Chinese Reminder Theorm

→ To know this need idea of

1. Inverse
2. Congruency

1. Inverse

3 modulo 7 = ?

∴ gcd (3, 7) = ?

Find gcd using extended eqn

$7 = 2 \times 3 + 1$ — (1)

$\therefore 1 = 7 - 2 \times 3$ ✗

∴ 3 modulo 7 = −2

$\frac{1}{7} = \frac{7}{7} - \frac{2 \times 3}{7}$

$\frac{1}{7} = -2 \times \frac{3}{7}$

$\therefore \frac{1}{3} = -2$

$\therefore 3^{-1} = -2$

21 modulo 46

∴ gcd (21, 46) = ?

$46 = 2 \times 21 + 4$ — (I)

$21 = 5 \times 4 + 1$ — (II)

$\Rightarrow 1 = 21 - 5 \times 4$

(II) →

$4 = 46 - 2 \times 21$

$\therefore 1 = 21 - 5 \times (46 - 2 \times 21)$

$= 21 - 5 \times 46 + 10 \times 21$

$= 11 \times 21 - 5 \times 46$ ✗

∴ 21 modulo 46 = 11

modulan multipicative Inverse

Unden mod m

$$A \times A^{-1} \equiv 1 \bmod n$$

$$\therefore (A \times A^{-1})/n = 1$$

$$\therefore 3 \times \underline{?} \equiv 1 \bmod 5$$

Here p is the modulo multipicative Inverse of 3 modulo 5

$$\therefore 3 \times (2) \equiv 1 \bmod 5 \quad (3 \times 2/5 = 1)$$

multipicative Inverse

B $2 \times ? = 1 \bmod 11 \longrightarrow ? = 6$

$5 \times 5^{-1} = 1$

$5 \times \frac{1}{5} = 1$

$A \times \frac{1}{A} = 1$

$\frac{1}{5}$ is the multiplicative Inverse of 5

a modulo b

∴ there exist a multipicative Inverse if a, b relatively prime /coprime on

gcd (a, b) = 1

gcd (2, 11) = 1

$AX = 1 (\text{mod } M)$

$X = \frac{1}{A} (\text{mod } M)$

$X = A^{-1} (\text{mod } M)$

X is M1 A modulo m

~~3 modulo 5~~

~~gcd (3,5) = ?~~

~~$\therefore 5 = 3 \times 1 + 2$~~

$3 = 2 \times 1 + 1$

3 modulo 5

gcd (3.5) = ?

$5 = 3 \times 1 + 2$

$3 = 2 \times 1 + 1$

$3 = 1 \times (5-3) + 1$

$3 = 5 - 3 + 1$

$\therefore 1 = -5 + 6$

$= -5 + (2 \times 3)$

$\therefore MI = 2$

$17x = 1 \pmod{43}$

$\therefore x = \frac{1}{17} \pmod{43}$

$\therefore x = 17^{-1} \bmod 43$

$43 = 17 \times 2 + 9$ ⟶ $9 = 43 - 17 \times 2$

$17 = 9 \times 1 + 8$ ⟶ $8 = 17 - 9$

$= 17 - (43 - 17 \times 2)$

$9 = 8 \times 1 + 1$

$\Rightarrow 1 = 9 - 8$

~~$= (17 - 2) - 2$~~

~~$= 9 - 17 + 9$~~

$= 43 - 17 \times 2 - 17 + 43 - 17 \times 2$

~~$= 43 - 3 \times 17 + 43$~~

$= 43 \times 2 - 5 \times 17$

$\therefore x = -5$

Congruency

(27)

a, b

$a \bmod m = r_1$

$b \bmod m = r_2$

∴ if $r_1 = r_2$ ∴ a is congrent of b mod m

$a \equiv b \pmod m$

≡ congrent sign

Theorm - 1

Theorm = 2

$$a = b + km$$

$$a - b = km$$

$$k = \frac{a-b}{m}$$

∴ if $(a-b) \bmod m = 0$ then

$$a \equiv b \pmod m$$

CRT (weak)

→ is used to solve a set of different congruent equations with one variable but different modup which are relatively prime (coprime)

$$X \equiv a_1 \pmod{m_1}$$
$$X \equiv a_2 \pmod{m_2}$$
$$\vdots$$
$$x_1 = a_n \pmod{m_n}$$

Find x

→ CRT states that above equations have a unique solution of the moduli are relatively prime

$$x \equiv \underset{a_1}{2} \pmod{\underset{m_1}{3}}$$

$$x \equiv \underset{a_2}{3} \pmod{\underset{m_2}{5}}$$

$$x \equiv \underset{a_3}{2} \pmod{\underset{m_3}{7}}$$

$x = ?$

(ही x exist if
$\gcd(m_1, m_2) = \gcd(m_2, m_3)$
$\gcd(m_1, m_3) = 1$
or coprime)

