Understanding sequences

and implementing them via programming

Sukrit Gupta and Nitin Auluck

February 5, 2024

Outline

Acknowledgement and disclaimer

All mistakes (if any) are mine.
I have used several other sources which I have referred to in the appropriate places.

Section 1

Intuition about sequences

Modeling the pandemic

- We have all been (directly or indirectly) affected by the Covid-19 pandemic.
- I want to present a (very) simplistic picture of the pandemic and understand the how it unfolds using sequences.
- Let's see how it unfolds.

- Let us assume that we have 100,000 people on an island. Each of these people meet exactly 2 different people everyday. The infection rate of the virus is 0.8, i.e. if you have the virus and you meet another person then there is a 0.8 probability that the other person will get the virus. The patient is infectious for 1 day after getting infected.
- Let us assume that 100 people initially get infected on day 0. They, however, continue meeting 2 people every day as before.
- The number of new infections on day 1 $= 100 * 2 * 0.8 = 160$
- What will be the number of new infections on day 5?
- What is the relation between the number of infections between day 0 and day n?
- BTW, what will be the total number of infected people on day 5?
- How many days will it take for the whole island of people to get infected?

Section 2

Sequences

Geometric Sequences

- A sequence is called geometric if the ratio between successive terms is constant. Suppose the initial term a_0 is a and the common ratio is r. Then we have:
  $a_0 = a$
  $a_1 = a_0 \cdot r$
  $a_2 = a_1 r = a_0 r \cdot r = a_0 r^2$ and so on ...
- Closed formula: $a_n = a \cdot r^n$
- Recursive definition: $a_n = a_{n-1} \cdot r$ with $a_0 = a$

Continuing our example ...

- Now, let's assume that the whole island has been infected once and they have developed immunity to the virus. The government opens the island for tourists and 1000 infected tourists visit the island every day. Out of these 10% need hospitalization and stay on the island, the remaining return.
- Let us assume that we have 1000 hospitalized people on day 0. How many hospital beds do you need on day 5?
- If the island has 5,000 hospital beds, how many days will it take to fill all of them?

Arithmetic Sequences

- If the terms of a sequence differ by a constant, we say the sequence is arithmetic. The sequence is given by:
  $a_0 = a$
  $a_1 = a_0 + d = a + d$
  $a_2 = a_1 + d = a + d + d = a + 2d$
  $a_3 = a_2 + d = a + 2d + d = a + 3d$
  and so on ...
- Closed formula: $a_n = a + d \cdot n$
- If the initial term a_0 of the sequence is a and the common difference is d, i.e. $a_n - a_{n-1} = d$. Then we have the recursive formulation:

$$a_n = a_{n-1} + d$$

  with $a_0 = a$.

Section 3

Induction

Proofs again ...

- What were the proof techniques that we studied?
- We studied proof techniques to convince ourselves and others that a mathematical statement is always true.
- While some mathematical statements can be proved by expanding on what they mean, others are very difficult to prove.
- Induction is a style of arguments that we can use to prove statements.

Proof by induction

Start with the statement that you want to prove: "Let $P(n)$ be the statement ...". To prove that $P(n)$ is true for all $n \geq 0$, you must prove the following two facts:

- **Base case**: Prove that P(0) is true. You do this directly. This is often easy.
- **Inductive case**: Prove that $P(k) \rightarrow P(k+1)$ for all $k \geq 0$, i.e. prove that for any $k \geq 0$ if $P(k)$ is true, then $P(k+1)$ is true as well.

Assuming you are successful on both parts above, you can conclude, "Therefore by the principle of mathematical induction, the statement P(n) is true for all $n \geq 0$."

Inductive case: Prove that $P(k) \rightarrow P(k+1)$ for all $k \geq 0$, i.e. prove that for any $k \geq 0$ if P(k) is true, then P(k + 1) is true as well.

Do you see something familiar in the inductive case?

The inductive case is the proof of an if ..., then ... statement.

You can assume P(k) is true. You must then explain why P(k + 1) is also true, given that assumption.

Example: Proof by induction

Show that $1 + 2 + ... + n = \frac{n(n+1)}{2}$ for all $n \geq 1$.

In other words, $P(n) = \frac{n(n+1)}{2}$ is true for any $n \geq 1$.

What is the base case here?

P(1) is true. Easy?

What is the inductive case?
For any $k \geq 1$, prove that if P(k) is true then P(k + 1) is true. Easy?
Let's see.

