DATE

PAGE

* Sieve of Eratosthenes :- (O(nlog(log N)))
used to find prime no. with least time complexicty.

```
class SieveOfEratosthenes
{

    void sieveOfEratosthenes(int n)
    {
        // Create a boolean array "prime[0.....n]"
        // and initialize all entries it as true
        // A value in prime[i] will finally be
        // false if i is not a prime, else
        //    true

    boolean prime[] = new boolean[n+1];
    for(int i=0; i<=n; i++).
        prime[i] = true;

    for ( int p=2; p*p <=n; p++).
    {
        // If prime[p] is not change the it
        // is a prime
        if (prime[p] == = true)
        {
            for(int i=p*p; j<=n ; i=i+p).
        }
    }

    for(int i=2 ;  i<=n) i++)
    {
        if (prime[i] == = true)
```

```
        S.o.println (j + " ");
    }
}.
```

* Euler's Totient function $\phi(n)$ for an input n is the count of numbers in (1, 2, 3 ---- n) that are relatively prime to n . i.e, the numbers whose GCD (Greatest Common Divisor) with n is 1.

→ If no. is prime,

prime = prime - 1

→ If no. is non-prime,

$$p^n = p^n - p^{n-1}$$

Q.1 $\phi(27)$

Sol: $\phi(27)$

3	27
3	9
3	3
	1

DATE

PAGE

$\phi(3^3)$

$= 3^3 - 3^{3-1}$

$= 27 - 9$

$= 18$

6^{γ} (9)

gcd (1, 27) ✓
gcd (2, 27) ✓
gcd (3, 27) ✗
gcd (4, 27) ✓
gcd (5, 27) ✓
gcd (6, 27) ✗
gcd (7, 27) ✓
gcd (8, 27) ✓
gcd (9, 27) ✗
gcd (10, 27) ✓
gcd (11, 27) ✓
gcd (12, 27) ✗
gcd (13, 27) ✓
gcd (14, 27) ✓
gcd (15, 27) ✗
gcd (16, 27) ✓
gcd (17, 27) ✓
gcd (18, 27) ✗
gcd (19, 27) ✓
gcd (20, 27) ✓
gcd (21, 27) ✗
gcd (22, 27) ✓
gcd (23, 27) ✓
gcd (24, 27) ✗
gcd (25, 27) ✓
gcd (26, 27) ✗

~~gcd (27~~

$n = 5$

1, 2, 3, 4

GCD(1,5) = 1
GCD(2,5) = 1
GC. =

$n = 6$

1, 2, 3, 4, 5

1, 6 = 1
2, 6 = 2 ✗
3, 6 = 3 ✗
4, 6 = 2 ✗
5, 6 = 1 ✓

BAMBOOS
grow like a bamboo

DATE _____

PAGE _____

→ import java.io.*;

Class Euler { {
{ static int gcd(int a, int b).

if (a == 0)

return b;

return gcd(b % a, a);

}

static int phi (int n)

{

int result = 1;

for(int i = 2; i < n; i++).

if (gcd(i, n) == 1)

result ++;

return result;

}

p.s.v.d (string[]args).

{

int n;

for(int i = 1, n <= 10, n++).

S.O.p.ln ("phi (" +n+ ") = " + phi(n))

}

On Incremented

* Segmented Sieve :- (If n is very large)

The idea of segmented sieve is to divide the range [0 ... n-1], in different segments & Compute primes in all segment one by one.

DATE
PAGE

* Congruent Eqn(s) :- (Remainder Theorem)

$X \% [num[o]) = (rem[o])$

↳ we will use this in our programming.

different congruent equations with one variable but different moduli (%).

$X \equiv a_1 \pmod{m_1}$
$X \equiv a_2 \pmod{m_2}$
$X \equiv a_3 \pmod{m_3}$

[a_1, a_2, a_3 are remainders
m_1, m_2, m_3 are divisors.]

Formula for X :-

$$X \equiv [a_1 M_1 \cdot M_1^{-1} + a_2 M_2 \cdot M_2^{-1} + \cdots + a_n M_n \cdot M_n^{-1}]$$

Moduli M

$M = m_1 \cdot m_2 \cdot m_3$.

$M_1 = \frac{M}{m_1}$

$M^{-1} \Rightarrow M_1 \cdot M_1^{-1} = 1 \bmod m1.$

$\frac{X}{m_1} = R(a_1)$

Q.1.

Sol:- num = {3, 4, 5}

rem = {2, 3, 1}.

