


Step 1: Study [509. Fibonacci Number](#)


Step 2: Ask ChatGPT: "How to implement LeetCode question: 509. Fibonacci Number. Please provide an iterative version solution and a recursive version solution. Please test the solutions with Input: n = 3, Output: 2"

python

 Copy code


```
def fib_iterative(n):  
    if n == 0:  
        return 0  
    elif n == 1:  
        return 1  
  
    fib_prev = 0  
    fib_current = 1  
  
    for _ in range(2, n + 1):  
        fib_next = fib_prev + fib_current  
        fib_prev, fib_current = fib_current, fib_next  
  
    return fib_current
```

python

 Copy code


```
n = 3  
print(fib_iterative(n))
```

 Copy code

 Copy code


2

python


 Copy code

```
def fib_recursive(n):  
    if n == 0:  
        return 0  
    elif n == 1:  
        return 1  
  
    return fib_recursive(n - 1) + fib_recursive(n - 2)
```

python

 Copy code

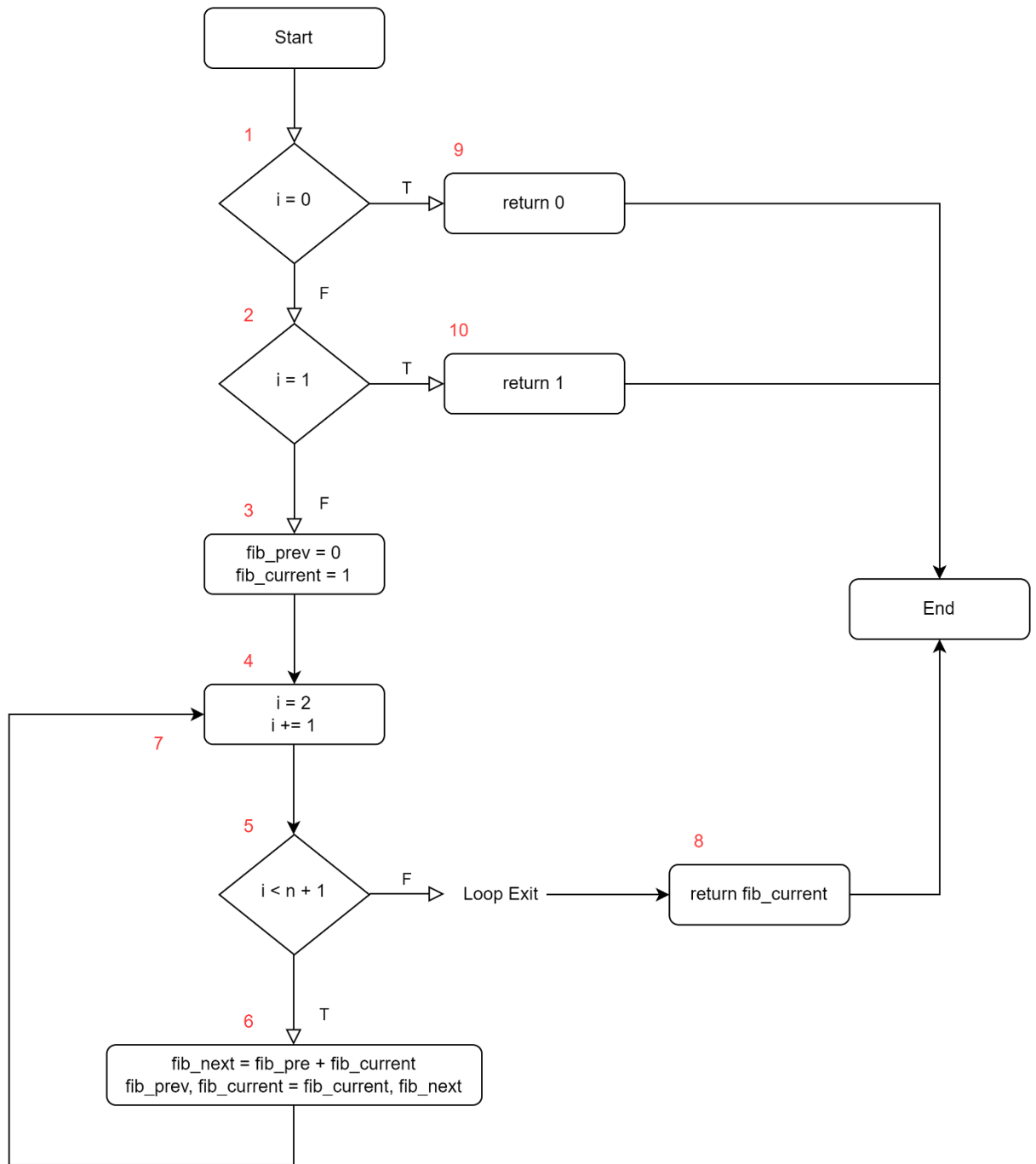
