- Introduction to Prime Nos.
- Get all primes from 1 to N
- Sieve of Eratosthenes
- Get no. of factors
- Sorted Permutation Rank

Prime No.

↳ have only 2 factors
1 and number itself

1 ✗	2 ✓	7 ✓	10 ✗
1	1 2	1 7	1, 2, 5, 10

Check if a number is prime or not ?

Count no. of factors

Approach 1: i → 1 to N <

```
if (N % i == 0)
    cnt++
```

TC: O(N)

return (cnt == 2)

Approach 2: Factors come in pairs

24

i	N/i
1	24
2	12
3	8
4	6
6	4
8	3
12	2
24	1

$i \leq N/i$

$\Rightarrow i^2 \leq N$

$\Rightarrow i \leq \sqrt{N}$

```
bool checkPrime(int N){
    int cnt = 0
    for(i=1 ; i <= sqrt(N) ; i++){        // i*i <= N
        if(N%i==0){
            if(i==N/i)      cnt+=1
            else            cnt+=2
        }
    }
    return (cnt==2)
}
```

TC : $O(\sqrt{N})$

Given a number N, print all prime nos. from 1 to N

ans

N = 10 → 2, 3, 5, 7

N = 20 → 2, 3, 5, 7, 11, 13, 17, 19

BF: Go to every num from 1 → N and check whether its prime or not

```
for (i = 1 ; i <= N ; i++) {
    if (checkPrime(i) == true)
        print(i)
}
```

TC : O(N√N)

SC : O(1)

N = 10

0	1	2	3	4	5	6	7	8	9	10
F	F	T	T	T	T	T	T	T	T	T

Sieve of Eratosthenes

N = 50

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50

Ass: Every no. is prime

	factors
10 →	1, 2, 5, 10

i	Multiples of i
2 →	4, 6, 8, ...
3 →	9, 12, 15, 18 ....
5 →	25, 30, ...
7 →	49, ...

i → i*i

1. Assumed all nos. are prime (except 1)
2. If a no. is prime, then we use to eliminate its multiple
3. Keep repeating same, leftover at end list of primes

```
bool isPrime[N+1] = {true}
isPrime[0] = false ; isPrime[1] = false ;

for (i=2 ; i ≤ √N ; i++) {
    if (isPrime[i] == true) {
        for (j=i*i ; j ≤ N ; j=j+i) {
            isPrime[j] = false
        }
    }
}

for (i=2 ; i ≤ N ; i++) {
    if (isPrime[i] == true) print(i)
}
```

num → 1st multiple

$i \rightarrow i^2 \leq N$

$\Rightarrow i \leq \sqrt{N}$

i (2 → $\sqrt{N}$)	j (i*i → N)
2	≈ $\frac{N}{2}$ iterations
3	≈ $N/3$ iterations
4	X
5	≈ $N/5$ iterations
6	X
7	≈ $N/7$
⋮	
$\sqrt{N}$	(N → N) 1 iteration

TC : $\frac{N}{2} + \frac{N}{3} + \frac{N}{5} + \frac{N}{7} + \dots$

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

↑ sum of reciprocal of primes

$= N(\log(\log N))$

TC : $O(N \log\log(N))$

SC : $O(N)$

$N = 2^{32}$ $\quad \log_2 \log_2 2^{32}$

$\Rightarrow \log_2 32 = 5$

Given N, count no. of factors for all nos. from 1 to N.

N = 10

	0	1	2	3	4	5	6	7	8	9	10
o/p:	—	1	2	2	3	2	4	2	4	3	4
factors		1	1 2	1 3	1 2 4	1 5	1 2 3 6	1 7	1 2 4 8	1 3 9	1 2 5 10

BF : Go to every num from 1 → N, count factors $\quad$ TC : $O(N\sqrt{N})$

Optimized : Sieve $\quad$ N = 20

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

11	12	13	14	15	16	17	18	19	20
2	~~2~~ ~~3~~ ~~4~~ ~~6~~ 6	2	~~2~~ ~~7~~ 4	~~3~~ ~~5~~ 4	~~2~~ ~~4~~ ~~8~~ 5	2	~~2~~ ~~3~~ ~~6~~ ~~9~~ 6	2	~~2~~ ~~4~~ ~~5~~ ~~10~~ 6

```
int cof[N+1] = {2}
cof[0] = 0 ; cof[1] = 1
for (i=2 ; i<=N/2 ; i++) {
    for (j=2*i ; j<=N ; j=j+i) {
        cof[j]++
    }
}
```