$X \equiv 2 \pmod 3$ (a)

$X \equiv 3 \pmod 4$

$X \equiv 1 \pmod 5$ (m)

$$X = (a_1 m_1 \cdot m_1^{-1} + a_2 m_2 \cdot m_2^{-1} + a_3 m_3 m_3^{-1}) \bmod m$$

$a_1 = 3$	$m_1 = 3$
$a_2 = 3$	$m_2 = 4$
$a_3 = 1$	$m_3 = 5$

$m = m_1 \cdot m_2 \cdot m_3 = 60$

$$m_1 = \frac{m}{m_1}$$

$m_1 = 20$

$m_2 = 15$

$m_3 = 12.$

$m_1^{-1} \Rightarrow m_1 m_1^{-1} = 1 \bmod 3$

$\therefore m^{-1} \in N.$

For $m_1^{-1} = 1$

$20 \times 1 = 1 \bmod 3,$

$20^{1} = 1 \bmod 3$ (a) (m)

$X = R \bmod div$

$\frac{20}{3} = R(1)$

$\therefore M_1^{-1} = 2$

$M_2^{-1} = 3$

$M_3^{-1} = 3$

$X = (2 \times 20 \times 2 + 3 \times 15 \times 3 + 1 \times 12 \times 3) \bmod 60$

$= (80 + 135 + 36) \bmod 60$

$= 260 \bmod 60.$

$X = 260 \% 60$

$= 11$

* Alice APPLE Tree :-
Minimum no. of apples to be collected from trees to guarentee M red apples.

* Euclidian Algorithm :- ([redacted])

```
import java.util.Scanner;
public Class Euclid
{
    public static int Euclid(int x, int y) {
        if (x==0 || y==0) {
            return 1;
        }
        if (x<y) {
            int t = x
            x = y
            y= temp;
        }
        if (x%y==0) {
            return y;
        }
        else {
            return Euclid(y, x%y).
        }
    }.

p.s.v.m(String[] args).
    S.o.p.ln("result!" + Euclid(10,150));
```

```
→ Static int findMinX (int num [], int rem[], int k).
{
      int x = 1

      while (true).
      {

            int j ;
            for (j = 0; j<k; j++).
                  if (x % num [j] != rem [j]).
                  break;

                  if (j == k).
                        return x;

                  x ++
```

* Strobogrammaticis numbers :-
When rotating 180° is again change to its original position.

9 9 0
IV O
90
σ0
d0

eg : 1
— — 90°
1 — 180   yes

eg. 8
∞ — 90°
8 - 180°.   Yes.

9006
6009
9006

DATE

PAGE

eg. 5.

5n - 90°

5 - 180° No.

∴ In no. 0 to 9.

2, 3, 4, 5, 7 not strobogrammatic's no.

(6,9)

& 0, 1, 6, 9, 8, strobogramic numbers.

→ def strobogrammaticis_num(n):

result = numdef(n,n);

return result

def numdef(n, length).

DATE
PAGE

* Binary Palindrome :-

```
public class Example {
    p.s.v.m (String [] args).

long num = 4, n1;
long reverse = 0;
n1 = num;
while (n1 > 0)
{

    reverse <<= 1;
if ((n1 & 1) == 1)
 reverse ^= 1;
n1 >>= 1.

 S.O.pln (n1 + " " + reverse);
}

if (num == reverse) {

S.O.pln (" is Palindrome").
}
else
 {
 S.O.pln (" is not palindrome.").
 }
}
```