Test #A5

$m = m_1 \cdot m_2 \cdot m_3 = 3 \times 5 \times 7 = 105$

$= m / m_1$

$M_1 = \frac{105}{3} = 35$

$Y_1 = 35 \text{ modulo } 3$

$= m_1 \text{ modulo } m_1$

$3 =$

$35 > 3$
can't do

35 modulo 3

$35 = 3 \times 11 + 2$

$3 = 2 \times 1 + 1$

$3 = 3 - 2$

$= 3 - 35 + 3 \times 11$

$= 12 \times 3 - 35$

ans :- 12

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

∴ Here,

$$M = m_1 \times m_2 \times m_3$$

$$= 3 \times 5 \times 7$$

$$= 105$$

$$m_1 = \frac{M}{m_1} \qquad m_2 = \frac{M}{m_2} \qquad m_3 = \frac{M}{m_3}$$

$$= \frac{105}{3} \qquad = \frac{105}{5} \qquad = 105/7$$

$$= 35 \qquad = 21 \qquad = 15$$

$m_1 \times m^{-1} = 1 \bmod m_1$

$35 \times m^{-1} = 1 \bmod 3$

$35 \times 2 = 70/3 = 1$

$\therefore m^{-1} = 35$ modulo 3

$m^{-1} = 2$

$m_2 \times m_2^{-1} = 1 \bmod m_2$

$21 \times m_2^{-1} = 1 \bmod 5$

$21 \times 1 = 21/5 = 1$

$m_2^{-1} = 1$

$m_3 \times m_3^{-1} = 1 \bmod m_3$

$15 \times m_3^{-1} = 1 \bmod 7$

$15 \times 1 = 15 \cdot 1 \cdot t = 1$

$m_3^{-1} = 1$

Extended euclid's

$ax+by=1$

Given $AX + BY = GCD(A, B)$

Find (x, y)

$AX + BY = GCD(B, A\%.B)$

$\therefore x = y_1$ → Hoon

$\therefore y = x_1 - \left\lfloor \frac{a}{b} \right\rfloor y_1$

Ans

$(2, -1)$

$(-1, 2)$

$18x + 30y = 6$

$18x + 30y = GCD(18, 30)$

$\Rightarrow 30x_1 + 18y_1 = GCD(30, 18/30)$

$(30, 18)$

$(1, -1)$

$x_2 = 1$

$y_2 = x_3 - \lfloor \frac{a}{b} \rfloor y_3 \Rightarrow 18x_2 + 12y_2 = GCD(18, 30\%18)$

$(18, 12)$

$= 0 - \lfloor \frac{18}{12} \rfloor 1$

$= -1$

$(0, 1)$

$x_3 = y_4 = 0 \Rightarrow 12x_3 + 6y_3 = GCD(12, 6)$

$y_3 = x_4 - \lfloor \frac{a}{b} \rfloor y_4$

$= 1 - \frac{12}{6} \times 0$

$= 1$

$(1, 0) \Rightarrow 6x_4 + 0y_4 = GCD(6, 0)$

$6x_4 = 6$

$\therefore x_4 = 1$

$y_4 = 0$

Base

	Index i	Quotient q_i	Reminder r_i	x_i $x_i = x_{i-2} - q_i x_{i-1}$	y_i $y_i = y_{i-2} - q_i y_{i-1}$
	0	~~240~~	240	1 $i-2$	0
	1	~~46~~	46	0 $i-1$	1
240, 46	2	240/46 = 5	240 % 46 = 10 (240 − 5×46)	$x_i = 1 - 5\times0$ = 1	$y_i = 0 - 5\times1$ = −5
46, 10	3	46/10 = 4	46 % 10 = 6	0 − 4×1 = −4	1 − 4×0 − 5 = ~~1~~ 21
10, 6	4	10/6 = 1	10 % 6 = 4	1 + 4 = 5	−5 − 21 = −26
6, 4	5	6/4 = 1	6 % 4 = 2	−4 − 5 = −9	21 + 26 = 47
4, 2	6	4/2 = 2	4 % 2 = 0	5 + 18 = ~~13~~ 23	−26 − 2×47 = 120

```
int ext_gcd ( int A, int B, int *X, int *Y)
{
    int x1 = 0, y1 = 1 ;
    int x2 = 1, y2 = 0;

    int x, y, r2, r1, q, r ;

    for ( r2 = A , r1 = B ; r1 != 0 ; r2 = r1,
                                r1 = r
                                x2 = x1
                                x1 = x
                                y2 = y1
                                y1 = y )
    {
        q = r2 / r1 ;
        r = r2 % r1 ;
        x = x2 - (q * x1) ;
        y = y2 - (q * y1) ;
    }
    *X = x2 ;
    *Y = y2 ;
    return r2 ;
```

