RESEARCH REPORT

Relationship between the notion of induction in mathematics and notion of recursion in computer science/programming.

NAVDEEP SINGH RANDHAWA
5/18/19
Discrete Mathematics

Contents

INTRODUCTION:

In this research report, the relationship between mathematical induction and recursion in computer science is analysed and explained using various examples.
In first few paragraphs we will understand the concepts related to mathematical induction and recursion in computer science separately, and then we will explore the connection between the both. The explanation regarding relationship is proved by help of codes and mathematical proofs simultaneously. So, let's get into our report and perform the mentioned tasks.

WHAT IS NOTION OF INDUCTION IN MATHEMATICS?

Mathematical induction is special method of proof which is used to prove different mathematical statements to be always true.
For example, if a statement holds true value for a value say, 'n' then if it satisfies the mathematical induction proof then it will satisfy for 'n+1' and so on (Leron and Zazkis, 1986).

STRUCTURE OF PROOF:

First consider a statement to be proved, let's call it $P(n)$, where $n \geq 0$.
Using mathematical induction, we prove the statement $P(n)$ is always true given that value of n is always greater than or equal to 0.

STEP 1: BASE STEP

In this step prove $P(n)$ for $n = 0$.

STEP 2: INDUCTIVE STEP

In this step we prove that $P(k) \rightarrow P(k+1)\ for\ all\ k \geq 0$.
This is a "if .... then ...." proof where $P(k)$ is our if statement and $P(k+1)$ is then statement.
We assume that $P(k)$ is always true for given domain of k, this is known as Inductive hypothesis.
We use inductive hypothesis and prove that $P(k+1)$ is also true with a valid reason (Leron and Zazkis, 1986).

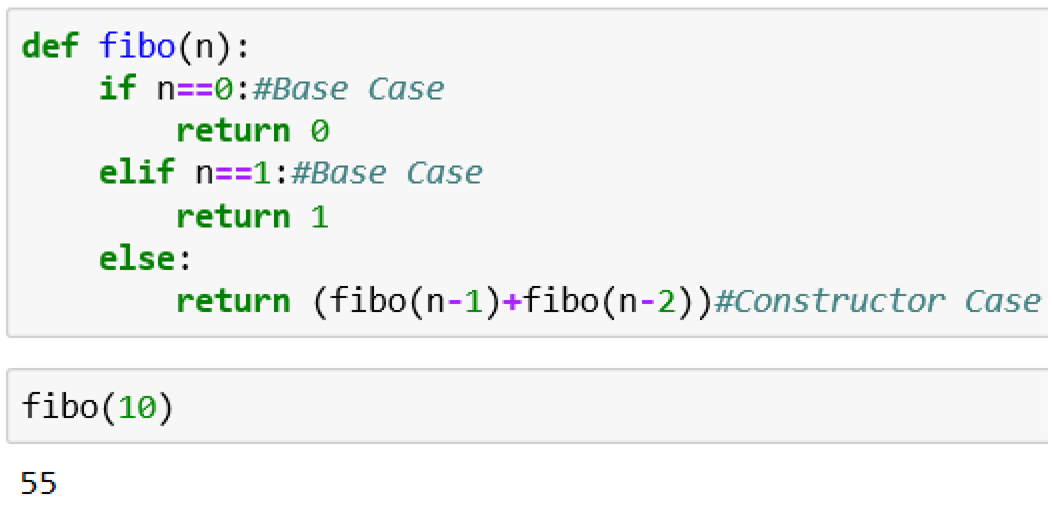
WHAT IS NOTION OF RECURSION IN COMPUTER SCIENCE?

Recursion is a method where a problem is solved using smaller instances of same problem.
In computer programming recursive algorithms is a tactic deployed to achieve solution of a bigger problem by dividing it into similar sub-problems and then finding solution to these sub-problems and finding results by combining solutions. This method is also known as Divide and Conquer tactic (Cormen and Balkcom, 2019).

RECURSIVE FUNCTION

1) A recursive function has base cases, or we can say input for which the function produces result without recursion.
2) A recursive function has recursive case, or we can say inputs for which the function produces result by recursively calling itself, also known as Constructor case.

Example of recursive function to find nth Term in Fibonacci Series:



```
def fibo(n):
    if n==0:#Base Case
        return 0
    elif n==1:#Base Case
        return 1
    else:
        return (fibo(n-1)+fibo(n-2))#Constructor Case
```

```
fibo(10)
```

```
55
```

The base case and recursive case are defined in code using comments.
Breakdown of above example:
This function uses recursive approach to find 10th term in Fibonacci series, when we get fibo(0) or fibo(1) our values in base cases are returned and then function starts adding previous values to find 10th term. So, basically first we go from $n \ to \ 0$ and then get outputs for values from $0 \ to \ n \ which \ gives \ us \ the \ n^{th} \ term$.