Short circuit AND is &&

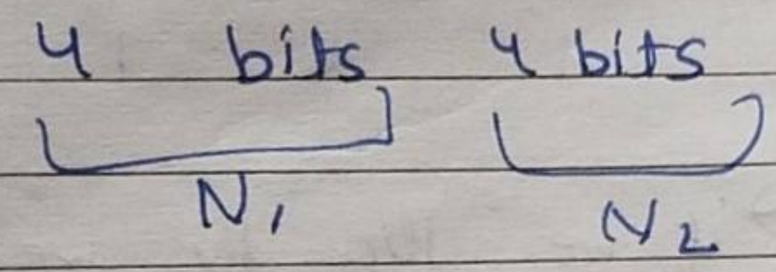
DATE

PAGE

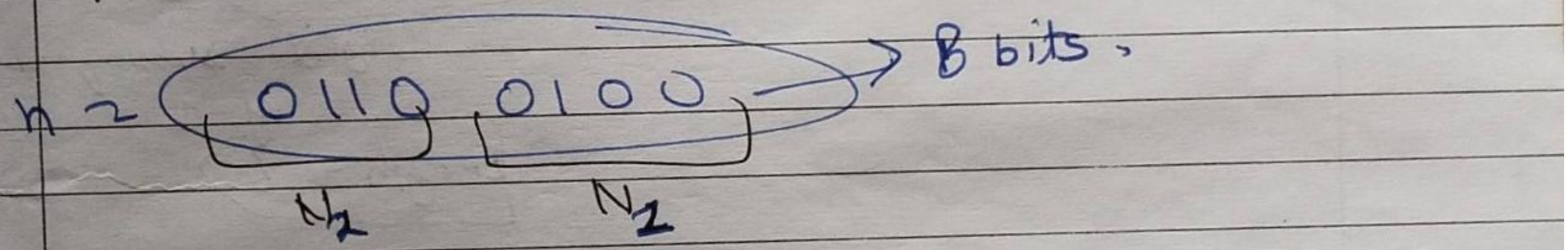
* Swapping 2 nibble in a byte :-

bit = 0, 1.
nibble = 4 bits.

Byte = 8 bits.



Ex n = 100.



n = (n & 0x0F) << 4.

0110 0100
0000 1111

0000 0100

&

n = n & 0XF0 >> 4.

Multiplier → M
mut

$7 \times 3 \Rightarrow 21$

8 4 2 1
7 → 0111

16 8 4 2 1
21 → 0001 0101

$Q_3\ Q_2\ Q_1\ Q_0\ Q_{-1}$
3 → 0011

Booth

* Multiple Algorithm :-

A_c - Accumulation ⇒ 0.
M → 7.
Q → 3. n → no. of operation
Q_{-1} → 0. $S_c(n) = n-1$

n == 0
n ≠ 0

A_c	Q.	Q_{-1}	n = 4.
0000	0011	0	4.
1001	0011 (1,0)	1	
1100	1001	1.	3
1110	0100	1	2.
0101	0100	1	
0010	1010	0	1
0001	0101	0	0

if (1,0)
$A_c = A_c - M$.

$-M = 2$'s of M.
$= A_c + (2$'s of M)

7 → 0111
1's → 1000
 +1
2's → 1001
-7. 1001
 0000
 1001

If (0,1)
$A = A_c + M$.

1110
0111
1|0 | 01

A_c		Q.	
128 64 32	16	8 4 2 1	
0 0 0	1	0 1 0 1	

16 + 4 + 1 ⇒ 21

7×3 = 21

DATE

PAGE

* Karatsuba algorithm:- (1960)

(Recursion algo):-

① divide & conqueror method (If 2 or more digit)

② find a*c, b*d, a+b * c+d.

③ Formula:-

$$a*c*10^{2half} + (ac+bd)*10^{half} + bd.$$

④ For single digit, $n = 2^1$

$$half = \frac{n}{2} = 1.$$

$$ac+bd \Rightarrow (a+b)(c+d) - ac - bd$$

33 - 5 - 12

⇒ 16.

③ - ② - ①

n=2

⇒ 1 2 3 4 , 5 6 7 8

a=12 b=34 c=56 d=78

ac

bd

(a+b)(c+d)

12 , 56

a=1 b=2 c=5 d=6

34 , 78

n=2

ac

bd (a+b)(c+d)

1,5

2,6

3, 11

↳5

↳12

↳33

1

2

3

3 - 2 - 2

Same as

If 2 digit we divide & conqueror

DATE

PAGE

* Longest subsequence of 1 after flipping a bit.

n = 59.

32	16	8	4	2	1
1	1	0	1	1	1

(1) if we flip this we get

1 1 1 1 1 1

longest subsequence of 1

length = 6

DATE _____

PAGE _____

∈ Leader algorithm :- arr[] = {16, 17, 4, 5, 2}

L=2

i) default Leader = n-1. L=17

ii) from left to right ⇒ L / 16 — L > 16 > L=5

from right to left. L < element

* Lexicographical Palindrome :-

Suppose we have a string s that is palindrome So, we have to change one character such that S is no longer a palindrome & it is lexicographically smallest.
So, if the input is like s = "level", then the output will be "clvle", as we can change the first "l" to "e" to get the lexicographically smallest string that is palindrome.

→ Lexical Order :-

Ex.

(1) civic
↳ n = 5 (odd)

i) Sort the word, ccivi

ii) write the occurence, c = 2
i = 2
v = 1

only 1 character should have odd frequency, otherwise palindrome not possible.

DATE _____

PAGE _____

* **Block swap Algorithm :-** O(n)

The block swap algorithm for array rotation is an effecient algorithm that is used for array rotation.

