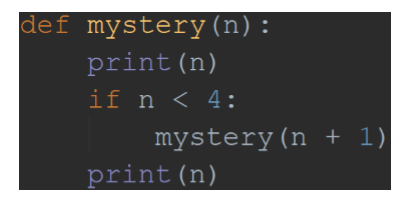
[Answer 8](#_Toc3661361)

[Q3 (f) Question 9](#_Toc3661362)

[Answer 9](#_Toc3661363)

# Q1: Question

Consider the following method:

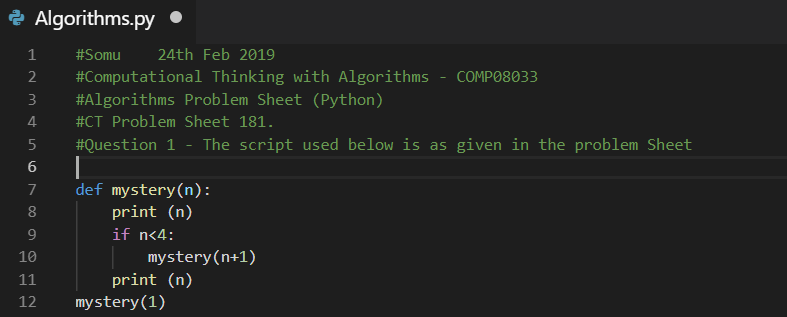
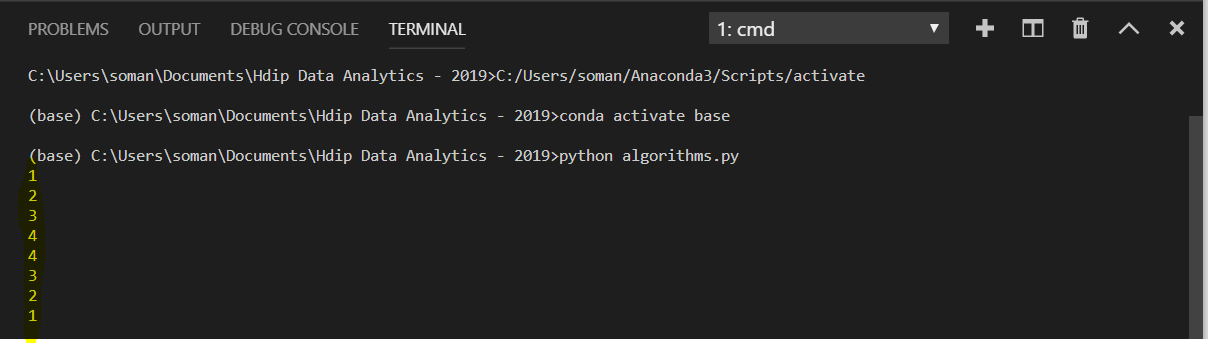


What will the output of the call mystery(1) be?

Write an explanation of the reasoning behind your answer, using the aid of either a recursion trace diagram or a stack diagram. Include any code which you write for testing or explanation purposes as part of your answer.

## Answer

Python Code: Algorithms.py execution output in python



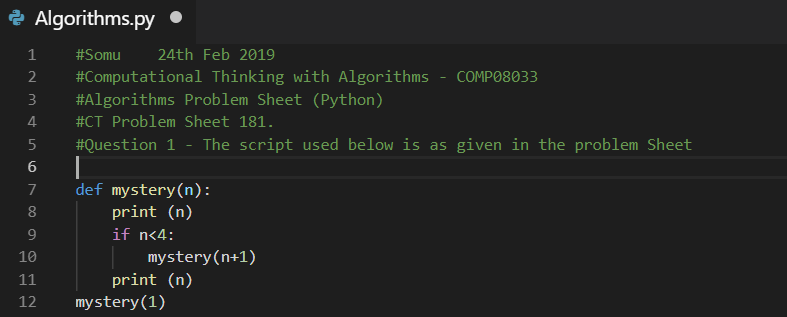
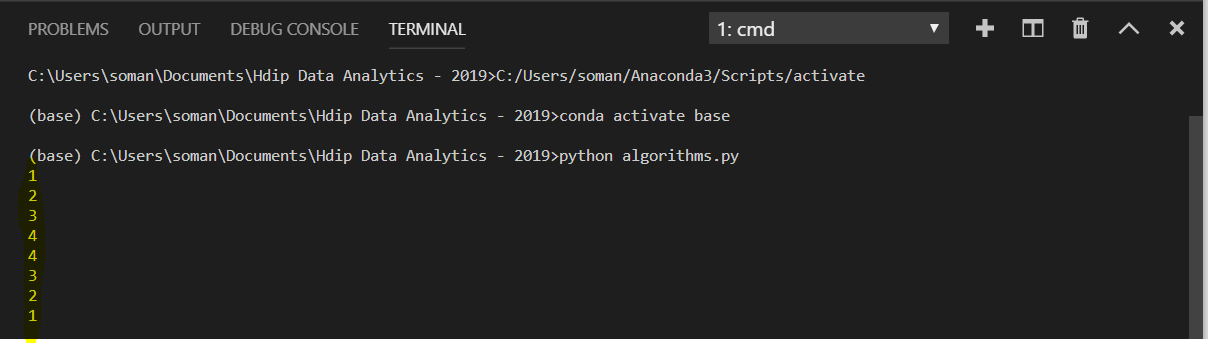
Output:

**1 2 3 4 4 3 2 1**

Recursion Diagram

The output of the given python code is 1 2 3 4

Modified Python Code: Algorithms.py execution output in python



**Output**:

**1 2 3 4 4 3 2 1**

Recursion Diagram

Mystery (1)

Mystery (2)

Mystery (3)

Mystery (4)

Return

Return

Return

Call

Call

Call

Call

* The initial call to the function “Mystery” is made by the statement in line#12 “mystery(1)”
* The function call “mystery(1)” in- turn calls function “Mystery” with an argument value of “2” in line #10
* The function call “mystery(2)” in- turn calls function “Mystery” with an argument value of “3” in line #10
* The function call “mystery(3)” in- turn calls function “Mystery” with an argument value of “4” in line #10
* The function call “mystery(4)” doesn’t meet the conditional statement in line #9 and returns the control to function call “mystery(4)”
* “mystery(3)” returns control to “mystery(2)”
* “mystery(2)” returns control to “mystery(1)”
* “mystery(1)” completes the execution of the function.

# Q2: Question 2

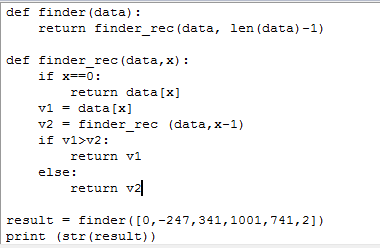
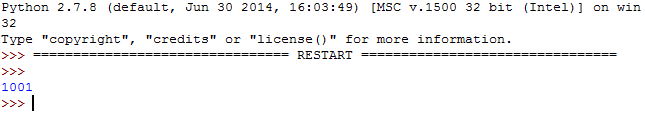
What value is returned by a call to finder when the following array is used as input?

[0,-247, 341,1001,741,22]

## Answer

The answer is **1001**

Python Code: Output:



# Q2 (b) Question

What characteristic of the input data set does the finder method determine? How does it determine the results?

## Answer

* The finder method determines the Largest data element in the input data set
* The largest data element is identified by comparing each data element against the other data element in the input data set.

Explanation with an example finder ([1,2,3,4])

* The initial call to the function “finder” is made by the statement in line#29
* The function call “finder ([1,2,3,4])” calls the function finder\_rec ([1,2,3,4],3)
* The recursive function calls happen in the following order

1. finder\_rec ([1,2,3,4],3)
2. finder\_rec ([1,2,3,4],2)
3. finder\_rec ([1,2,3,4],1)
4. finder\_rec ([1,2,3,4],0)

* The recursive calls results in control returning back in LAST-IN and FIRST OUT order. The function call “finder\_rec ([1,2,3,4],0])” results in exiting the function and the

1. The variables (v1, v2) hold the first 2 value in the input data set
2. Each function call returns the largest of the two variables (v1, v2) and it is stored in variable v2
3. The largest value(v2) is compared against the next value in the input data element.
4. The steps 2 and 3 are executed by the recursive control blocks as if it is in a loop
5. The Largest number is returned by the finder function.

finder ([1,2,3,4])

finder\_rec ([1,2,3,4],2)

finder\_rec ([1,2,3,4],3)

finder\_rec ([1,2,3,4],1)

Returns 4

(v1=3 and v2=4)