Prove that if P(k) is true, then P(k + 1) is true (for any $k \geq 1$).

- Since $P(k)$ is true, $P(k) = 1 + 2 + ... + k = \frac{k(k+1)}{2}$.
- For $P(k+1)$, we should get $\frac{(k+1)(k+2)}{2}$. How?
- $P(k+1) = P(k) + (k+1)$ which translates to $P(k+1) = \frac{k(k+1)}{2} + (k+1)$
- If $P(k)$ was true, then $P(k+1)$ is also true.

Therefore by the principle of mathematical induction, the statement P(n) is true for all $n \geq 1$.

Quirky Monkey Q

Using proof by induction show that $1 + a + a^2 + ... + a^n = \frac{a^{n+1} - 1}{a - 1}$ for all $n \geq 0$. Assume $a \neq 1$.

Example: Proof by induction

You need to mail a package, but don't yet know how much postage you will need. You have a large supply of 8-cent stamps and 5-cent stamps. Prove that it is possible to (exactly) make any amount of postage greater than 27 cents using just 5-cent and 8-cent stamps.

In other words, P(k) is true for any $k \geq 28$.

What is the base case here?

Demonstrate that P(28) is true. Easy?

What is the inductive case?

Prove that for any $k \geq 28$ if P(k) is true, then P(k + 1) is true. Easy? Let's see.

Prove that if P(k) is true, then P(k + 1) is true (for any $k \geq 28$).

- Since $P(k)$ is true $k = 8(x) + 5(y)$.
- For $k \geq 28$, there are **at least** 3 stamps of 5 cents or 3 stamps of 8 cents.
- Replace 3 stamps of 5 cents by 2 stamps of 8 cents. Or replace 3 stamps of 8 cents by 5 stamps of 5 cents.
- If $P(k)$ was true, then $P(k + 1)$ is also true.

Therefore by the principle of mathematical induction, the statement P(n) is true for all $n \geq 28$.

Section 4

Coding sequences

Constructs for implementing sequences

- We have seen arithmetic and geometric sequences.
- We have also seen the idea of induction.
- How do we implement sequences in programming (specifically Python)?
- Loops! You got me.
- What if I told you that there is a more powerful idea (not construct) that will help you implement sequences in Python?

Recursion

Although there are multiple definitions for recursion, the most elegant one I found is:

Definition

Recursion is the process of defining a problem (or the solution to a problem) in terms of a *simpler version of itself*.

Nature has recursion.

Nature has recursion.

Figure: Snowflake [1]

Figure: Tree branch [2]

[1] Alexey Kljatov: https://www.flickr.com/photos/chaoticmind75/49268744233

[2] https://mobile.twitter.com/mageed/status/1311020561347170304/photo/1

Nature has recursion.

Figure: Plant [3]

[3] Credits: Maa

Recursive function

Figure: Russian Dolls

- **Base case**: Input(s) for which the function produces a result trivially (without recurring);
- **Recursive case**: Input(s) for which the program recurs (calls itself).

Recursive function

```python
def func(n):
    """This is a recursive function"""
    if condition:
        base case # cannot be broken any further
    else:
        recursive case #calls func again
```

Example: Factorial

Factorial(n) $= 1 \cdot 2 \cdot 3 ... (n-1) \cdot n$

```python
def factorial(n):
    """This is an iterative function
    to find the factorial of an integer"""
    fact = 1
    while n > 1:
        fact *= n
        n -=1

    return fact
```

We have seen this before. Let's see a recursive solution.

Factorial: Recursive

$\text{Factorial(n)} = \text{n} \cdot \text{Factorial(n-1)}$

```
def factorial(n):
    """This is a recursive function
    to find the factorial of an integer"""
    if n == 1 or n == 0:
        return 1 #base case
    else:
        return (n * factorial(n-1)) #recursive case
```

Let's see how this works.

