

$10^9 \text{ ops} \rightarrow 1 \text{ sec}$] $3 \times 10^9 \text{ cycles/sec.}$

We can perform 10^9 operations in one second

$t = 10^4$
 $n = 10^9$

```
int t = scn.nextInt();
for(int i = 0; i < t; i++){
    int n = scn.nextInt();

    int count = 0;
    for(int div = 1; div <= n; div++){
        if(n % div == 0){
            count++;
        }

        if(count == 2){
            System.out.println("prime");
        } else {
            System.out.println("not prime");
        }
    }
}
```

Annotations: 10^4 (pointing to $i < t$), 10^9 (pointing to $div \leq n$)

Calculation: $10^4 \times 10^9 = 10^{13}$

NOW AS PER CODE WRITTEN IN WRONG ATTEMPT WE HAD EXCEED THE TIME LIMIT

BECAUSE OUR OUTER LOOP RUNS 10^4 TIMES

WE NEED TO RUN INNER OPERATION LOOP 10^9 MAX TIMES

THUS IF WE SEE THE INNER LOOP RUNS 10^{13} TIMES WHICH IS GREATER THAN GIVEN OPERATIONS

The whiteboard shows two columns of factor pairs. The left column is for the number 24, and the right column is for the number 36. Each pair consists of a circled number multiplied by another circled number, with an arrow pointing from the first to the second. Below the 24 column, there is a box containing the conditions for finding prime factors: $p > \sqrt{n}$ and $q > \sqrt{n}$. The number 24 is written at the top left, and 36 is written at the top right.

Left Side (24):

- 1×24
- 2×12
- 3×8
- 4×6

Right Side (36):

- 1×36
- 2×18
- 3×12
- 4×9
- 6×6

Boxed Conditions:

$$p > \sqrt{n}$$

$$q > \sqrt{n}$$

Left Side (24):

The teacher is listing all the factor pairs of 24:

- 1×24
- 2×12
- 3×8
- 4×6

Then it stops at 5, because:

- After 4×6 , we start repeating the same factors (like 6×4 , 8×3 , etc.).
- So we can **stop checking at $\sqrt{24} \approx 4.9$** .

💡 **Only check up to square root of the number to save time.**

Smart Trick for Prime Check:

If you want to check if a number n is **prime**, you only need to check for divisibility from **2 to \sqrt{n}** .

If no number divides n in that range, it's a prime!

OPTIMIZED SOLUTION

```
for (int i = 1; i <= t; i++) {  
    System.out.print(s:"ENTER NUMBER:- ");  
    int num = input.nextInt();  
  
    int count = 0;  
    for (int j = 2; j <= Math.sqrt(num); j++) {  
        if (num % j == 0) {  
            count++;  
            break;  
        }  
    }  
    System.out.println(count);  
  
    if (count == 0) {  
        System.out.println(x:"PRIME");  
    } else {  
        System.out.println(x:"NOT PRIME");  
    }  
}
```

LET CONSIDER OUTER LOOP TAKES 10^4 TIMES

AND INNER LOOP CAN TAKE MAX 10^9 TIMES BUT AS WE DO SQRT IT TAKES NOW 10^3 TIMES

THUS MAX IT TAKES 10^7 TIMES THUS IT DOESN'T EXCEED TIME

WORK FLOW:-

If user enters 7:

- Loop checks: 2, 3 (since $\sqrt{7} \approx 2.6$)
- 7 is not divisible by 2 or 3 \rightarrow count stays 0 \rightarrow PRIME

If user enters 9:

- Loop checks: 2, 3 (since $\sqrt{9} = 3$)
- 9 is divisible by 3 \rightarrow count becomes 1 \rightarrow NOT PRIME