Consider the Fibonacci numbers, defined by

\[ F(0) = F(1) = 1 \]

and

\[ F(n) = F(n - 1) + F(n - 2). \]

For example, the first several Fibonacci numbers are:

1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, ...

Consider the following divide-and-conquer algorithm to compute Fibonacci numbers.

```python
def Fibonacci(n):
    if n == 0 or n == 1:
        return 1
    return Fibonacci(n-1) + Fibonacci(n-2)
```

1. Is this algorithm correct?

2. What is the running time of this algorithm? You don’t need to find it exactly, but is it \( O(n) \) \( O(n^2) \) \( O(n^3) \) \( O(n^c) \) for any constant \( c \)?

3. How could you make this algorithm better?