Fibnocci Series

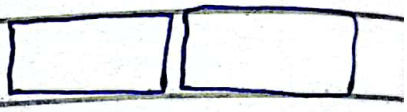
Binomial Coefficient

Matrix Chain Multiplication

Additive

LCS

Divide & Conquer



Non overlapping sub problems

Dynamic Programming

Fibnocci Series (1 Dimensional Problem)

Recursive algorithm

Example

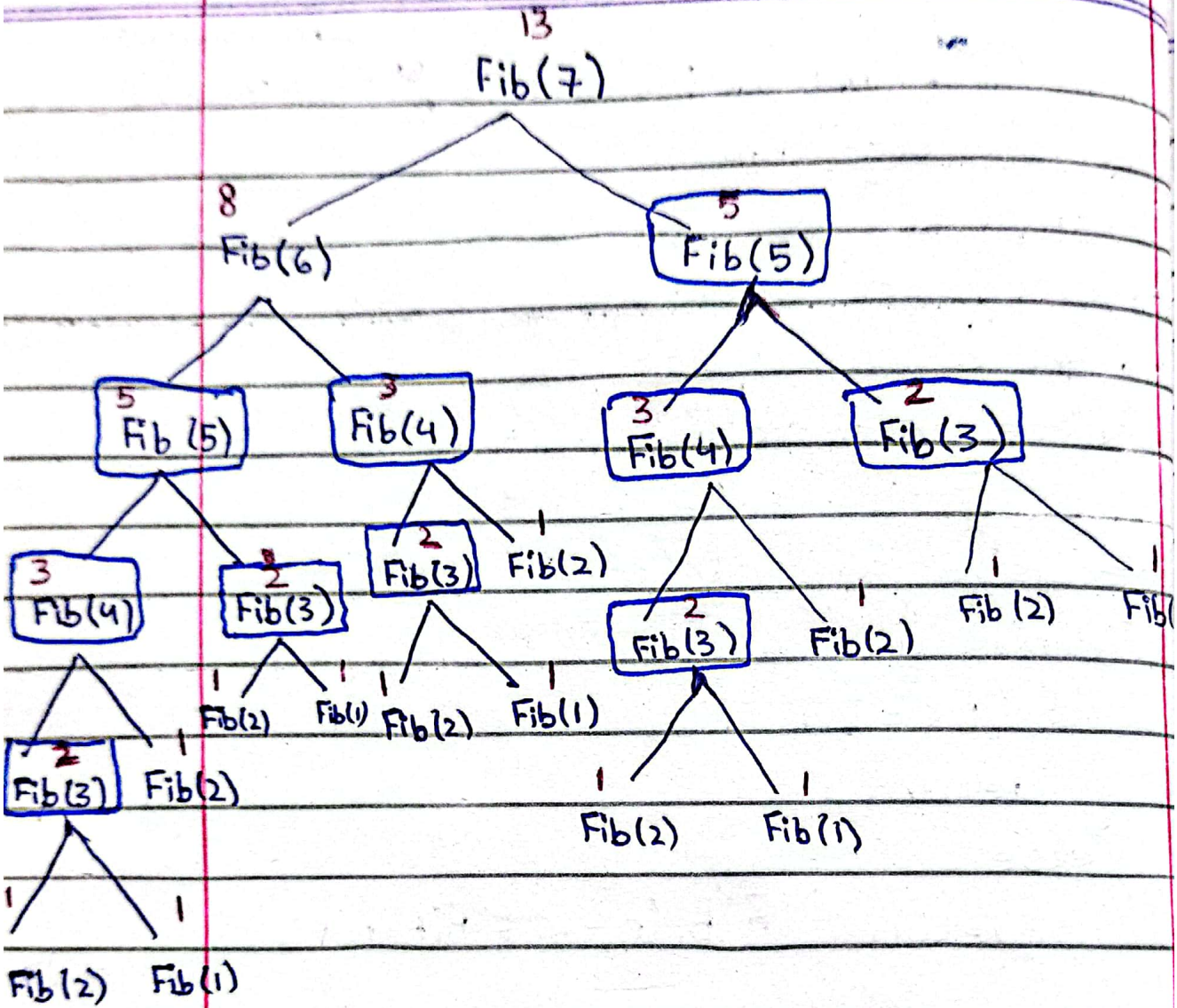
check ↑ Memorised and write algo

Then check pattern

Write iterative algo

Recursive algorithm :-

```
Fib(n)
  if (n<=2)
    return 1
  else
    return  Fib(n-1) + Fib(n-2)
```