Ex. 0 1

arr[] = {1 2} 3 4 5 6 7 8 , K = 3,

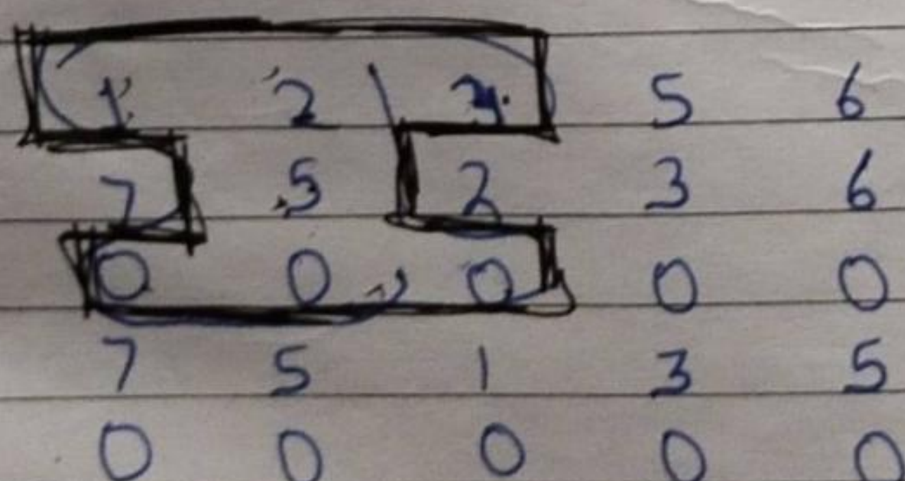
0,1

O/P 34567812

* **Maximum Product Subarray :-**

Given an array that contain both positive & negative integers, find the product of the maximum product subarray. Expected time complexity O(n) & only O(1) extra space can be used.

* **Maximum sum of an Hourglass :-**



(12)

we will check for ⧖ & find max. sum

DATE
PAGE

* Maximum Equilibrium :-

Maximum equilibrium sum in an Array
Given an array Arr[]. Find maximum value of prefix sum which is also suffix sum for index i in Arr[].

Ex. arr[] = {-1, 2, 3, 0, 3, 2, -1}
Output :- 4.

Prefix sum of arr [0 ... 3] = Suffix sum of arr [3 ... 6].

* Majority element :-

The majority element is the element that appears more than ⌊n/2⌋ times.

DATE _____

PAGE _____

③ MaMa

n = 4. (even).

a a m m.

a = 2

m = 2

a	a
m	m

0	1	2	3
a	m	m	a.

* Selection Sort :-

We can create a java program to sort array elements using Selection sort.

In Section sort algorithm, we search for the lowest element & arrange it to the proper location. - We swap the current element with the next lowest number.

Very high time complexity ($O(n^2)$)

DATE
PAGE

→ Code :-

```
void sort (int arr[])
{
    int n = arr.length.

for (int i = 0, i<n, i++)
{
    int min.idx = i;
for (int j = i+1; j<n ; j++)
    if (arr[j] < arr[min.idx])
    min.idx = j;
}

int temp = arr[i].
arr[i] = arr[min.idx]
arr[min.idx] = temp;
}
```

* Quick Sort :- O(nlogn)

i j

0 1 2 3 4 5

24, 9, 29, 14, 19, 27.

i<j (✓) 0<5(✓)

arr[i] ≤ pivot	arr[j] > pivot.	i<j
24 ≤ 24 ✓	27 > 24 ✓	1<4 ✓
i++ // (1)	j-- // (4).	Swap(arr[i], arr[j])
9 ≤ 24 ✓	19 > 24 x	
i++ // (2)		
29 ≤ 24.		

DATE

PAGE

⇒ Alphanumeric

Normal

A11, A2, A3

It sorts ↓

A11, A2, A3.

Natural

↓

A12, A2, A3

↓

In natural sort sequence of digit considered as single digit

∴ O/P A2, A3, A11.

Ex. A(A)20, A30, A40

↓

(65) → ASCII

A(65)20, A30, A40

↓

O/P A30, A40, A(65)20

Start comparing using ASC II, digit.

DATE _____

PAGE _____

* Move spaces / hyphens to front of string traversal.

Ex.

String S = "move space to front of string"
String Str = "move-hyphens-to-front-of-string"

O/P = _____ move space to front of string.
- - - - - - - move hyphens to front of string.

* Manacher's Algorithm :- (T.C = O(N), S.C = O(N))

It is used to find longest palindromic substring in any string.

* Sorted unique permutation
Next Permutation.

* Combination :-

* Manwering algorithm.

