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

```java
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

```
PS C:\Users\Archit\Desktop\cprog> cd "c:\Users\Archit\Desktop\cprog\" ; 

Enter an integer x to find floor(sqrt(x)) : 0
floor(sqrt(0)) = 0
PS C:\Users\Archit\Desktop\cprog>
```

```
PS C:\Users\Archit\Desktop\cprog> cd "c:\Users\Archit\Desktop\cprog\"

Enter an integer x to find floor(sqrt(x)) : 1
floor(sqrt(1)) = 1
PS C:\Users\Archit\Desktop\cprog>
```

```
PS C:\Users\Archit\Desktop\cprog> cd "c:\Users\Archit\Desktop\cprog\" ;

Enter an integer x to find floor(sqrt(x)) : 105
floor(sqrt(105)) = 10
PS C:\Users\Archit\Desktop\cprog>
```

```
PS C:\Users\Archit\Desktop\cprog> cd "c:\Users\Archit\Desktop\cprog\" ;

Enter an integer x to find floor(sqrt(x)) : 1089
floor(sqrt(1089)) = 33
PS C:\Users\Archit\Desktop\cprog>
```

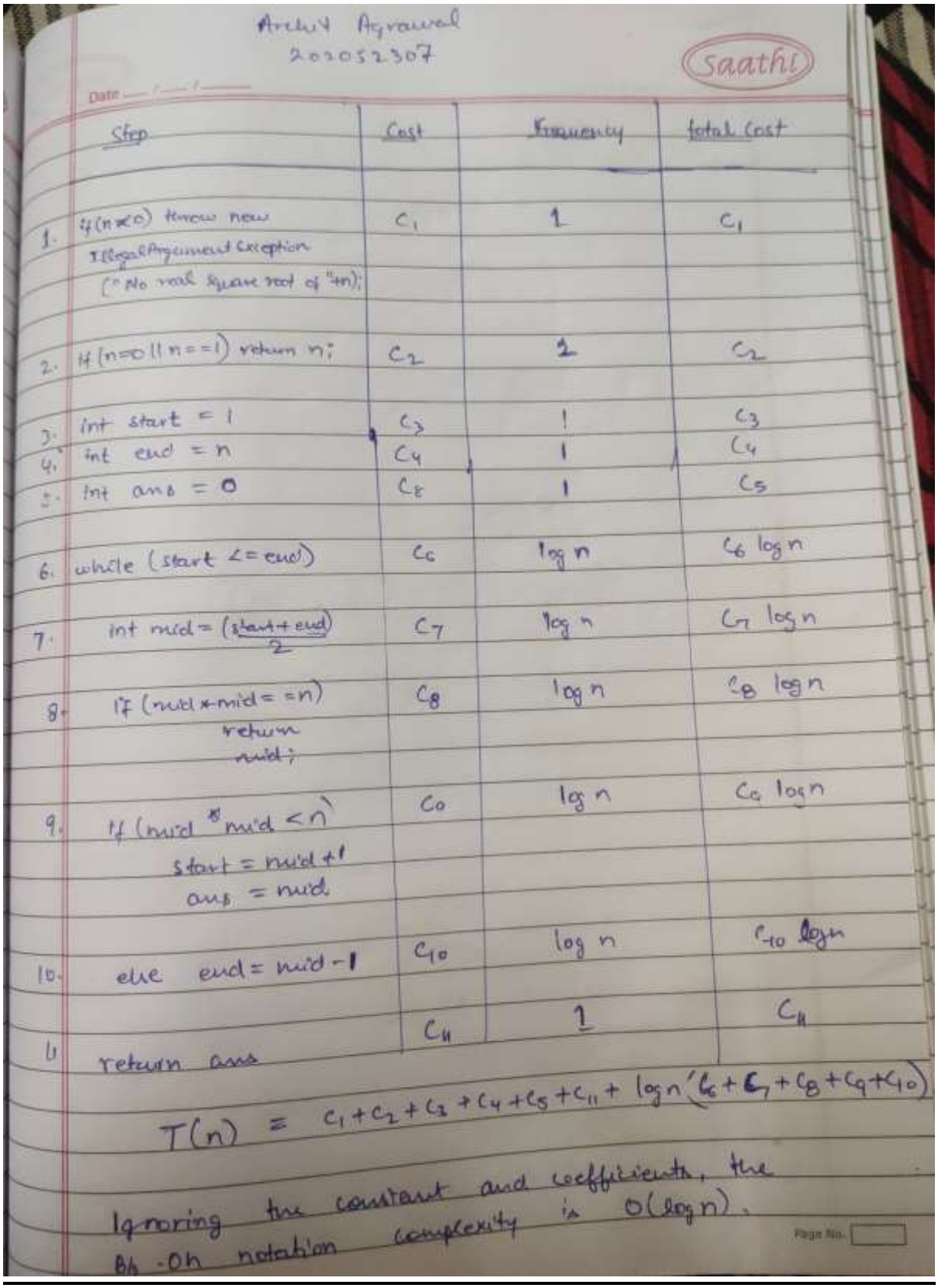
```
PS C:\Users\Archit\Desktop\cprog> cd "c:\Users\Archit\Desktop\cprog\" ;

Enter an integer x to find floor(sqrt(x)) : 530
floor(sqrt(530)) = 23
PS C:\Users\Archit\Desktop\cprog>
```

```
PS C:\Users\Archit\Desktop\cprog> cd "c:\Users\Archit\Desktop\cprog\" ; if ($?) { javac Bi

Enter an integer x to find floor(sqrt(x)) : -47
Exception in thread "main" java.lang.IllegalArgumentException: No real square root of -47
        at BinarySearch.floorSquareRoot(BinarySearch.java:6)
        at BinarySearch.main(BinarySearch.java:38)
PS C:\Users\Archit\Desktop\cprog>
```

Time Complexity Analysis



Archit Agrawal
202052307

Saathi

Date

	Step	Cost	Frequency	total Cost
1.	if(n<0) throw new IllegalArgumentException ("No real square root of "+n);	c_1	1	c_1
2.	if (n==0 || n==1) return n;	c_2	2	c_2
3.	int start = 1	c_3	1	c_3
4.	int end = n	c_4	1	c_4
5.	int ans = 0	c_5	1	c_5
6.	while (start <= end)	c_6	log n	$c_6 \log n$
7.	int mid = (start+end)/2	c_7	log n	$c_7 \log n$
8.	if (mid*mid == n) return mid;	c_8	log n	$c_8 \log n$
9.	if (mid*mid < n) start = mid+1 ans = mid	c_9	log n	$c_9 \log n$
10.	else end = mid-1	c_{10}	log n	$c_{10} \log n$
11.	return ans	c_{11}	1	c_{11}

$T(n) = c_1 + c_2 + c_3 + c_4 + c_5 + c_{11} + \log n (c_6 + c_7 + c_8 + c_9 + c_{10})$

Ignoring the constant and coefficients, the BA-Oh notation complexity is O(log n).

Page No.

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