Algorithm Analysis Of Recurrence/Recursive Relations

In this part 1 of 5 we will understand and analyze complexity of a recursive relation.

Algorithm Analysis And Design

Getting started with Recurrence Relations

Objectives:

* What is a recurrence relations
* Important points while solving recursive problem
* Understanding recursive relation using example
* Analysis of recursive relation

What is a recurrence relation?

* An algorithm is recursive if its calls itself.
* They solve the problem by reducing a larger input into a smaller input and then solve problem for a smaller input.

2 important points while solving a recursive problem

* Always identify the base case and result associated.
* Make sure that recursive call is for a smaller problem.

A base case is lowest threshold of a function. The case for which we have to find the lowest value is a base case.

Example: Let us consider a basic example of a factorial. We know that a factorial of a function is given as:

F(n) = (n) \* (n-1) \* (n-2) \* (n-3) \* …… \* 3 \* 2 \* 1

Where n is an integer.

Suppose we are planning to find the factorial of 5, so by using the above function we have:

F(5) = 5 \* 4 \* 3 \* 2 \* 1

=120

Now let’s write a basic algorithm that would calculate a factorial of a number using recursive relation.

Algorithm:

<script src="https://gist.github.com/NavneetPrakashSingh/9c1921058fe18787c13cd7b74b778421.js"></script>

So in the above algorithm a lot has happened. Before beginning to analyze the algorithm, we need to understand what’s actually happening above. The two important points we mentioned above were always identify the base case, that we have taken care of by defining case for n = 0 which returns a value of 1 since 0! = 1.The second important point to consider was that the recursive call is for a smaller problem which is taken care of in else case. In the else case it calls for a smaller value every time till it hits the base case. Lets explain that further

Suppose we want to find the factorial of 5 using the above algorithm

<script src="https://gist.github.com/NavneetPrakashSingh/1ac7e13ba44b2387973cceed73ef8f14.js"></script>

Analysis of Above Algorithm

For the first two lines of algorithm i.e. the base case which we hit only once hence the running time for the algorithm is O (1). Now for the other part where we don’t hit the base case we know that we go through the algorithm (n-1) times, making the complexity for the non-base cases = O(n-1).

Let us assume ‘c’ as a constant value for large values of T(n) which is given as follows:

<script src="https://gist.github.com/NavneetPrakashSingh/660d35cc683096b23119ce40c3bce829.js"></script>

That’s all folks! We just created our very first recursive function, understood it via example and analyzed it using simple technique. Now it’s time to up the stakes. In our next tutorial we will understand 4 important techniques for algorithm analysis for recursive functions. Those are as follows:

Substitution Method

Iterative Method

Recursive Method

Master Method

So strap on you learning seat belt and lets learn to analyze algorithm using different techniques.