DATE
PAGE

Same as above.

c
3 ≤ s

cd
7 ≤ s ✗

d
4 ≤ s

de
9 ≤ s ✗

e
5 ≤ s

ef
11 ≤ s ✗

f
6 ≤ s ✗

g
7 ≤ s ✗

h
8 ≤ s ✗

∴ O/P = (7)

DATE _____

PAGE _____

* Weighted substring :- $(O(N^2))$

Given a string P consisting of small English letters & a string Q consisting of weight of all characters of English alphabet such that for all 'i', $0 \leq Q[i] \leq 9$. The task is to find the total no. of unique substring with sum of weight atmost K.

Ex. P = abcdefgh
Q = 1 2 3 4 5 6 7 8
K = 5.

a	b	c	d	e	f	g	h
1	2	3	4	5	6	7	8.

a = 1
b = 2
c = 3
d = 4
e = 5
f = 6
g = 7
h = 8

Sum of substring $\leq K$.

Consequetive should be there.

ij

a b c d e f g h

a ✓ $1 \leq 5$ ab ✓ $3 \leq 5$ abc ✗ $6 \nleq 5$ break here.

ij

Now a b c d e f g h

b ✓ $2 \leq 5$ bc ✓ $5 \leq 5$ bcd ✗ $9 \nleq 5$ break here

DATE _____

PAGE _____

i ↓ j

Pivot 24 9 19 14 29 27

~~i~~ 24 ≤ 24	27 > 24
i = 1	j = 4.
9 ≤ 24	29 > 24
i = 2	j = 3
19 ≤ 24	14 > 24 (x)
i = 3	
14 ≤ 24	i < j
i = 4.	4 < 3 X
29 ≤ 24 X	Swap(arr[i], pivot)
	pivot = j

14 ~~24~~ 9 19 | 24 | 29 27

p1 > p2

i j

14, 9, 19 | 29 | 27

piv

i j

9, 14, 19, 24, 27, 29.

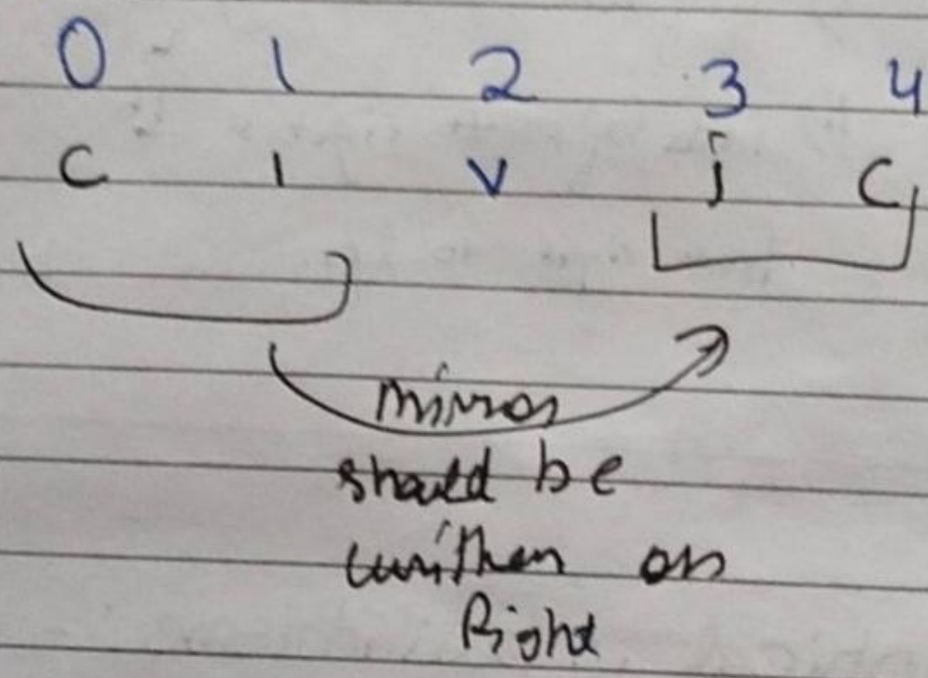
* Natural Sort Order algo :-

A correct natural sorting algorithm states that you order alphabetically but when you encounter a digit you will order that digit and all subsequent digits as a single character.

Natural sorting has nothing to do with sorting by string length first, & then alphabetically when 2 strings have the same length.

DATE
PAGE

then write odd freq. char in the middle.



② Malayalam.

a a a a l l m m y..

a = 4, l = 2, m = 2, y = 1

n = 9 (odd).

Divide the occurence

0	1	2	3	(4)	5	6	7	8.
a	a	l	m	y	m	l	a	a