

# Assignment 1

Kuntal Sudhir Kokate - EE18BTECH11028

Download all latex-tikz codes from

<https://github.com/Kkuntal990/C-DS/blob/main/Assignment1/assignment1.tex>

## 1 PROBLEM

(Q 48) Consider the following C function.

```
int tob(int b, int *arr){
    int i;
    for (int i = 0; b > 0; i++){
        if(b%2)
            arr[i] = 1;
        else
            arr[i] = 0;
        b = b / 2;
    }

    return (i);
}
```

```
int pp(int a, int b){
    int arr[20];
    int i, tot = 1, ex, len;
    ex = a;

    len = tob(b, arr);
    for (int i = 0; i < len; i++)
    {
        if(arr[i] == 1)
            tot = tot * ex;
        ex = ex * ex;
    }

    return tot;
}
```

The value returned by  $pp(3,4)$  is ?

## 2 SOLUTION

$$pp(3,4) = 81$$

## 2.1 Explanation

Characteristics of  $tob$  function:

- 1) Converts positive integers to their binary representation.
- 2) In the case of negative integer, it returns 1 as output.
- 3) Returns the decimal number in binary representation as output

$$tob(b) = \begin{cases} 1 & b < 0 \\ (b)_2 & b \geq 0 \end{cases}$$

where  $b \in \mathbb{Z}$

For eg.  $tob(4) = 100$ .

Mathematical description of  $pp$  function:

$$pp(a, b) = \begin{cases} a^b & b \geq 0 \\ 1 & b < 0 \end{cases}$$

where  $a, b \in \mathbb{Z}$

$$\Rightarrow pp(3,4) = 81$$

## 2.2 Complexity Analysis

- In  $tob$ , for loop is halving the value of  $b$  every time, there are only  $\log_2 b$  steps possible.
- Complexity of  $tob$ :  $O(\log_2(b))$ .
- In  $pp$ , iterations are equal to number of bits in binary representation of  $b$ .
- Complexity of  $pp$ :  $O(\log_2(b))$ .

## Simulation results:

- 1) Used cmake library to import C functions into python.
- 2) Used C itself to store the time data into a '.dat' file.

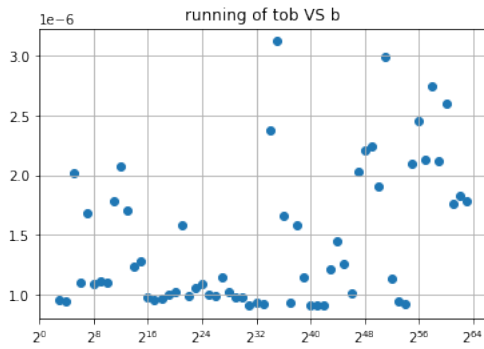


Fig. 1: X-axis represents input  $b$  and Y-axis represents time in seconds. The time for executing our function is very small compared to the overhead added time. So the simulation results are not clear enough.

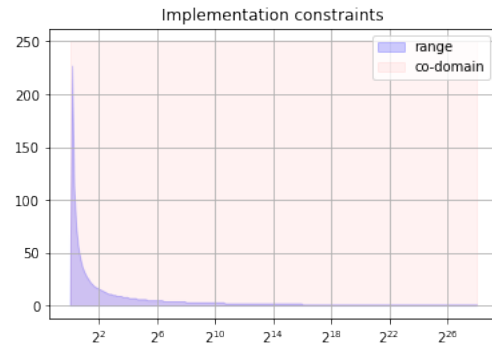


Fig. 2: Implementation restricts our range to the small purple shaded region.

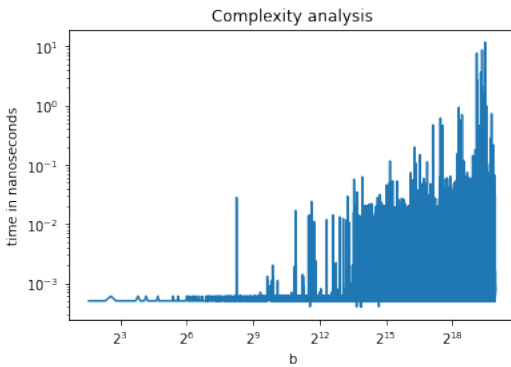


Fig. 2: Results seem more reasonable than previous attempt.

[https://github.com/Kkuntal990/C-DS/blob/main/Assignment1/codes/A1\\_4.cpp](https://github.com/Kkuntal990/C-DS/blob/main/Assignment1/codes/A1_4.cpp)

of the repository.

- Time complexity :  $O(n)$ , where  $n$  is the length of the large number. It is worse than previous logarithmic complexity.
- But we can convert integers containing more than 64 bits to their corresponding binary representation.
- Better range of our function can be obtained.

### 2.3 Implementation constraints

- Inputs are restricted by length of the array we are using to store binary representations,  $y < 2^{\text{length}(\text{array})}$ , where  $\text{length}(\text{array}) = 20$ .
- We are using `int` data type in C to represent our integers so,  $x^y < 2^{32}$ .
- We can improve those constraints at the cost of time complexity.
- Store large decimal number as a string.
- Treat individual character as a single digit number and take the remainder to the next character.
- This way we can perform division, multiplication and addition of the large number represented as string.
- Using that we convert that decimal string to binary string.
- Code for that can be found in