$x_i = \frac{x_{i-2}}{x_2} - \frac{q_i x_{i-1}}{x_1}$

modular multiplicative Inverse using Ext eculid

$$AX \equiv 1 \pmod{m}$$

$AX + mY = 1$

x modulo Inverse of A modulo m

MI of 3 mod 5

A	B	Remainder R	Quotient Q	T_1	T_2	$T = T_1 - T_2 \times Q$
5	3	5%3 = 2	5/3 = 1	0	1	-1
3	2	1	1	1	-1	2
2	1	0	2	-1	2	-5
1	0	X	X	2	-5	

Ans

Linear Diophantine Equation

$4x + 10y = 8$

$\therefore$ GCD $(4, 10) = 2 = g$; since 2 divides 8

solution exists //

$\therefore \frac{a}{g} = \frac{b}{g} = \frac{c}{g}$; $\frac{4}{2} = \frac{10}{2} = \frac{8}{2} \rightarrow 4$

$\therefore 2x + 5y = 4 \rightarrow (1)$ → Find solution using ext-gcd

$\therefore$ From ext_gcd → $x, y = -2, 1$

$\therefore$ this solution is for $ax + by = 1$

$\therefore$ we need to multiply 4

$\therefore$ $\boxed{(-8, 4)}$.

Ans

```
bool linearDiophantine
        (int A, int B, int C, int *X, int *Y)

    int g=gcd (A,B)

    if (C%g != 0) return false

int a = A/g   b = B/g   c = C/g

    ext_gcd (a, b, x, y)

    if (g < 0)
    { a = a*-1      b = b*-1    c = c*-1 }

*X = x * c

*y = y * c

return true

}
```

Sieve of Eratosthenes

```
int max_test=7;
bool isPrime[max];
vector<int> prime;
```

Pseducode

```
void sieve(){
    is
```

```
Void sieve ()
{
  int n;
  vector<bool> isPrime (n+1, true);

  isPrime[0] = isPrime[1] = false;

  for (int i = 2 ; i*i <= n ; i++ ) {

      if (isPrime[i] == true)
        { for (int j = i*i ; j<=n ; j += i)
            { isPrime[j] = false }
        }
}
```

Initial all number true or prime

n = 16

→ 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

i = 2 Prime and all its gunda not prime

→ (2) 3 4 5 6 7 8 9 10 11 12 13 14 15 16

i = 3 Prime

→ 2 (3) 4 5 6 7 8 9 10 11 12 13 14 15 16

i = 4

→ isprime[4] = 4 = not prime

= 2nd loop not work

i = 5

→ 5 × 5 = 25 > n loops break

(2) (3) 4 (5) 6 (7) 8 9 10 (11) 12 (13) 14 15 16

Primes →

Bit sieve

```
int N = 100;
int status[100/32];
```

→ inital all value 0

→ status arrr is integer type
Here it has 32bit in
1 index

→ all these 32 bit indicates
32 different numbers
whether they prime or not

→ status[0] = 0 — 31 number

→ a number index = i/32;

→ number status of prime
is at status[i/32] in
i%32 bit

```
bool check(int index,
           int position)
{
   return
   (bool)(index & (1<<position));
}
```

```
bool set(int index, int position)
{ return index = index | (1<< position)
}
```

```
void sieve
{ * we only hunt for odd number
    even num by default not prime ]

for(int i=3 ; i<= sqrt(N) ; i+=2 )
```

```
{
if (check(status[i/32], i%32) == 0 )
for (int j = i * i ; j <= N ; j += (2i) )
  {
    status[j/32] = set (status[j/32) , j%32)
  }

prime.pushback(2);
for (int i=3 ; i <= N ; i += 2)
{
  if (check(status[i/32], i%32) == 0 )
      prime.push back[i];
}
```

How works

Suppose **i = 3**

now i/32 = 3/32 = 0 status[0] index 0

i%32 = 3%32 = 3 pos : 3 → 3rd bit

* check (status [3/32], 3%32)
  0 3

→ index value = 0
∴ in binary index value = 0 (binary)

→ 1 << position = 1 * 2^pos
= 1 * 2^3
= 8
= 1000 (in binary)

now index & (1<<position
→ 0 & 1000
→ 0

and
1000
& 0000
0000

∴ ans == 0 ; 3 is prime number

now mark all its multpies

J = i x i → J = 3 x 3 = 9

J = 9

status [9 / 32] = 0 (something) ~~index~~

9 % 32 = 9 → position

status [9/32] =

set (status [9/32] , 9 % 32)

