CS263 ASSIGNMENT 2

NAME:

ARCHIT AGRAWAL

ROLL NO.:

202052307

SECTION:

A

***Problem***

Given an integer n, find the greatest integer lesser than square root of n.

For example:

Input: n = 57

Output: 7

Explanation: Since, square root of 57 is 7.5498, therefore floor(sqrt(57)) = 7

Input: n = 2500

Output: n = 50

Explanation: Since, square root of 2500 is 50, therefore floor(sqrt(2500)) = 50

***Algorithm***

It can be solved using Binary Search.

* Taking base cases into consideration and returning n when n = 0 or n = 1.
* If n is negative, throw an exception.
* Initialize lower bound variable start = 0 and upper bound variable end = n.
* Make a variable that stores answer(ans).
* Make a variable mid which stores the middle value of the search space.
* Run a loop until l <= r, the search space vanishes.
* If the square of mid ((start + end)/2) is less than or equal to n. If yes, then search for a larger value in second half of search space, i.e start = mid + 1 and update ans = mid.
* Else if the square of mid is more than n then search for a smaller value in first half of search space, i.e end = mid – 1.
* Print ans

***Code***

import java.util.\*;

public class BinarySearch{

    public static int floorSquareRoot(int n){

        // Base Cases

        if(n < 0) throw new IllegalArgumentException("No real square root of "+n);

        if (n == 0 || n == 1) return n;

        //Binary Searching floor(sqrt(x))

        int start = 1;

        int end = n;

        int ans = 0;

        while (start <= end){

            int mid = (start + end) / 2;

            // If x is a perfect square

            if (mid \* mid == n) return mid;

            // updating the answer when mid\*mid is

            // smaller than x, and moving closer to sqrt(x)

            if (mid \* mid < n){

                start = mid + 1;

                ans = mid;

            }

            else end = mid-1;

        }

        return ans;

    }

    public static void main(String args[]){

        Scanner sc = new Scanner(System.in);

        System.out.println();

        System.out.print("Enter an integer x to find floor(sqrt(x)) : ");

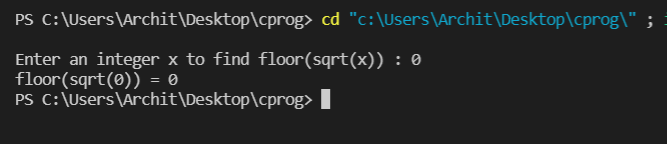
        int x = sc.nextInt();

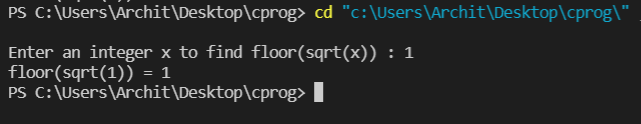
        System.out.println("floor(sqrt(" + x + ")) = " + floorSquareRoot(x));