Fib(3)
Fib(4)
Fib(5)
} is repeated

Time Complexity

$T(n) = T(n-1) + T(n-2) + c$

$T(n) = T(n-1) + T(n-1) + c$

$T(n) = 2T(n-1) + c$

$T(1) = c$

$T(n) = 2T(n-1) + c \rightarrow (1)$

$T(n-1) = 2T(n-2) + c \rightarrow (2)$

$T(n) = 2[2T(n-2) + c] + c$

$T(n) = 4T(n-2) + 2c + c \rightarrow (3)$

$T(n-2) = 2T(n-3) + 3c$

$T(n) = 4(2T(n-3) + c) + 2c + c$

$T(n) = 8T(n-3) + 4c + 2c + c$

$T(n) = 2^3 T(n-3) + 2^2 c + 2^1 c + 2^0 c$

$= 2^k T(n-k) + \underbrace{2^{k-1} c + \cdots\cdots 2^0 c}_{k+1}$

$n - k = 1$

$n = k$

$= 2^n c$

$= O(2^n)$

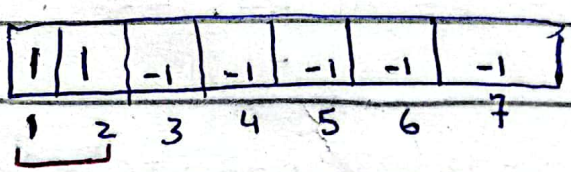
$2^0 + 2^1 + \ldots + 2^n + 2^{n+1}$

2^n

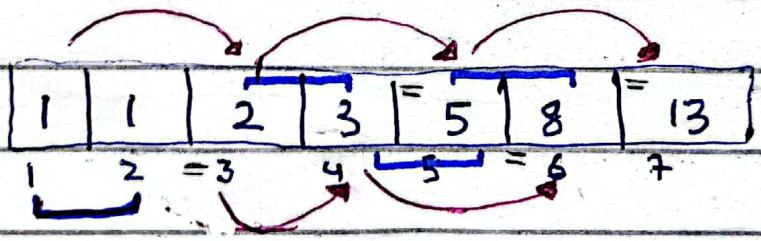
Memorized

```
Fib_mem(n)
{
    if (A[n] != -1)
        return A[n]
    else
        A[n] = A[n-1] + A[n-2]
        return A[n]
}
```

