Today's agenda

- IsPrime
- Sieve of eratosthenes
- Smallest Prime factor of all numbers 1 to n.
- factors for multiple queries
- Co-Prime

Q) isPrime

↳ Check whether a given number is Prime or not.

N=10: Not Prime

N=7: Prime

Idea

→ Count of factors == 2 → Prime no.

: i → 1 to N → T.C: O(N)

: i → 1 to $\sqrt{N}$ → T.C: O($\sqrt{N}$)

N = 100

i

1 x 100

2 x 50

4 x 25

5 x 20

10 x 10

---

20 x 5

25 x 4

50 x 2

100 x 1

Q) Sieve of eratosthenes

↳ Print all primes from 1-N.

Ex: N=10: 2 3 5 7

//Idea1

↳ iterate from 2 to N and for every number check isPrime().

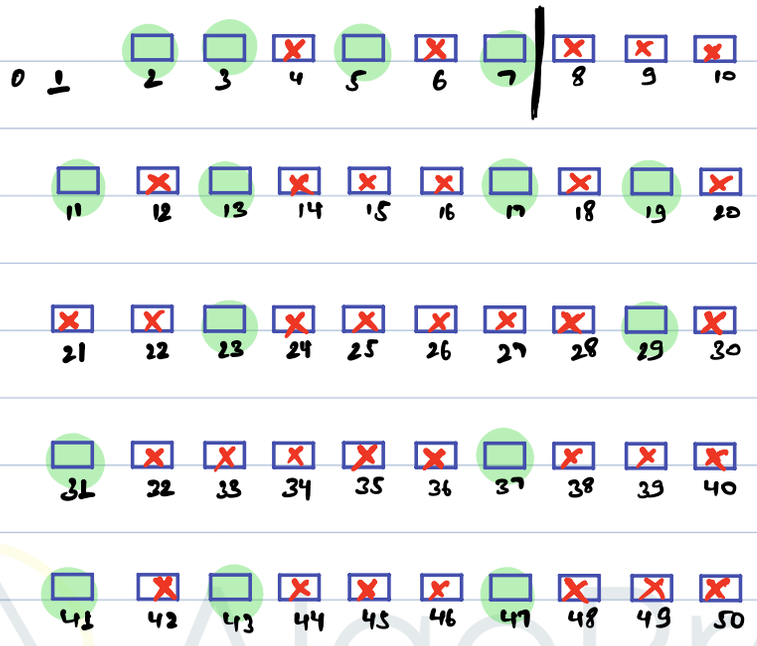
T.C: $O(N\sqrt{N})$ S.C: $O(1)$

Idea 2

N = 50

→ ☐ → Prime → true

☒ → Not Prime → false



$N = 50 \rightarrow \sqrt{50} = 7$

2*2	~~3*2~~	4	~~5*2~~
2*3	3*3	⋮	~~5*3~~
2*4	3*4	|	~~5*4~~
2*5	3*5	4*5	5*5
2*6			5*6
2*7	|		5*7
⋮	⋮		
|	|		
|			

//Psuedo code

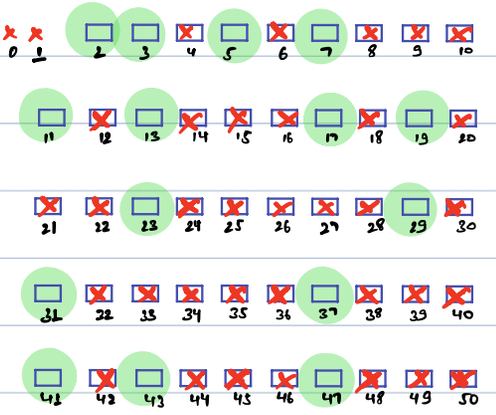
```
void allPrimes (int N){                                   <- 10^8
    boolean [] P = new boolean[N+1];
    Arrays.fill(P, true);
    P[0] = P[1] = false;

    for (i = 2; i*i<=N ; i++ ){                           <- i<=N
        if (P[i] == true){
            for(int j = 2*i; j<=N; j+=i){                 <- i*i
                P[j] = false;
            }
        }
    }

    for(int i=1; i<=N; i++ ){
        if (P[i] == true){ Print(i); }
    }
}
```

T.C: O(NloglogN) ≈ O(N)

S.C: O(N)

```
boolean [] P = new boolean[N+1];
Arrays.fill(P, true);
P[0] = P[1] = false;

for (i = 2; i*i<=N; i++) {
    if (P[i] == true) {
        for (int j = 2*i; j<=N; j+=i) {
            P[j] = false;
        }
    }
}
```

→ □ → Prime → true
☒ → Not Prime → false




```
for (int i=1; i<=N; i++) {
    if (P[i] == true) { Print(i); }
}
```