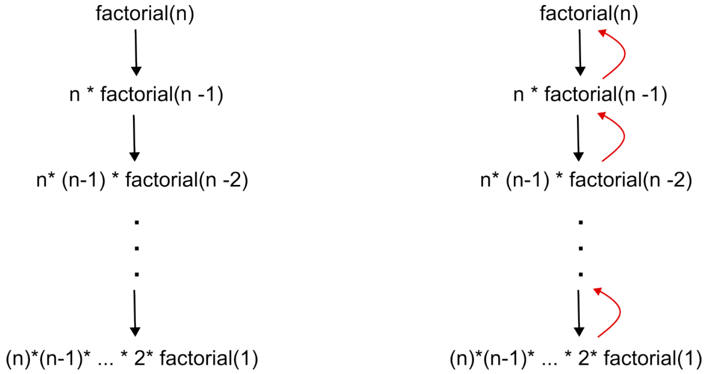
Factorial: Recursive

factorial(n)

factorial(n)
$\downarrow$
n * factorial(n -1)

factorial(n)
$\downarrow$
n * factorial(n -1)
$\downarrow$
n* (n-1) * factorial(n -2)

Factorial: Recursive



Example: Fibonacci numbers

Fibonacci numbers are given by: 0, 1, 1, 2, 3, 5, ...

Basically, n^{th} number in the Fibonacci sequence = $(n-1)^{th} + (n-2)^{th}$ numbers in the sequence

Interesting anecdote from graduate days ...

Fibonacci numbers: Iterative solution

```python
def fibo(n): #computing the nth number in Fibonacci series
    num_1, num_2 = 0, 1
    current_num = 0
    while n > 0:
        current_num = num_1 + num_2
        num_2 = num_1
        num_1 = current_num
        n -= 1

    return current_num
```

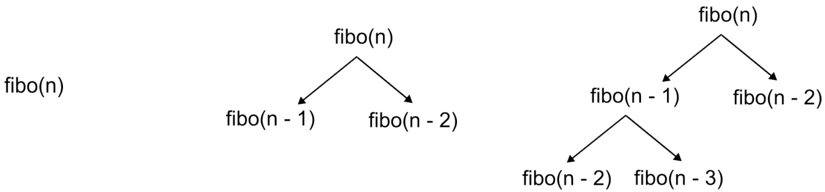
Figure out how this works as homework.

Fibonacci numbers: Recursive solution

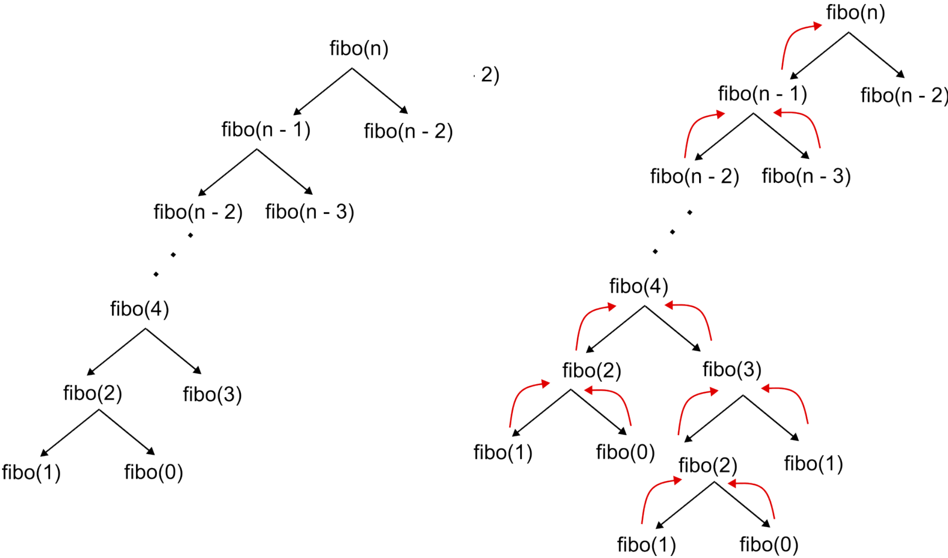
```python
def fibo(n): #computing the nth number in Fibonacci series
   if n == 1 or n == 0:
       return n
   else:
       return(fibo(n-1) + fibo(n-2))
```

Let's see how this works.

Fibonacci numbers: Recursive



Fibonacci numbers: Recursive



Palindrome

A palindrome is a word, phrase, number, or sequence of words that reads the same backward as forward. For example: 'rotor', 'mom', et cetera.

```
def is_palindrome(S, start, end):
    if len(S) == 1:
        return True
    if S[start] != S[end]:
        return False
    if start < end - 1:
        return is_palindrome(S, start+1, end-1)
    return True

check_str = 'malayalam'
print(is_palindrome(check_str, 0, len(check_str)))
```

What did we learn today?

1. Intuition about sequences
2. Sequences
3. Induction
4. Coding sequences

Thank you!