Check pattern



First 2 fixed with value 1 and 1



1+1 = 2

2+3 = 5

3+5 = 8

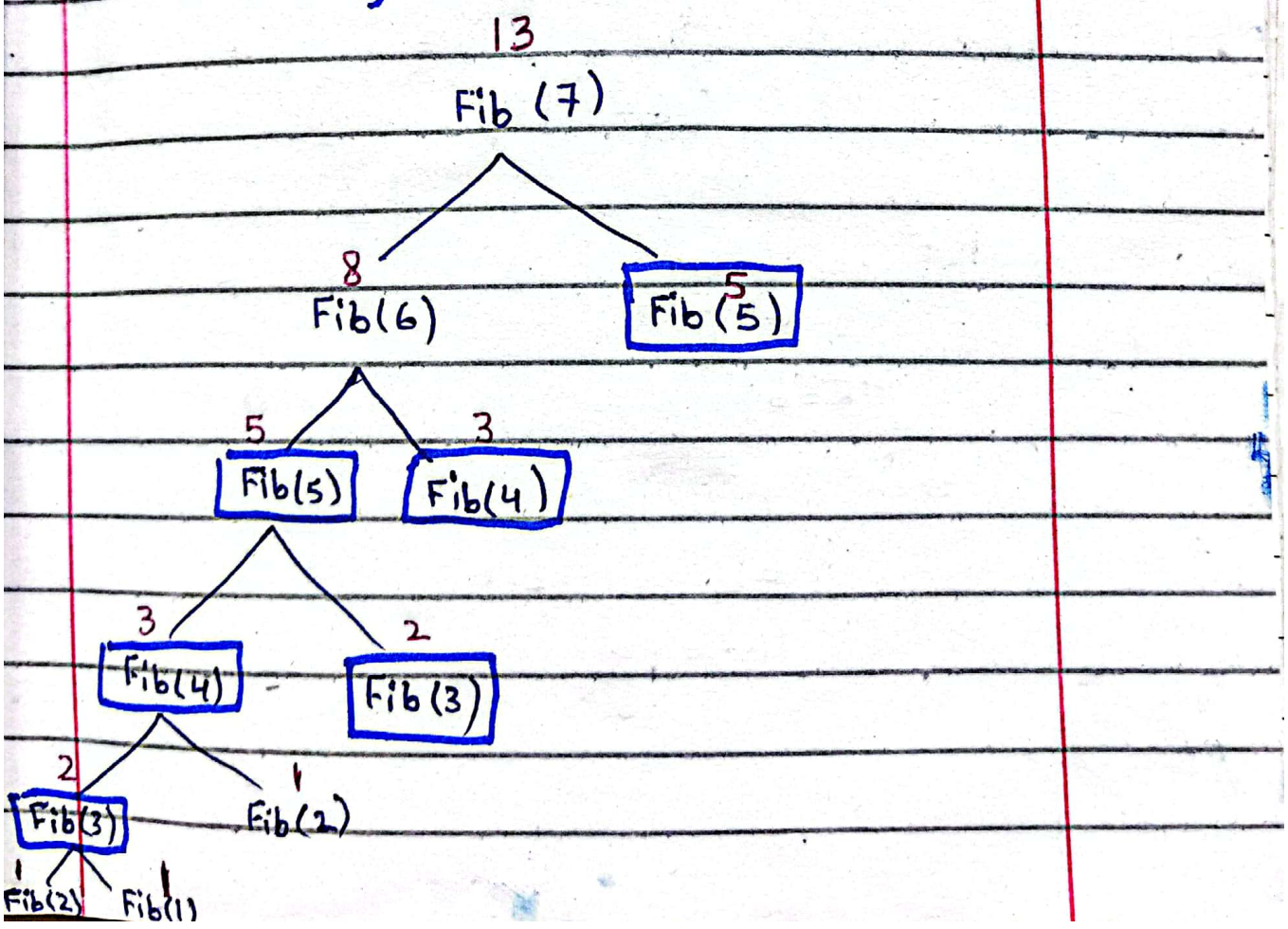
5+8 = 13

Iterative Algo

```
Fib - iterative (n)
    F1 = 1
    F2 = 1
    for (i = 3 to n)
        Fn = F1 + F2 ;
        F1 = F2 ;
        F2 = Fn ;
```

O(n)

Diagram



8 May 2023

(2D Problem)