0 9

→ Index = 0

= 0 (binary)

→ 1 << position = 1 * 2^9

= 512

= 100 000 000 (binary)

→ index | (1 << position) → 0 | 1000 00000

→ 1000 00 000

∴ index = 512

∴ status [9/32] = 512 (setted)

i = 71

71/32 = 2 → status[2] → index 2

71%32 = 7 → pos → 7

check (2 , 7)

∴ index 2 = 10 (binary)

∴ 1 << position = 2^7 = 128

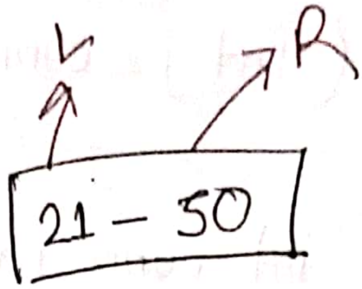
= 10000000 (binary)

∴ Index & 1 << position = 10 & 10000000

= 0

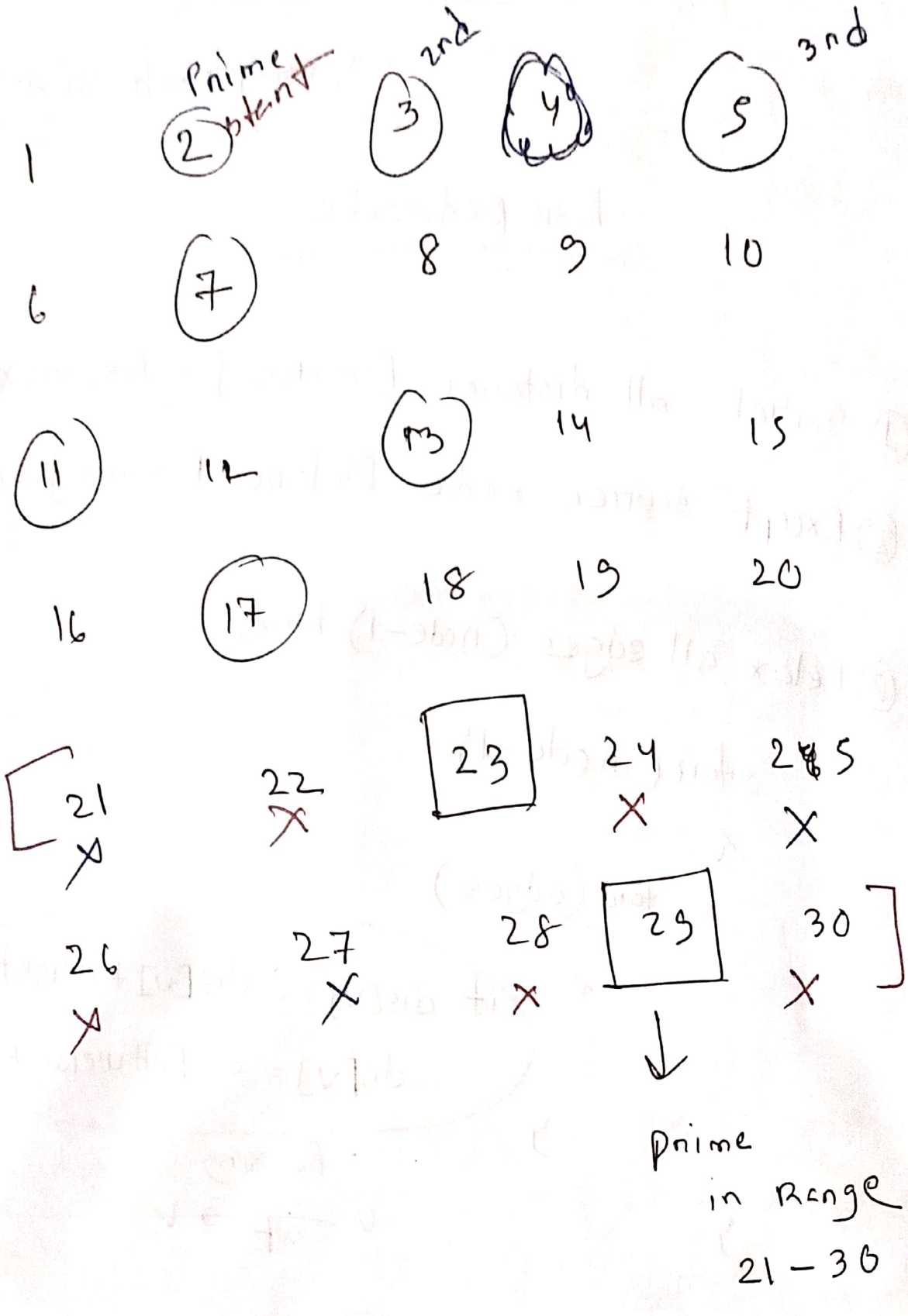
∴ 71 is prime

Segmented Sieve




```
vector<int> prime;
sieve();

void segSieve( int L , int R )
{
    vector<bool> isprime(R-L+1, true)   // Intial all true
    if (L==1) isprime[0] = false   → extreme case

    for (int i=0 ; prime[i]* prime[i] <= R ; i++)
    {
    ② int currentPrime = prime[i]
       int currbase = (L/currprime) * curr prime
           20           21    2          2
       if (currbase < L)  currbase += curr prime
           20        21            22
```

```
for (int j = currbase ; j <= n ; j += currprime)
{
    int currindex = j - L      * index

    isprime [currIndex] = false
}

if (currbase == currprime)
        isprime [currbase - L] = true
}

for (int i = 0 ; i < (n-l+1) ; i++)
{
    if(isprime [i])
        cout << i+L << endl.
}
```


Prime
2nd
3rd
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
Prime
in Range
21 - 30

Bellman Fond

→ complexity $O(VE)$

basic pseudocode

① Initial all distance [vertice] = INT_MAX (∞)

② Except source node Distance [source] = 0

③ Relax all edges (node-1) times

```
for( node-1)
{
    for (edges)
    {
    }
}
```