$i \rightarrow j$

$2i \le N$

$\Rightarrow i \le N/2$

i (2 → N/2)	j (2i → N)
2	N/2
3	N/3
4	N/4
5	N/5
⋮	
N/2	1

TC: $\frac{N}{2} + \frac{N}{3} + \frac{N}{4} + \frac{N}{5} + \ldots + 1$

$= N\left(\frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} + \ldots + \frac{1}{N}\right)$

TC $\approx O(N \log N)$

10:25

Given a string A containing distinct characters. Find rank of this string among all permutations when sorted in lexicographical order.

↓ dictionary

Ex "bca"

ans = 4

abc
acb
bac
bca
cab
cba

BF: Generate all possible permutations and then do a string match on the permutation

TC: $O(N! \times N)$

$20! \approx 2 \times 10^{18}$

Optimized:

ex: a f c h e

ans = 0

e 4 3 2 1

+ 4!

a f c h e

a e _ _ _ 3!

d e _ _ _ 3!

+ 2 × 3!

a f c h e

+ 0

a f c h e

a f c e _

+ 1

a f c h e

c ≡

No. of string before = 37

Rank = 38

dc =

dc =

df

Ex: b f a h ans = 9

Total permutations = 4! = 24

ans = 0

b f a h

a _ _ _ + 3!

b f a h

b a _ _ + 2!

b f a h + 0

b f a h STOP

String before = 8

Rank = 9

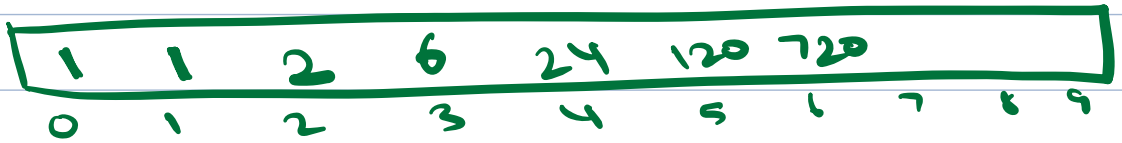
a { abfh, abhf, afbh, afhb, ahbf, ahfb } 6

ba { bahf, bafh } 2

bfah

factorial(i) = i x (i-1) x (i-2) x ... 1

= i x factorial(i-1)



fact → 1 to 9

```
int N = s.length()
int fact[N+1]
fact[0]=1
for (i = 1 ; i <= N ; i++){
    fact[i] = i * fact[i-1]
}
```

```
int rank (string s){
    int ans=0
    for (i=0 ; i <= N-2 ; i++){
        char ch = s[i]
        // all chars smaller than ch
           on the right
        int smaller=0
        for (j= i+1 ; j < N ; j++){
            if (s[j] < ch){
                smaller++
            }
        }
```

ans += smaller * factorial[N-i-1]

return ans + 1

N = 7

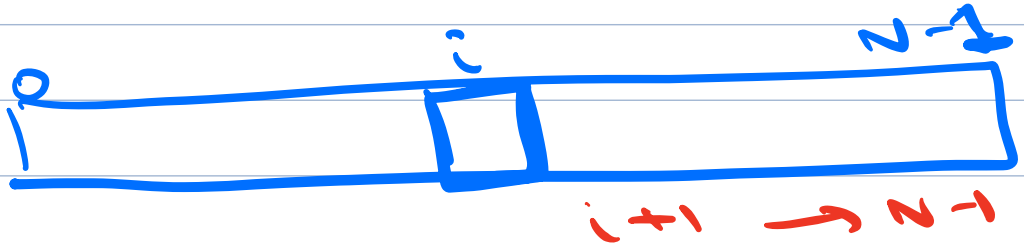
i

0 1 2 3 4 5 6

d e g h a c z

deg — 3!

h c z — 3!



length = N-1 - (i+1) + 1

= N-1-i-1+1

= N - i - 1

TC : O(N²)

SC : O(N)

↓

factorial array