```
n = 3  
print(fib_recursive(n))
```

 Copy code

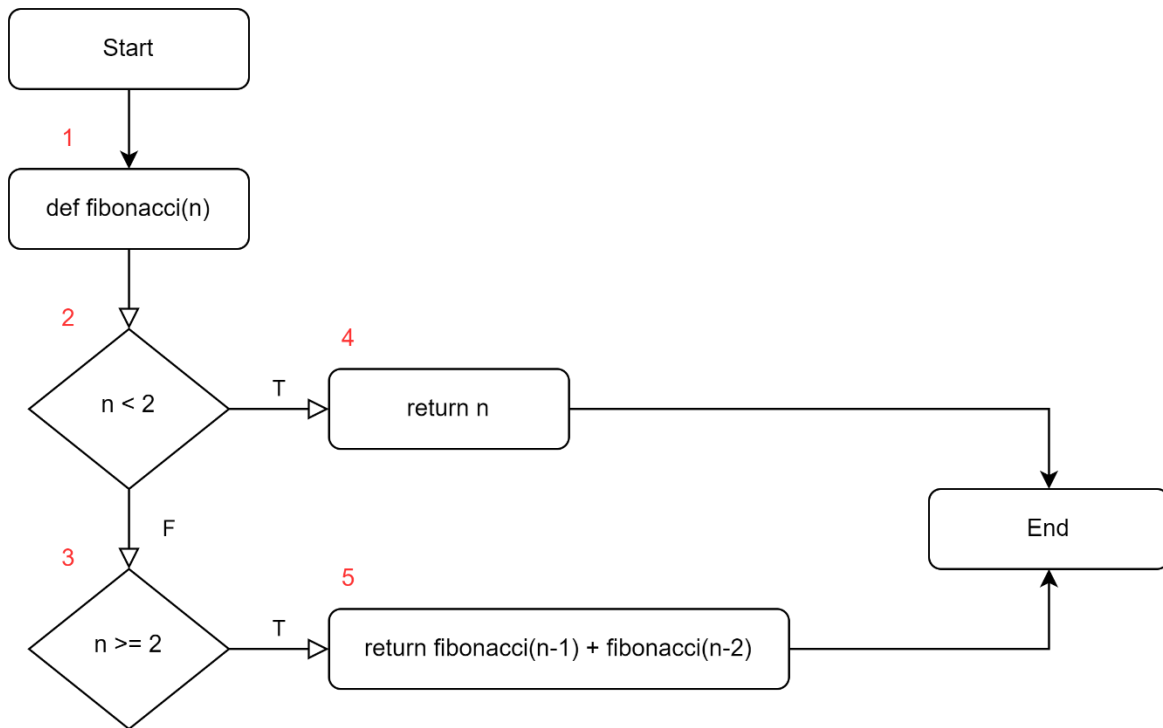
2

Step 3: Convert ChatGPT's Python code into two [flowcharts](#). One flowchart is for Iterative version, the other is for Recursive version.

Iterative:



Recursive:



Step 4: Verify your two flowcharts with two trace tables. One truth table is for the Iterative version, the other is for the Recursive version. You only need to trace "Input: n = 3, Output: 2".

Iterative:

Step	RAM					screen
	i	i < n + 1	fib_prev	fib_current	fib_next	
1	0					
2	1					
3			0	1		
4	2					
5		T				
6			1	1	1	
7	3					
5		T				
6			1	2	2	
7	4					
5		F				
8						2

Recursive:

Step	RAM			screen
	fibonacci(n)	n < 2	n >= 2	fibonacci(n-1) + fibonacci(n-2)
1	fibonacci(n) defined			
2		F		
3			T	
5				2

Step 5: Modify the ChatGPT code to compare the [execution time](#) of both the Iterative version code and the Recursive version code.