Binomial Coeffici

$$\binom{n}{k} = \frac{n!}{(n-k)!\,k!}$$

Put

$n = 6$

$k = 0$

then

$$\frac{6!}{(6-0)!\,0!} = 1$$

Put

$n = 6$

$k = 6$

then

$$\frac{6!}{(6-6)!\,6!} = 1$$

Recursive definition:-

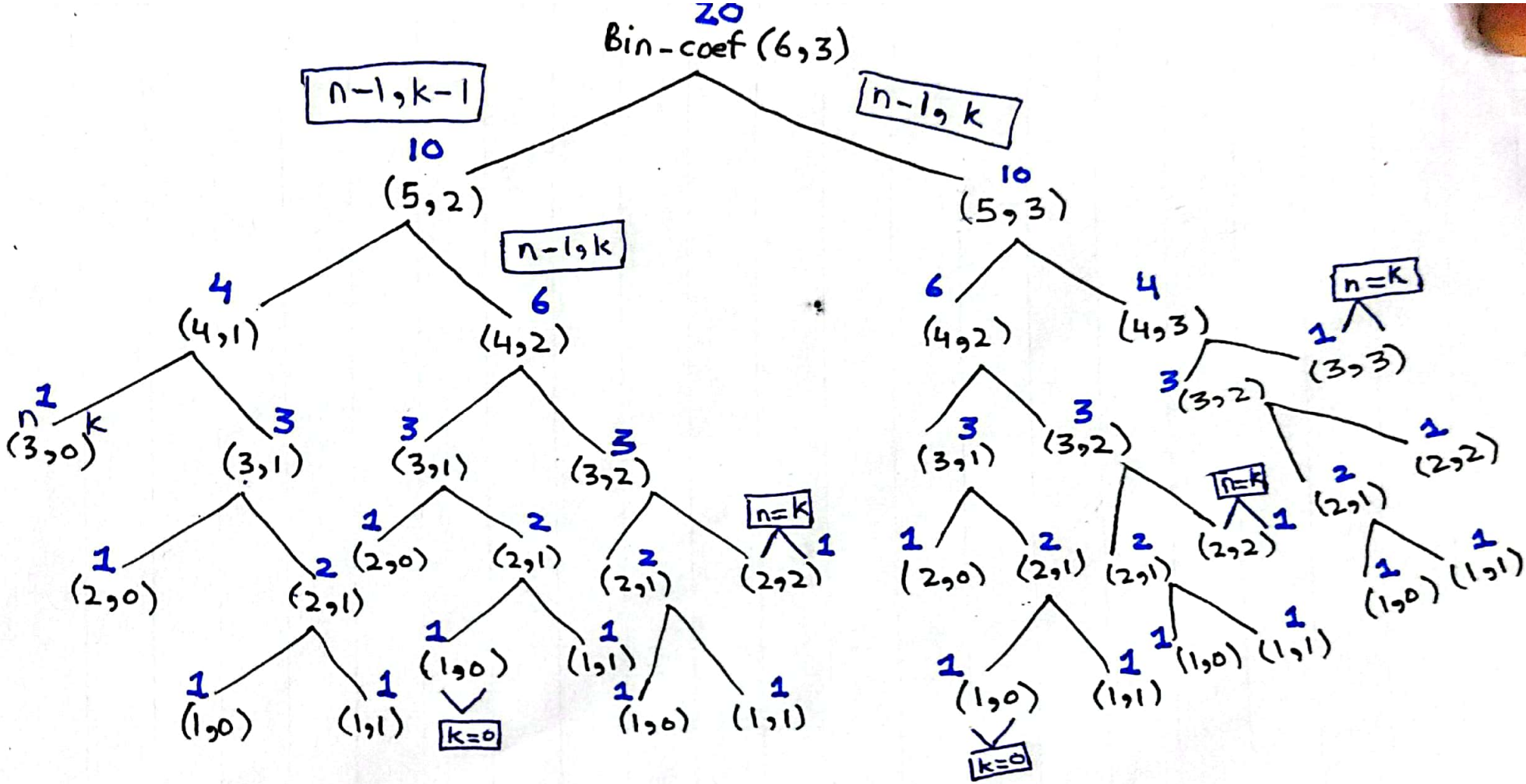
$$R.C\binom{n}{k} = \begin{cases} 1 & n=k \;||\; k=0 \\ & \text{otherwise} \\ R.C(n-1,k-1) + R.C(n-1,k) \end{cases}$$

Recursive algo:-

```
Bin_coef (n,k)
{
    if (n==k || k==0)
        return 1
    else
        Bin_coef (n-1,k-1) + Bin_coef (n-1,k);
}
```