Returns 3

(v1=2 and v2=3)

(v1=1 and v2=2)

v1=4

v1=3

finder\_rec ([1,2,3,4],0)

Returns 2

(v1=1 and v2=2)

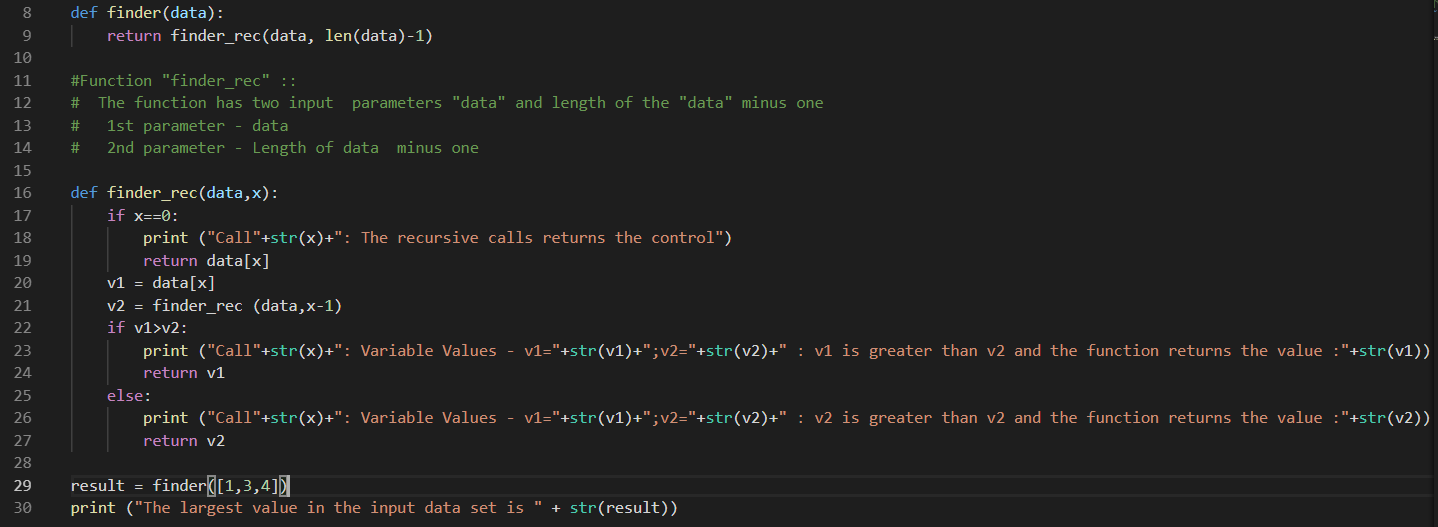
v1=2

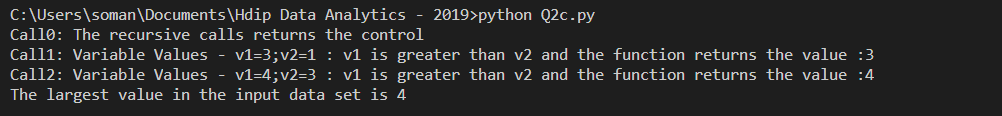
# Q2 (c) Question

Can you add some inline comments to the code above to explain how it works? (2 marks)

## Answer

The modified source code and results are attached





Example input data set is [1,3,4]

* The “finder” method calls function “finder\_rec” which has 2 parameters
  + Input data set --- [1,3,4]
  + Length of the input data set minus one --- 2
* Function “finder\_rec” in turn recursively calls the “finder\_rec” function which has 2 parameters
  + Input data set
  + Length of the input data set, until count is zero
* The “v1” variable is set to 3 and 4 by the recursive calls (i.e. call 1 and call 2 by “finder\_rec”)
* The “v2” variable is set to 1 by the last recursive function call
  + finder\_rec ([1,3,4], 2) v1 = 4 Call 0
  + finder\_rec ([1,3,4], 1) v1 = 3 Call 1
  + finder\_rec ([1,3,4], 0) v2 = 1 Call 2
* Call 2
  + finder\_rec ([1,3,4], 0) - The “if statement” condition is met, and the execution exists from the function “finder\_rec”
* Call 1
  + The control goes back to the “Call 2” and the variable values are set as follows

v1 = 3 and v2 = 1

* + As v1 is greater than v2, the value 3 is assigned to variable v2 by “finder\_rec” function
* Call 0
  + The control goes back to the “Call 1” and the variable values are set as follows v1 = 4 and v2 = 3
  + As v1 is greater than v2, the value 4 is assigned to variable v2 by “finder\_rec” function
* The method “finder” returns the value “4”

