Assignment 1

Name : Hrishikesh Rajan
Email : hrishikeshrajan3@gmail.com
LinkedIn : https://www.linkedin.com/in/hrishikesh-rajan-96aa70165

Q 1)
Compute and return the square root of x, where x is guaranteed to be a non-negative integer. And since the return type is an integer, the decimal digits are truncated and only the integer part of the result is returned. Also, talk about the time complexity of your code.

Test Cases:
Input: 4
Output: 2

Input: 8
Output: 2

Explanation: The square root of 8 is 2.8284…., the decimal part is truncated and 2 is returned.

*A Javascript implementation is provided

Answer :

```
function calculate(num,low,high){

  if(low<=high){
    let mid = Math.floor(low + ((high - low)/2));
    const square = mid * mid;

    if(square === num){
        return mid;
    }
    if(square < num && ((mid+1)*(mid+1)) > num){
        return mid;
    }
    if(square > num){
        return calculate(num,low,mid-1)

    }
    if(square < num){
        return calculate(num,mid+1,high)
    }
  }

return -1;
}
```

```
function findSqrt(num){
return calculate(num,0,num)
}

const result = findSqrt(8)
console.log(result);
```

Output : 2

Time Complexity:
The recurrence relation is formed by

T(n) = T(n/2) + c , c = constant. (1)

The relation gives T(n/2) because our search space for multiplication was limited to half of n, that is n/2, where n is the input number.

Applying Master's Theorem

T(n) = aT(n/b) + f(n) , where f(n) = $\Theta(n^k log^p n)$ (2)

From the above recurrence relation (1)
a=1
b=2
k=0
p=0

Substituting values
log_a^b **=** log_2^1 **=** 0
That means log_a^b **= k** and **P > -1,** This relation comes under case 2.
Then the equation becomes **$\Theta(n^k log^{p+1} n)$**

= $\Theta(n^0 log^{0+1} n)$
= $\Theta(1 * log^1 n)$
= $\Theta(log\ n)$

Time Complexity = O(logn)
Space Complexity = O(1)