i.e

$$\binom{6}{3} = \frac{6!}{(6-3)!\,3!} = 20$$

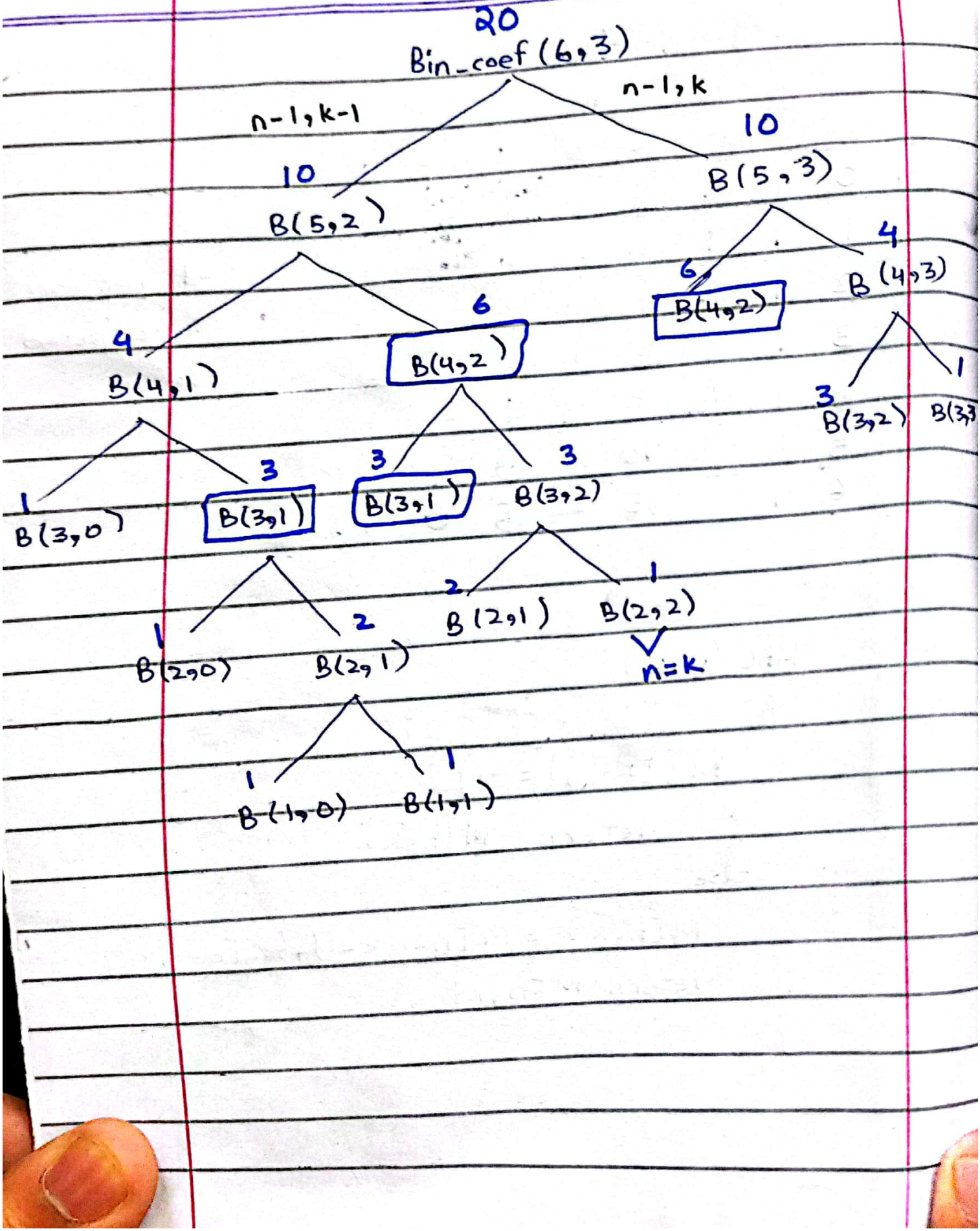


Memorized

n \ k	0	1	2	3	4	5	6
0	1	-1	-1	-1	i-1,j-1	i-1,j	-1
1	1	1	-1	-1	-1	i,j	-1
2	1	2	1	-1	-1	-1	-1
3	1	3	3	1	-1	-1	-1
4	1	4	6	4	1	-1	-1
5	1	5	10	10	5	1	-1
6	1	6	15	20	15	6	1

-1 represent uncomputed value

```
MBC (n,k)
{
    M([n,k] != -1)
        return M(n,k)
    else
        M[n,k] = M[n-1,k-1] + M[n-1,k]
        return M[n,k]
}
```



Iterative algorithm

```
Iterative (n,k)
{
    for (i=0 to n)
    for (j=0 to k)
        if (i=j || k=0)
                A[i,j] = 1
        else
                A[i,j] = -1
for (i=0 to n)
for (j=0 to k)
        A[i,j] = A[i-1,j-1]+A[i-1,j];
}
```

MATRIX MULTIPLICATI

A x B

(p x q) (q x r)

Number of multiplication = pqr

Order of Resultant = pr

every row of A

p ← for (i = 0 to p)

every column of B

r ← for (j = 0 to r)