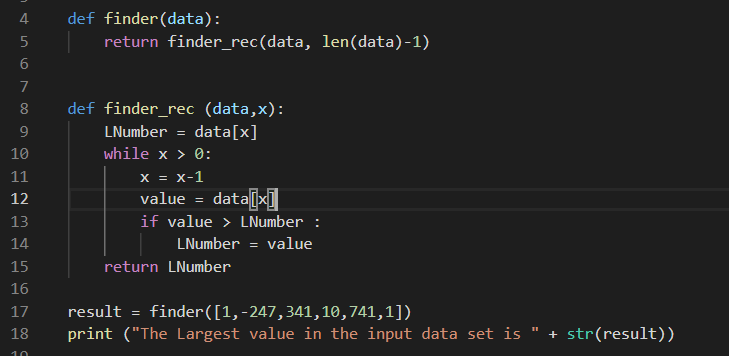
# Q2 (d) Question

Write a method which achieves the same result as finder, but which uses an iterative approach instead of recursion.

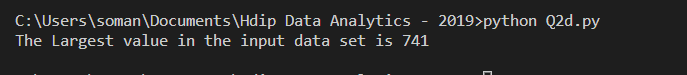
## Answer

The recursive function call is replaced with iterative “while loop”

Source code:



Output:



# Q3 (a) Question

What is the best-case time complexity for this method, and why?

## Answer:

Best-case time complexity: Best case is the function which performs the minimum number of steps on input data of n elements

* The function checks for duplicate elements in the input
* Say there are n elements in the input, the function compares each element in the input against other elements for duplicate except itself.
* If the 2nd element is duplicate of the 1st element in the input argument, then the function does only one comparison and this is the best-case scenario for this function

Example: Best-case input to the function contains\_duplicates ([**1,1**,2,3,4,5])

# Q3 (b) Question

What is the worst-case time complexity for this method, and why?

## Answer:

Worst-case time complexity: Worst case is the function which performs the maximum number of steps on input data of size n

* The function checks for duplicate elements in the input
* Say there are n elements in the input, the function compares each element in the input against other elements for duplicate except itself.
* If the last 2 elements are duplicate, then the function need to perform comparison of each n-1 elements against n elements in the input

Example: Worst-case input to the function: contains\_duplicates ([**0,1**,2,3,**5,5**])

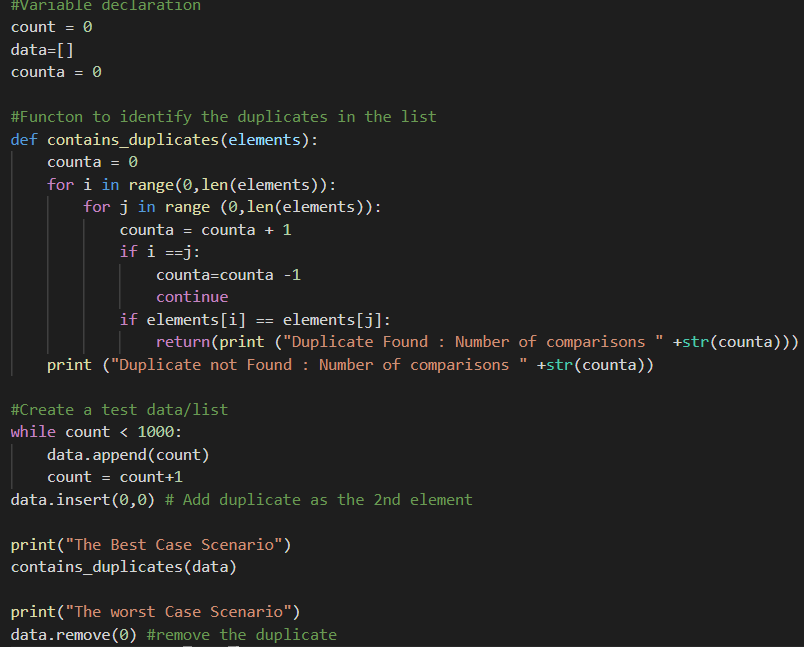
# Q3 (c) Question

Modify the code above, so that instead of returning a boolean indicating whether or not a duplicate was found, it instead returns the number of comparisons the method makes between different elements until a duplicate is found.