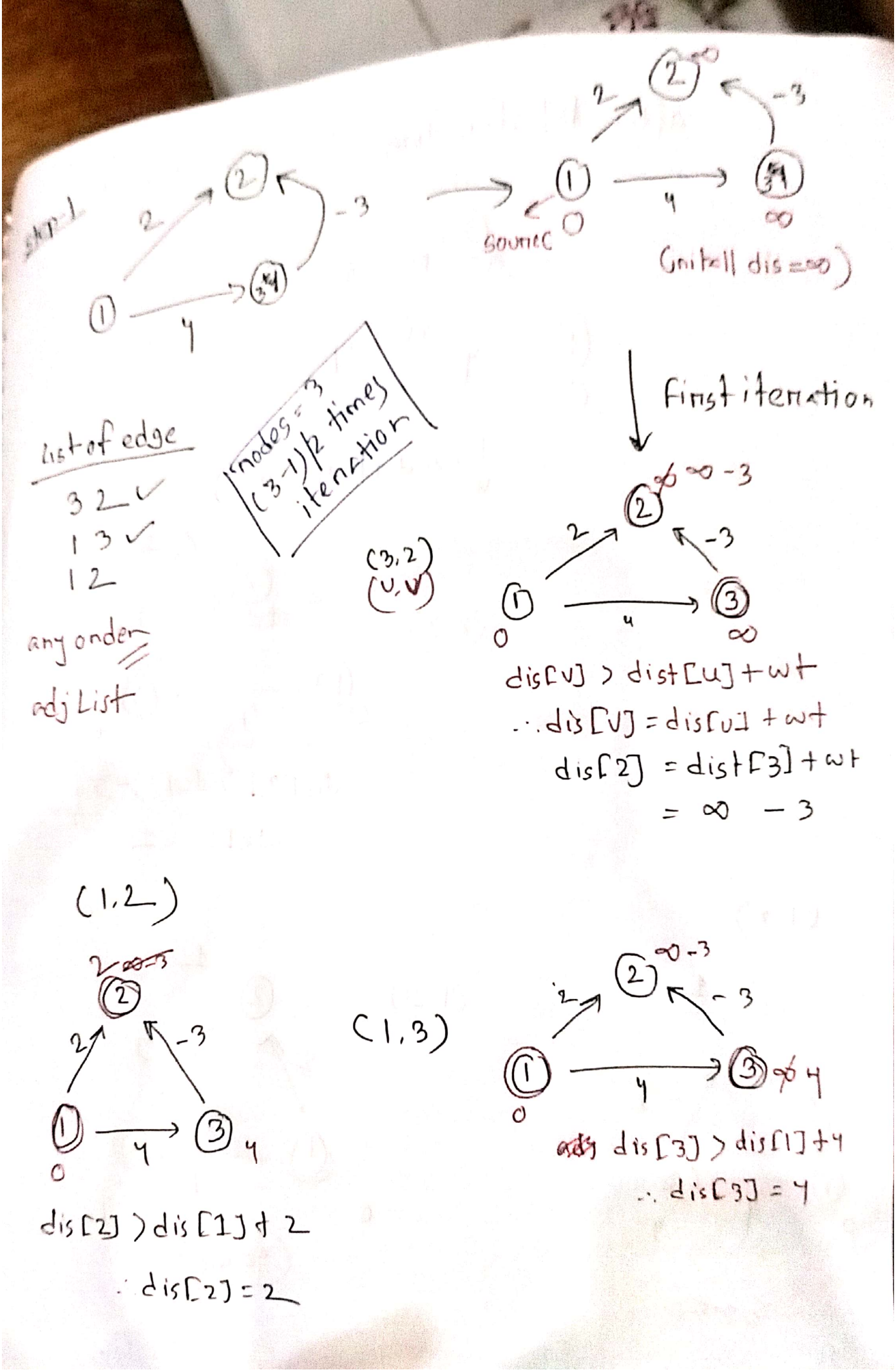
if dis[v] > dis[u] + weight

∴ dis[v] = dis[u] + weight

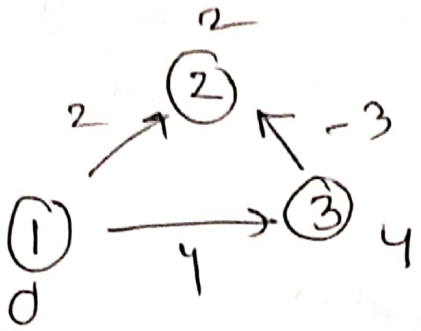
Relaxing

u —wt→ v

④ Negative weight cycle Detection

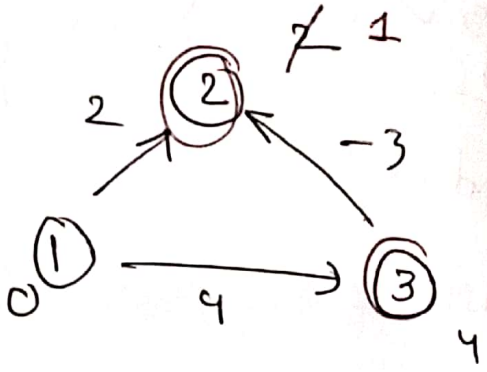

skipped
2
-3
4
source
0
∞
Initall dis = ∞)
first iteration
nodes = 3
(3-1) 2 times
iteration
(3,2)
(u,v)
list of edge
3 2
1 3
1 2
any order
adj List