T.C: $\frac{N}{2} + \frac{N}{3} + \frac{N}{5} + \frac{N}{7} \cdots -$

$$N\left(\frac{1}{2} + \frac{1}{3} + \frac{1}{5} + \frac{1}{7} \cdots -\right)$$

→ log logN

↳ convergence / divergence

$= N \log\log N \approx N$ $\qquad 10^9 \approx 2^{30}$

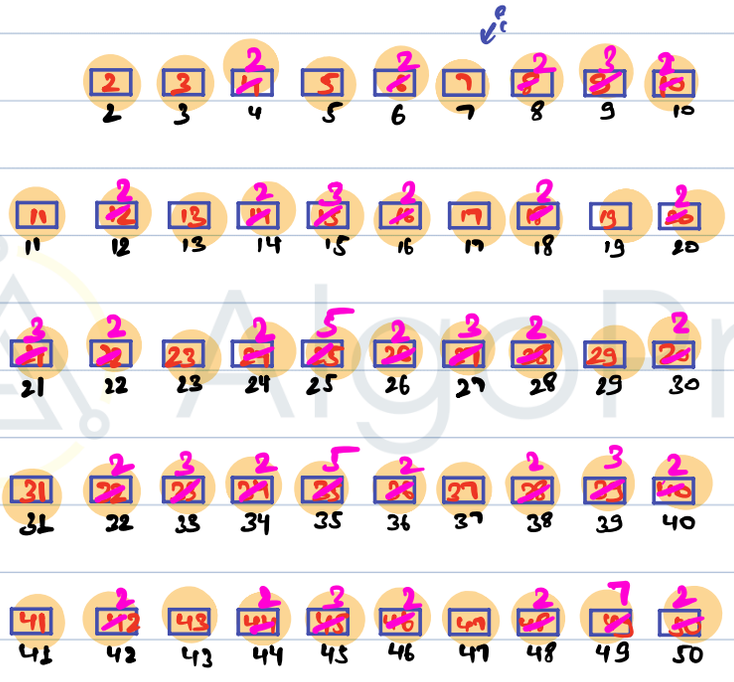
$N = 10^9 \rightarrow \log\log N \approx \log\log 10^9 \approx \log\log 2^{30} \approx \log_2 30 \approx 5$

Q) Smallest Prime factors

To find smallest prime factor for all numbers 1-N.

N=10

1	2	3	4	5	6	7	8	9	10
-1	2	3	2	5	2	7	2	3	2



//Pseudo Code

```
int[] SmallestPrimeFactor(int N){
    int[] spf = new int[N+1];

    for(int i=1; i<=N; i++){
        spf[i] = i;
    }

    spf[0] = spf[1] = -1;

    for(int i=2; i*i<=N; i++){
        if(spf[i] == i){
            for(int j=i*i; j<=N; j+=i){
                spf[j] = min(spf[j], i);
            }
        }
    }

    return spf;
}
```

T.C: $O(N \log\log N)$ S.C: $O(N)$

Q) Prime factors for multiple queries

↳ Given N and multiple queries of integers in the range 1-N, you have to Prime factorize for every query.

N = 50

Q: 10 → 2 5

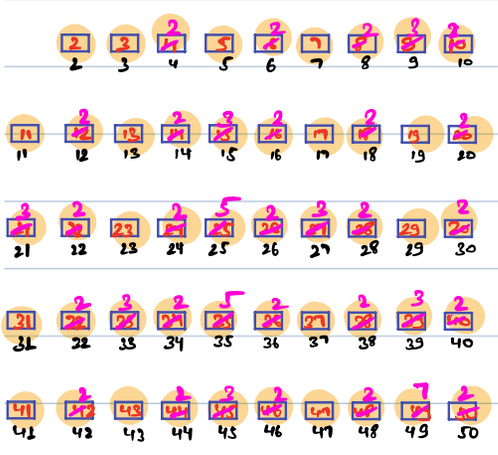
24 → 2 2 2 3

//idea1

↳ Check for Prime factors for every query.

T.C: O(√N * Q)

//idea2



N = 50

10 → 2 5

24 → 2 2 2 3

2	24
2	12
2	6
3	3
	1

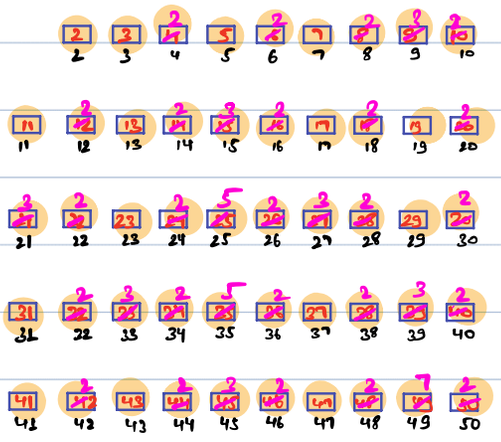
//Pseudo Code

```
void primefactorization(int N, int[] Q){

    int[] spf = SmallestPrimefactor(N);

    for(int i=0; i<Q.length; i++){
        int n = Q[i];

        while(n > 1){
            Print(spf[n]);
            n = n/spf[n];
        }
        System.out.println();
    }

}
```

T.C: $O(N \log\log N) + O(Q * \log N)$ S.C: $O(N)$

Q: | 10 | 24 |

```
for (int i=0; i< Q.length; i++){
    int n = Q[i];
    while (n > 1) {
        Print (spf[n]);
        n = n / spf[n];
    }
    System.out.println();
}
```



n = 24 12 6 3 1

2 5

2 2 2 3

Co-Prime

↳ Two numbers (a,b) is a co-Prime if $gcd(a,b) = 1$.

ex: $(2, 3) = 1$ → A co-Prime

$(5, 7)$ → Co-Prime (Prime, Prime)

$(7, 20)$ → Co-Prime (Prime, not Prime)

$(9, 10)$ → Co-Prime (not Prime, not Prime)

Q) you have N number given → {1, 2, 3 . . . — N} → {N is even}

Your task is to create N/2 pairs such that each pair is co-prime. if it is not possible return -1.

N = 10 → {1, 2, 3 . . . 10}

1 2
3 4
5 6
7 8
9 10

→ gcd(n, n+1) = 1