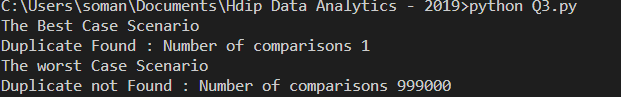
Answer:

* Counter included inside the 2nd For loop and incremented for each step
* Counter is decremented in the self-comparison if loop statement
* Return true statement is replaced with “Duplicate Found : Number of comparison” print statement
* Return false statement is replaced with “Duplicate Not Found : Number of comparison” print statement
* Test list data is generated to display the Best-Case and Worst Case Scenarios

Source Code:



Output:



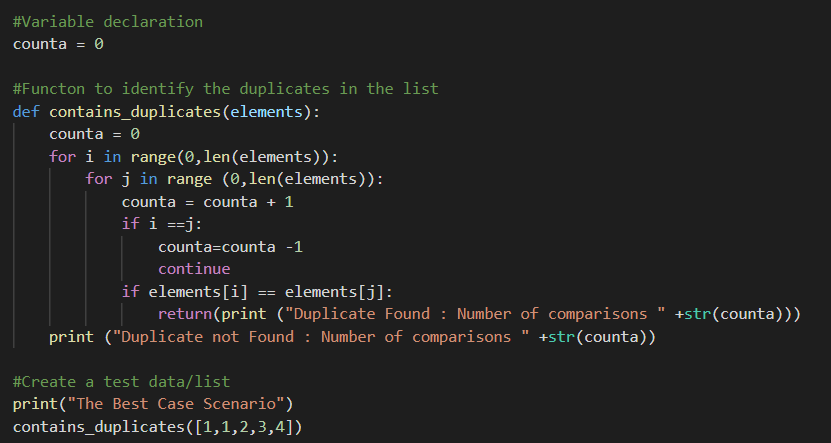
# Q3 (d) Question

Construct an input instance with 5 elements for which this method would exhibit its best-case running time.

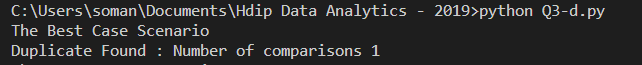
Answer:

Input Element: [1,1,2,3,4]

Source Code:



Output:

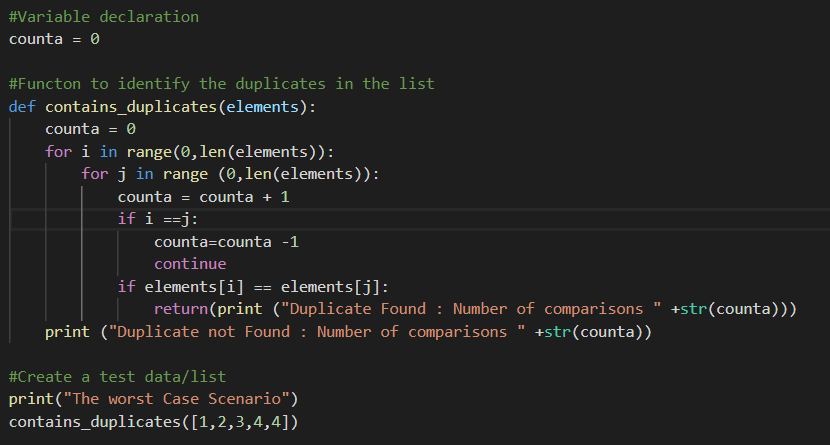


# Q3 (e) Question

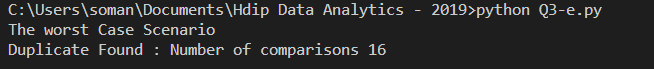
Construct an input instance with 5 elements for which this method would exhibit its worst-case running time. (1 mark)

Answer:

Input Element: [1,1,2,3,4]

Source code:

Output



# Q3 (f) Question

Which of the following input instances, [10,0,5,3,-19,5] or [0,1,0,-127,346,125] would take longer for this method to process, and why?

## Answer

[10,0,5,3,-19,5] – This will take longer to execute as the number of required comparison is 15

[0,1,0,-127,346,125] – This will take shorter time to execute as the number of required comparison is 2