C[i,j] = 0

q ← for (k = 0 to q)

every column of / every row of

C[i,j] += A[i,k] + B[k,j]

Complexity

$O(pqr)$

A	B
2 x 3	3 x 6
(p x q)	(q x r)

Number of multiplication = pqr

= 2 x 3 x 6

= 36

Order of resultant matrix = 2 x

①	A	B	C
	2x3	3x5	5x10

2 way of multiplication

→ (A (B x C))

→ ((AB) C)

$(A\ (B \times C))$

$3 \times 5 \times 10 = 150$

$2 \times 3 \times 10 = 60$

210

$((AB)C)$

A B C

2×3 3×5 5×10

$2 \times 3 \times 5 = 30$

$2 \times 5 \times 10 = 100$

130

130 is best

LCS

Longest common Subsequence

$w = abcd$

↓

ab by cd , da

$2^4 = 16$

Subsequence always computed between 2 words

$S_1 = a\ b\ c\ d\ e\ f\ g$

$S_2\ \ a\ \ c\ \ g$

acg ac , ag

longest subsequence never intersect

Example

X = a b a a b a

Y = b a b b a b

Matrix chain multiplication

1
B(3.

Same

Diagonal ki value mein 1 add

different mein maximum left or upar wala

maximum check from here

X = i \ Y = j		a	b	a	b	b	a	b
	null	0	0	0	0	0	0	0
a		0	←0	↖1	←1	←1	←1	←1
b		0	↑1	↖1	↖2	←2	←2	↖2
a		0	↑1	↖2	←2	↑2	↖3	←3
a		0	↑1	↖2	←2	←2	↖3	←3
b		0	↑1	↖2	↖3	↑3	↑3	↖4
a		0	↑1	↖2	↑3	↑3	↑3	↑4

LCS must be 4 character

Algo:

if $x_i = y_j$

$X[i] == Y[j]$

OR

$A[i,j] = 1 + A[i-1, j-1]$

else

$A[i,j] = \max(A[i-1,j], A[i,j-1])$;

Algorithm

if (A[i] == B[j])

LCS [i,j] = 1 + LCS [i-1, j-1]

else

LCS [i,j] = LCS [i-1, j] , LCS [i, j-1]

Diagonal value considered ↖

X = cellular OR

Y = Excellent

ASCII code

X = 1 1 1 0 0 1 1 0

Y = 0 1 1 0 0 0 1 1

X_i \ Y_j	λ	E	x	c	e	l	l	e	n	t
λ	0	0	0	0	0	0	0	0	0	0
c	0	←0	←0	↖1 (0+1)	←1	←1	←1	←1	←1	←1
e	0	↑0	↑0	↑1	↖2 (1+1=2)	←2	←2	↖2 (1+1)	←2	←2
e	0	↑0	↑0	↑1	↑2	↖3 (1+2)	↖3 (2+1)	←3	←3	←3
l	0	↑0	↑0	↑1	↑2	↖3 (2+1)	↖4 (3+1)	←4	←4	←4
u	0	↑0	↑0	↑1	↑2	↑3	↑4	←4	←4	←4
l	0	↑0	↑0	↑1	↑2	↑3	↑4	←4	←4	←4
a	0	↑0	↑0	↑1	↑2	↑3	↑4	←4	←4	←4
r	0	↑0	↑0	↑1	↑2	↑3	↑4	←4	←4	←4

4 ← LCS

cell

Algorithm in LCS

ASCII code

X = 1110110

Y = 0110011

x_i \ y_j	λ	0	1	1	0	0	1	1
λ	0	0	0	0	0	0	0	0
1	0	← 0	↖ 0+1 1	↖ 0+1 1	← 1	← 1	↖ 0+1 1	↖ 0+1 1
1	0	← 0	↖ 0+1 1	↖ 1+1 2	← 2	← 2	↖ 1+1 2	↖ 1+1 2
1	0	← 0	↖ 0+1 1	↖ 1+1 2	← 2	← 2	↖ 2+1 3	↖ 2+1 3
0	0	↑ 0+1 1	← 1	↑ 2	↖ 2+1 3	↖ 2+1 3	← 3	← 3
1	0	↑ 1	↖ 1+1 2	↖ 1+1 2	↑ 3	← 3	↖ 3+1 4	↖ 3+1 4
1	0	↑ 1	↖ 1+1 2	↖ 2+1 3	← 3	← 3	↖ 3+1 4	↖ 4+1 5
0	0	↖ 1+0 1	↑ 2	↑ 3	↖ 3+1 4	↖ 3+1 4	← 4	↑ 5