dis[v] > dist[u] + wt
∴ dis[v] = dis[u] + wt
dis[2] = dist[3] + wt
= ∞ - 3
(1,2)
(1,3)
dis[2] > dis[1] + 2
∴ dis[2] = 2
dis[3] > dis[1] + 4
∴ dis[3] = 4

After 1st iteration



↓ (2nd iteration)

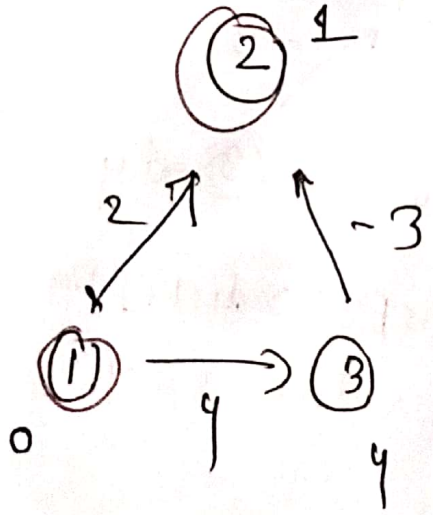
(2,3)



dis[2] > dis[3]-3

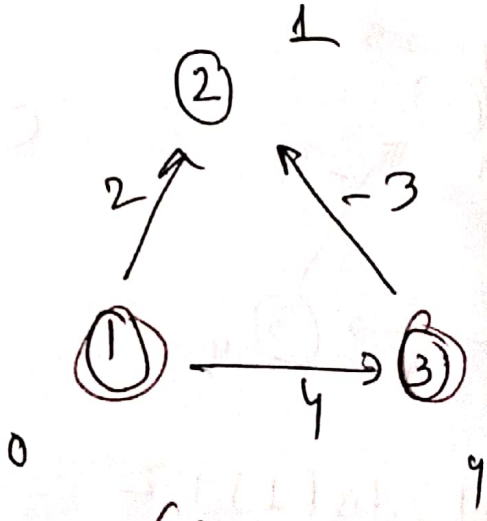
dis[2] = 1

(1,2)



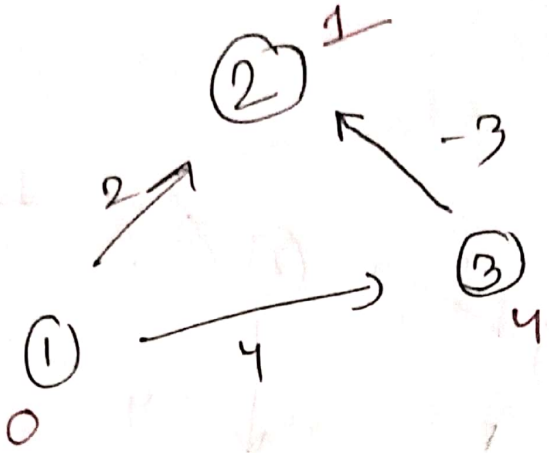
Same

(1,3)