CONCLUSION FROM ILLUSTRATIONS OF RECURSION AND MATHEMATICAL INDUCTION:

There seems to be some similarity between mathematical induction and recursion, which can be noticed in the definitions explained above but now with help of some examples and analysis of those examples we will draw an inference about the relationship between two.

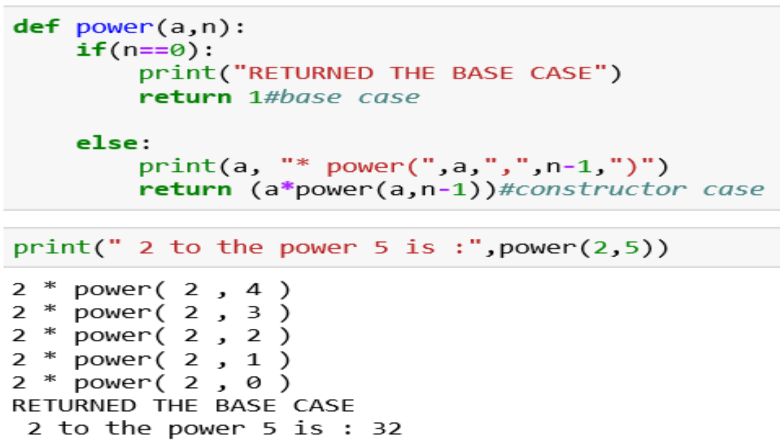
RELATIONSHIP BETWEEN MATHEMATICAL INDUCTION AND RECURSION

In this part the relation between induction and recursion is analysed using two examples.

First Example: Power Function(a^n)

- ***Recursive function***: To find $a^n, where \ a \ is \ non \ zero \ real \ number$ and $n \ is \ zero \ or \ any \ positive \ integer$.

  Here is the illustration of recursive function 'power' along with its output which gives an idea of each recursive step taken to achieve the task.



```
def power(a,n):
    if(n==0):
        print("RETURNED THE BASE CASE")
        return 1#base case

    else:
        print(a, "* power(",a,",",n-1,")")
        return (a*power(a,n-1))#constructor case
```

```
print(" 2 to the power 5 is :",power(2,5))
```

```
2 * power( 2 , 4 )
2 * power( 2 , 3 )
2 * power( 2 , 2 )
2 * power( 2 , 1 )
2 * power( 2 , 0 )
RETURNED THE BASE CASE
 2 to the power 5 is : 32
```

- **<u>PROOF WITH MATHEMATICAL INDUCTION:</u>**

$$\mathrm{P(n)} = \mathrm{a^n} \textit{ where } n \geq 0$$

$$Domain: n = \{0,1,2,3,4, \dots \dots\}$$

1) **<u>Base Step:</u>**

$$Let's \quad take \quad n = 0$$
$$P(0) = a^0 = power(a, 0) = 1$$

Note: From recursive function base case when $n == 0$, $power(a, 0)$ returns 1.

So,

$P(0)\ is\ true\ for\ both\ recursive\ function\ and\ mathematical\ induction.$

2) **<u>Inductive Step:</u>**

$$Let's \quad take\ ,$$
$$P(k) = a^k, for\ every\ k \geq 0.$$

Inductive Hypothesis (assuming the below given statement to be true)

$$P(k) = power(a, k) = a^k, where\ k \geq 0, for\ all\ a\ such\ that\ a\ is\ not\ equal\ to\ 0$$

Using inductive hypothesis to show that:

$$P(k+1) = \ power(a, k+1) = a^{k+1}, where\ k \geq 0$$

From recursive function (power) above $k \geq 0$, therefore $k + 1 \geq 0$.

We can rewrite above function as,

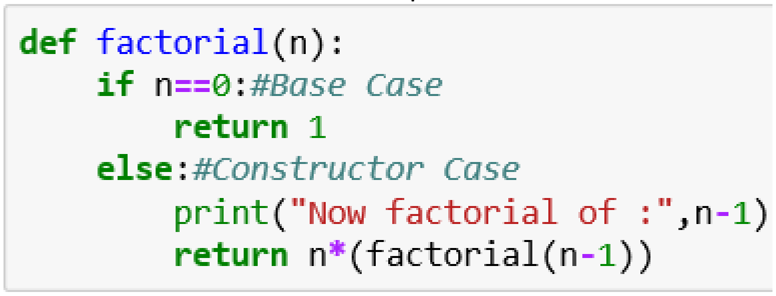
$$power(a, k+1) = a * power(a, k)$$
$$(From\ inductive\ step\ power(a, k) = a^k)$$
$$power(a, k+1) = a * a^k$$
$$power(a, k+1) = a^{k+1}$$

Hence, we have proved that $P(k) \rightarrow P(k+1)$ (Moura, 2010).