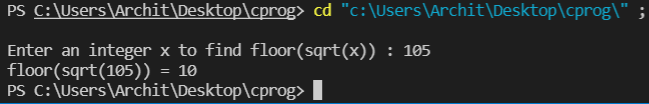
    }

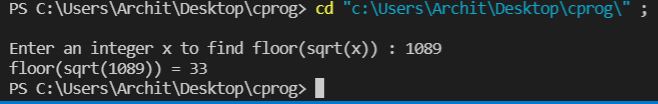
}

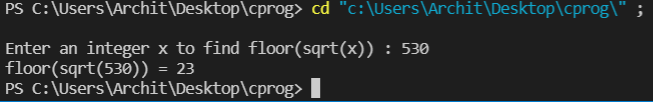
***Output***

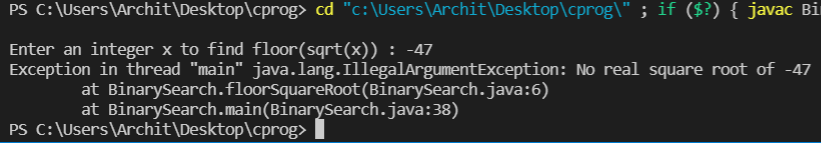




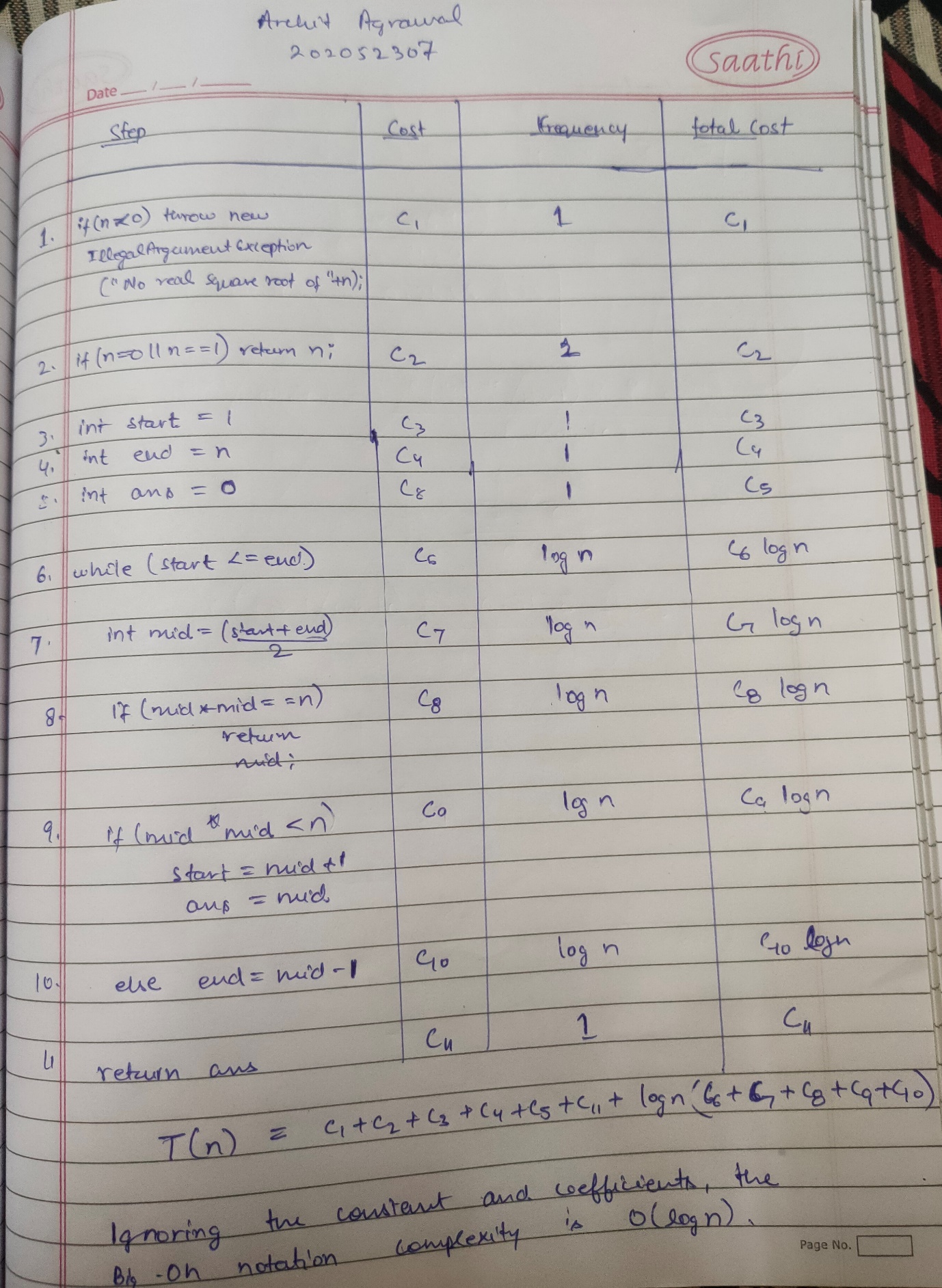








***Time Complexity Analysis***

******

The overall time complexity is thus O(log n).