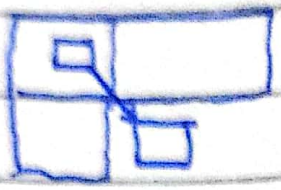
5 ← length of binary code

11011

Algorithm

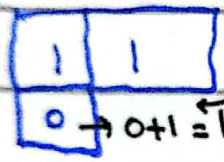
Edit distance

If r = c



diagonal will be copy

r ≠ c



take minimum from these 3 and add one

Example:

$W_1 = A\ B\ F\ G\ L\ K$

$W_2 = X\ B\ O\ G\ R\ L$

W_1 \ W_2	λ	X	B	O	G	R	L
λ	0	0	0	0	0	0	0
A	0	0+1 ← 1	0+1 ↑ 1	0+1 ↑ 1	0+1 ↑ 1	0+1 ↑ 1	0+1 ↑ 1
B	0	0+1 ← 1	↖ 1	1+1 ← 2	1+1 ↑ 2	1+1 ↑ 2	1+1 ↑ 2
F	0	0+1 ← 1	1+1 ← 2	1+1 ↖ 2	2+1 ← 3	2+1 ↑ 3	2+1 ↑ 3
G	0	0+1 ← 1	1+1 ← 2	2+1 ↑ 3	↖ 2	2+1 ← 3	3+1 ↑ 4
L	0	0+1 ← 1	1+1 ← 2	2+1 ← 3	2+1 ↑ 3	2+1 ↑ 3	↖ 3
K	0	0+1 ← 1	1+1 ← 2	2+1 ↖ 3	3+1 ↑ 4	3+1 ↑ 4	3+1 ↑ 4

w_1 = CAMPUS

w_2 = CAMERA

w_1 \ w_2	λ	C	A	M	E	R	A
λ	0	0	0	0	0	0	0
C	0	0	1	1	1	1	1
A	0	1	0	1	2	2	1
M	0	1	1	0	1	2	2
P	0	1	2	1	1	2	3
U	0	1	2	2	2	2	3
S	0	1	2	3	3	3	3

3 basic operations

Convert

ALGORITHM

to

ALTRUISTIC

	λ	A	L	T	R	U	I	S	T	I	C
λ	0	0	0	0	0	0	0	0	0	0	0
A	0	0	1	1	1	1	1	1	1	1	1
L	0	1	0	1	2	2	2	2	2	2	2
G	0	1	1	1	2	3	3	3	3	3	3
O	0	1	2	2	2	3	4	4	4	4	4
R	0	1	2	3	2	3	4	5	5	5	5
I	0	1	2	3	3	3	3	4	5	5	6
T	0	1	2	2	3	4	4	4	4	5	6
H	0	1	2	3	3	4	5	5	5	5	6
M	0	1	2	3	4	4	5	6	6	6	6

Brute Force Algorithm

Document : Text

Pattern : (Word search)

m → text length

Text → length bigger → n

Pattern → smaller → m

n * m

(n - m + 1) m

(text ky har character)

j ← 0 to n - m do

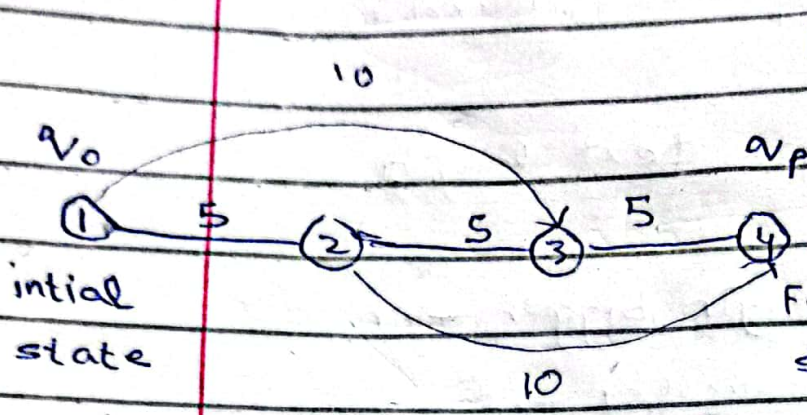
last mein jab pattern sy 1 kam reah jay gah tu possible