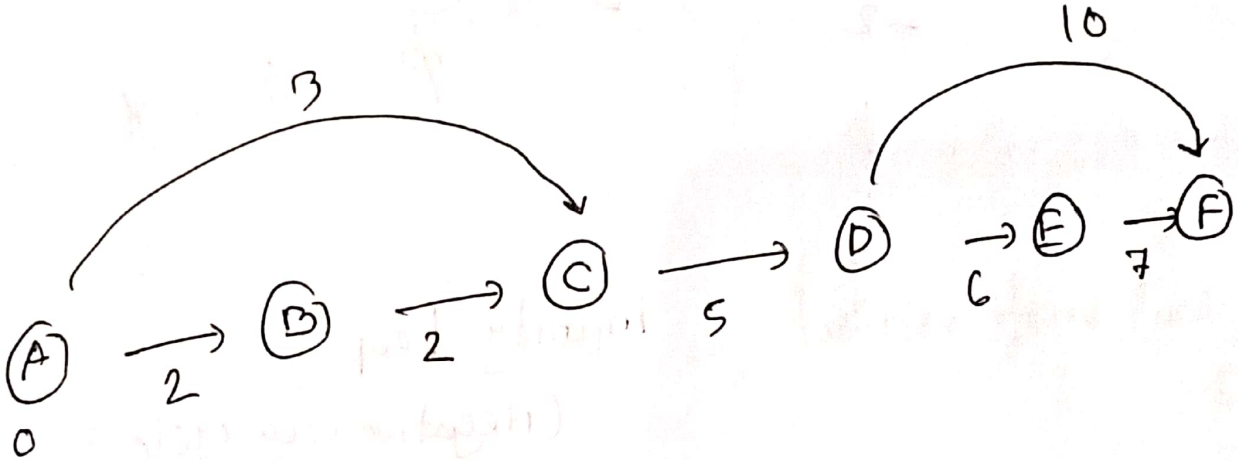


(same)

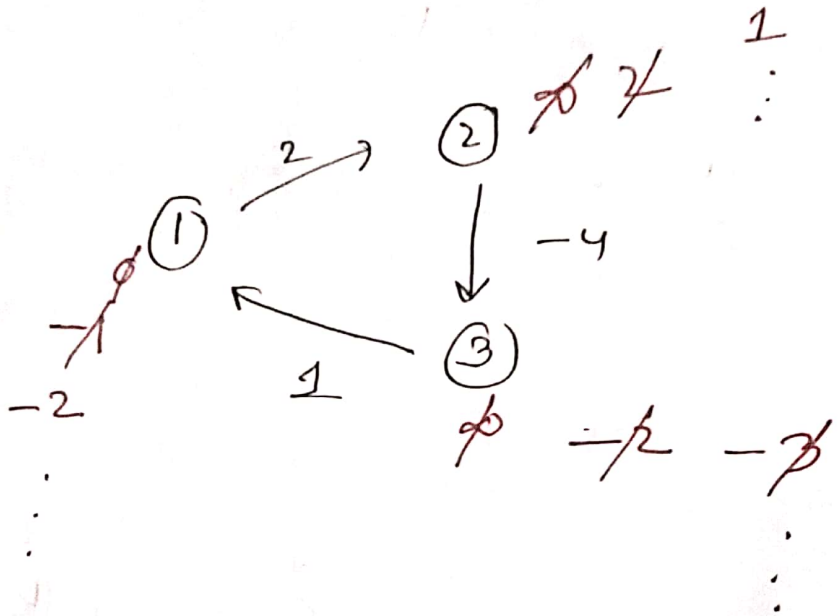
After all iteration graph is now



1	2	3
0	1	4



Negative wt cycle



infinity Loop
(Negative weg cycle)

```
int main( )
{   node, edge ;
    vector < vector < int > > adj ;
    for( int i = 0 ; i < edge ; i++ )
    {       node1, node2, weight
        adj. pushback ( { node1, node2, weight} )
    }

vector <int> dis = bellman ford (node, 1,
                                 adj )
```

```
Vector<int> bellmanford (int node, int sourec,
                         vector<vector<int>> adj)
{
    # inital all ∞ distance
    Vector<int> dis (node+1, Int_max)
    vector<int> parent (node, 0)
    dis[sourec] = 0;
#Relaxing nodes/edges
 for( i = 0 to node - 1 )
    {
        for (auto edge : adj)
        {
            node1 = edge[0]    wt = edge[2]
            node2 = edge[1]

            if (distance [node1] + wt < dis [node2]

            dis [node 2 = dis [node 1] + wt

            parent [node2] = node 1
        }
    }

}
```

