**Primality Testing**

Given a positive integer n, we will print all the prime numbers from 1 to n. A prime is a natural number greater than 1 that has no positive divisors other than 1 and itself.

Input:  n = 15

Output:  2 3 5 7 11 13

Input:  n = 3

Output: 2 3

Find the Worst Case time complexity for the following codes:

1. **Naive approach:**

|  |
| --- |
| **bool prime[n]= {0};**  **void isPrime(int n)**  **{**  **for(int i=2;i<=n;i++){**  **int cnt = 0;**  **for(int j=2;j<i;j++){**  **if(i%j==0)cnt++;**  **}**  **if(cnt==0)prime[i]=1;**  **}**  **for(int i = 2; i <= n; i++) {**  **if(prime[i])**  **System.out.print(i + " ");**  **}**  **}** |

1. **Optimal Sieve**

|  |
| --- |
| **void sieveOfEratosthenes(int n)**  **{**  **boolean prime[] = new boolean[n+1];**  **for(int i=0;i<n;i++)**  **prime[i] = true;**    **for(int p = 2; p<=sqrt(n); p++)**  **{**  **// If prime[p] is not changed, then it is a prime**  **if(prime[p] == true)**  **{**  **// Update all multiples of p**  **for(int i = p\*p; i <= n; i += p)**  **prime[i] = false;**  **}**  **}**    **// Print all prime numbers**  **for(int i = 2; i <= n; i++)**  **{**  **if(prime[i] == true)**  **System.out.print(i + " ");**  **}** |

**Recursion Tree Time Complexity**

Find the Worst case time Complexity of the following recursive functions

1. T(n) = T(n/2)+n-1, T(1) = 0

2. T(n) = T(n-1)+n -1, T(1) = 0

3. T(n)=T(n/3)+2T(n/3)+n

4. Proof that for T(n)=2T(n/2) + n^2, the worst case complexity will be n^2.

**Pseudocode to Coding**

Represent the following pseudocode with java coding