- Test with 2 options
 - Option 1: n = 20 cycles

```

test.py x
test.py > ...
1 def fib_iterative(n):
2     if n == 0:
3         return 0
4     elif n == 1:
5         return 1
6
7     fib_prev = 0
8     fib_current = 1
9
10    for _ in range(2, n + 1):
11        fib_next = fib_prev + fib_current
12        fib_prev, fib_current = fib_current, fib_next
13
14    return fib_current
15
16 import time
17 n = 20
18
19 def measureTime():
20     start = time.time()
21     fib_iterative(n)
22     return time.time() - start
23
24 print("time cost for ", n, " iterative: ", measureTime())
  
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

PS D:\MSCS\CS455 Algorithm\w2> C:\Users\odody\AppData\Local\Programs\Python\Python311\python.exe "d:\MSCS\CS455 Algorithm\w2/test.py"

time cost for 20 iterative: 0.0

PS D:\MSCS\CS455 Algorithm\w2>

The screenshot shows a Visual Studio Code editor with a file named `test.py`. The code defines a recursive function `fib_recursive(n)` and a `measureTime()` function. The `measureTime()` function calls `fib_recursive(n)` and returns the time taken. The `main` block sets `n = 20` and prints the time cost for the recursive function.

```
1 def fib_recursive(n):
2     if n == 0:
3         return 0
4     elif n == 1:
5         return 1
6
7     return fib_recursive(n - 1) + fib_recursive(n - 2)
8
9 import time
10 n = 20
11
12 def measureTime():
13     start = time.time()
14     fib_recursive(n)
15     return time.time() - start
16
17 print("time cost for ", n, " recursive: ", measureTime())
```

The terminal output shows the execution of the script:

```
PS D:\MSCS\CS455 Algorithm\w2> & C:/Users/odody/AppData/Local/Programs/Python/Python311/python.exe "d:/MSCS/CS455 Algorithm/w2/test.py"
time cost for 20 recursive: 0.0010027885437011719
PS D:\MSCS\CS455 Algorithm\w2>
```

- Option 2: $n = 100000$ cycles

The screenshot shows a Visual Studio Code editor with a file named `test.py`. The code defines an iterative function `fib_iterative(n)` and a `measureTime()` function. The `measureTime()` function calls `fib_iterative(n)` and returns the time taken. The `main` block sets `n = 100000` and prints the time cost for the iterative function.

```
1 def fib_iterative(n):
2     if n == 0:
3         return 0
4     elif n == 1:
5         return 1
6
7     fib_prev = 0
8     fib_current = 1
9
10    for _ in range(2, n + 1):
11        fib_next = fib_prev + fib_current
12        fib_prev, fib_current = fib_current, fib_next
13
14    return fib_current
15
16 import time
17 n = 100000
18
19 def measureTime():
20     start = time.time()
21     fib_iterative(n)
22     return time.time() - start
23
24 print("time cost for ", n, " iterative: ", measureTime())
```

The terminal output shows the execution of the script:

```
PS D:\MSCS\CS455 Algorithm\w2> & C:/Users/odody/AppData/Local/Programs/Python/Python311/python.exe "d:/MSCS/CS455 Algorithm/w2/test.py"
time cost for 100000 iterative: 0.09075665473937988
PS D:\MSCS\CS455 Algorithm\w2>
```

The screenshot shows the Visual Studio Code interface. The editor window displays a file named `test.py` with the following Python code:

```
1 def fib_recursive(n):
2     if n == 0:
3         return 0
4     elif n == 1:
5         return 1
6
7     return fib_recursive(n - 1) + fib_recursive(n - 2)
8
9 import time
10 n = 100000
11
12
13 def measureTime():
14     start = time.time()
15     fib_recursive(n)
16     return time.time()-start
17
18 print("time cost for ", n, " recursive: ", measureTime())
```

The output window at the bottom shows the execution results:

```
File "d:\MSCS\CS455 Algorithm\w2\test.py", line 7, in fib_recursive
    return fib_recursive(n - 1) + fib_recursive(n - 2)
[Previous line repeated 995 more times]
RecursionError: maximum recursion depth exceeded
PS D:\MSCS\CS455 Algorithm\w2>
```

The status bar at the bottom indicates the current file is `test.py`, line 10, column 11, with 4 spaces, UTF-8 encoding, and CRLF line endings. The Python version is 3.11.3 64-bit.

Step 6: Describe your scientific observation after trying the two options on both iterative program and recursive program.

- Sample observations
 - Test Environment
 - Windows 10
 - RAM size: 1G
 - Test Results

Option	Iterative	Recursive
n = 20 cycles	The program's execution time is 0	The program's execution time is 0.00100278854

n = 100000 cycles	The program's execution time is 0.09075665474	The program has segmentation fault after running 1000 cycles
Big-O	O(n)	O(2^n)

- Note:
 - Please include a screendump to show the test results. See above screenshots.
 - Study [Fibonacci](#) and compare the Big-O for iterative and recursive Fibonacci series implementations
- Step 7: Optional homework: Practice more LeetCode questions about [Subject: Recursion](#)
 - Step 8.1: [509. Fibonacci Number](#), Easy - LC
 - Step 8.2: [Leetcode questions related to Fibonacci](#) - LC