After doing 1 more relaxation if any distance get change that means graph → negative wt cycle

```
for(auto edge : adj)
{
    node1, node2, a edge wt

    if(dis[node] > dis[node1] + wt)
    {
        cout << "Negative wt cycle";
        exit(0);
    }
}
```

parent printing / path

∴ int presentnode = node

∴ vector<int> path

while (parent[presentnode] != 0)

{

path.push_back(presentnode)

presentnode = parent[presentnode]

}

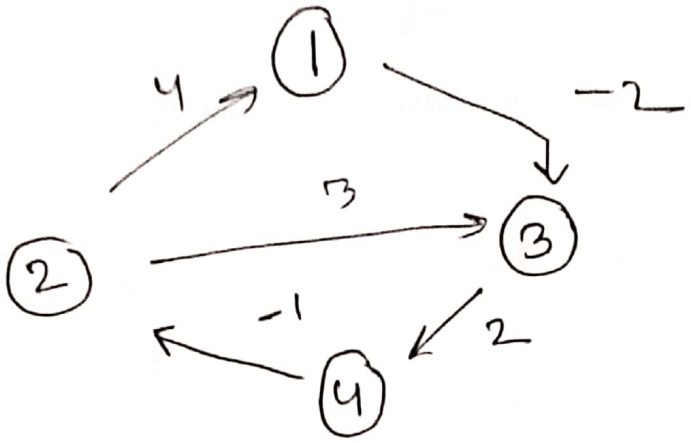
reverse (path.begin(), path.end())

for (auto node : path)

cout<<node

Floyd warshall

→ $O(N^3)$ time



dist

	1	2	3	4
1	0	∞ (no edge)	-2	∞
2	4	0	3	∞
3	∞	∞	0	2
4	∞	-1	∞	0

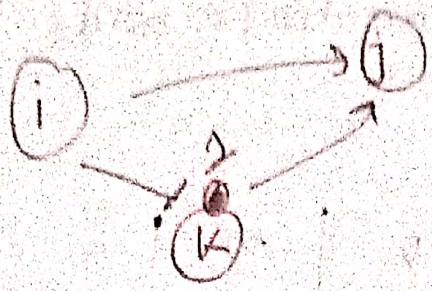
0
1 -2→ 3
2 4→ 1
2 3→ 3
3 2→ 4
4 -1→ 2

adj List

dis_mat

1as intermediate vertex

2nd



update path

	1	2	3	4
1	0	∞	-2	∞
2	4	0	2	∞
3	∞	∞	0	∞
4	∞	-1	∞	0

1

as 1 1st row and col not change

$$d[i][j] = \min(d[i][j],\ d[i][u]+d[k][j])$$

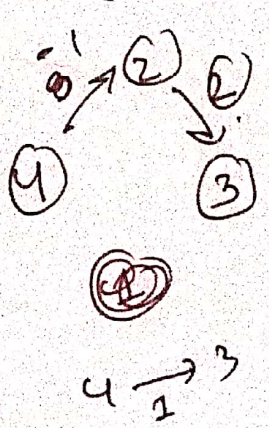
3rd

2 as intermediate

	1	2	3	4
1	0	∞	-2	∞
2	4	0	2	∞
3	∞	∞	0	2
4	∞	-1	∞	0

2→3

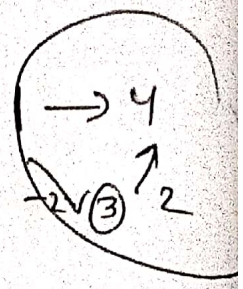
check dis from table not graph



4 → 1 → 3

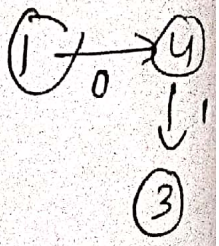
④ 3rd as intermediate node

	1	2	3	4
1	0	~~∞~~	-2	~~∞~~ 0
2	4	0	2 P	~~∞~~ 2+2=4
3	∞	∞	0	2 P
4	~~∞~~ 3	-1	1	0



⑤ 4 as intermediate node

	1	2	3	4
1	0	~~∞~~ 0+-1=-1	-2	0
2	4	0	2	4
3	~~∞~~ 2+3=5	~~∞~~ 2	0	2
4	3	-1 P	1	0



	1	2	3	4
1	0	-1	-2	0
2	4	0	2	4
3	5	1	0	2
4	3	-1	1	0

→ 1st check graph then table if possible

① → ④ → ③ ✗ not work

↑ ② ← ⑤ ↙

overlap doesn't work