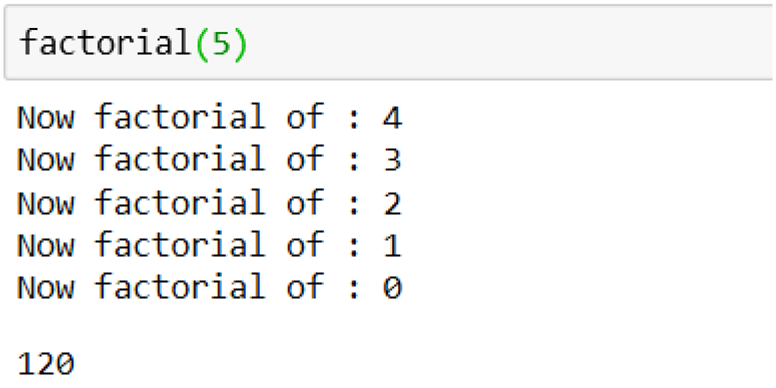
Second Example: Factorial Function $(n!)$

- Recursive function to find factorial of a given non negative integer $'n'$.

  Here is the illustration of recursive function 'factorial' along with its output which gives an idea of each recursive step taken to achieve the task.



```python
def factorial(n):
    if n==0:#Base Case
        return 1
    else:#Constructor Case
        print("Now factorial of :",n-1)
        return n*(factorial(n-1))
```



```python
factorial(5)
```

```
Now factorial of : 4
Now factorial of : 3
Now factorial of : 2
Now factorial of : 1
Now factorial of : 0

120
```

- **PROOF WITH MATHEMATICAL INDUCTION:**

$$\mathrm{P(n) = n.(n-1)!}\ \ where\ n > 0, for\ the\ given\ function\ 'factorial'$$

$$Domain: n = \{1,2,3,4, \ldots \ldots\}$$

1) **Base Step:**

$$Let's\ \ take\ \ n = 1$$
$$P(1) = 1.(1-1)! = factorial(0) = 1$$

Note: From recursive function when base case, $n == 0$, $factorial(0)$ returns 1.
So,
$P(1)\ is\ true\ for\ both\ recursive\ function\ and\ mathematical\ induction.$

2) **<u>Inductive Step:</u>**

$$Let's\ \ take,$$
$$P(k) = k.(k-1)!, for\ every\ \ k > 0.$$

Inductive Hypothesis (assuming the below given statement to be true)
$P(k) = factorial(k) = k.(k-1)!, where\ every\ \ k > 0.$
Now,
Using inductive hypothesis to show that
$P(k+1) = factorial(k+1) = (k+1).k!\ , where\ k > 0$
From recursive function (power) above $k > 0$, therefore ($k + 1 \geq 0$).
We can rewrite above function as,
$$factorial(k+1) = (k+1) * factorial(k)$$
$$(From\ inductive\ step\ factorial(k) = k.(k-1)!)$$
$$factorial(k+1) = (k+1).(k.(k-1)!),$$
from above equation we know that, $(\mathrm{k}.(\mathrm{k}-1)!) = \mathrm{k}!$
$$factorial(k+1) = (k+1).k!$$
Hence, we have proved that $P(k) \rightarrow P(k+1)$ (Moura, 2010).

CONCLUSION:

Using the mathematical induction in this instance we have proved that the recursive function used to find power of a given number is true for as large a value unless it does not violate the given conditions or the base case (since in base case a pre-defined value is returned).
By mathematical induction we have activated a series of steps in which for every case, the subsequent case will also be true given all the condition are met and this implies that if we stop at a certain value for example in example 2 if we stop at$(n == l)$, then the given factorial function is going to be true while going from $(l)\ to\ 0$. This notion of induction is almost same as notion of function calling itself in recursion, hence induction can be used by computer programmers to verify the correctness of a recursive function.
As seen in examples above, induction is 'upward' intuition whereas recursion is 'downward' intuition and the 'base rule' of induction is basically 'stop rule' in recursion.
Now the question arises how can we use induction and recursion together? The simple answer is that supposedly if a computer scientist or a programmer is writing a recursive function to solve a problem then if he/she has good knowledge of Induction then using inductive proof for the given problem as we did in example 2, using the inductive proof, one can give authority to the recursive algorithm.
We used this simple version of mathematical induction to prove simple recursive functions and furthermore we have structural induction which is used to prove functions dealing with more complex data structures like trees, lists and problems like tower of Hanoi reason (Leron and Zazkis, 1986).

Reference List

Leron, U. and Zazkis, R. (1986). Computational Recursion and Mathematical Induction. *For the Learning of Mathematics*, [online] 6(2), pp.25-28. Available at: http://www.jstor.org/stable/40247809 [Accessed 5 May 2019].

Moura, L. (2010). *Induction and Recursion*.

Cormen, T. and Balkcom, D. (2019). *Recursion*. [online] Khan Academy. Available at: https://www.khanacademy.org/computing/computer-science/algorithms/recursive-algorithms/a/recursion [Accessed 7 May 2019].