For pattern

for j ← 1 to m do

$$O\left((n-m+1)*m\right)$$

Finite State Automation



length = 5 / 10

Pepsi = 15

Transition: moving from 1 state trailer to another

Q : finite state

Σ represent language

δ (Sigma) transition function

Pattern: Prefix

[a b c a b c a] +1 = 8

Suffix / Postfix

Σ (a, b, c)

[a] Prefix

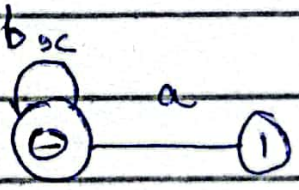
[a] sufix [b] sufix

	a	b	c
0	1	0	0
1	1	2	0
2	1	0	3
3	4	0	0
4	1	5	0
5	1	0	6
6	7	0	0
7	1	5	0

δ ← a = a, b = b, c = c

[a b
a a] = 1

[ab] Starting 2



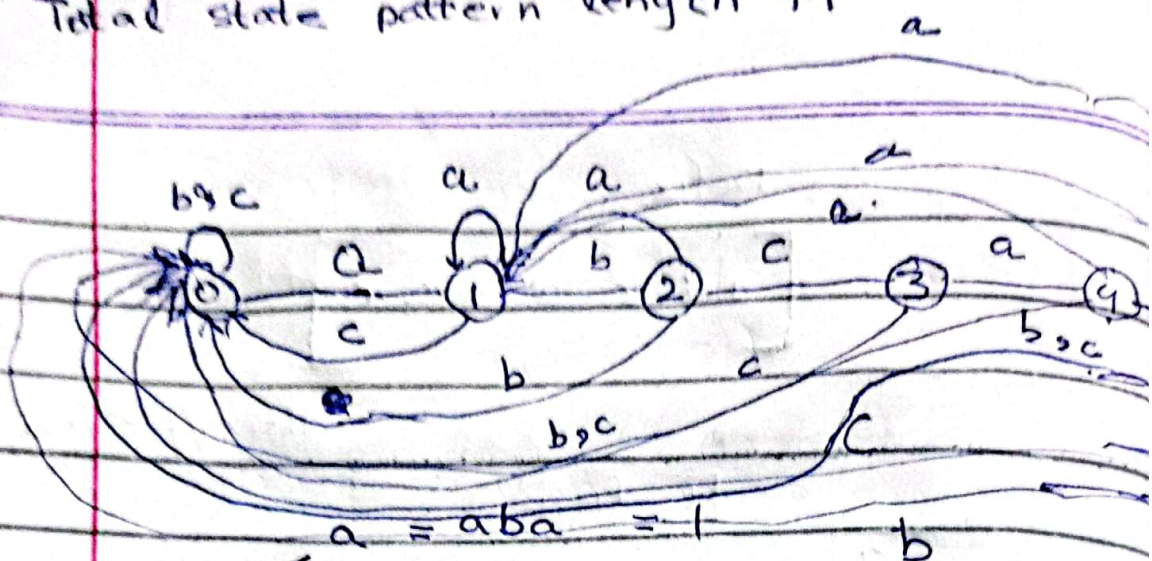
ending

[aa] = 1

a → b: ab = 2

a → c: ac = 0

Total state pattern length +1



a = aba = 1

ab ← b = abb = 0

c = abc = 3

aba ✗

abc

ba ✗

ab

a

1

a

abb ✗

abc

bb ✗

ab

a

c

a b c ← a = abca 4

b = abcb 0

c = abcc 0

Ending

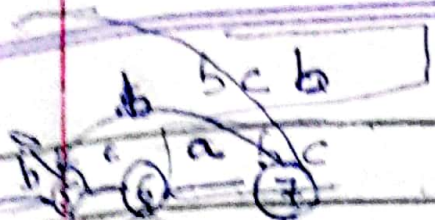
start

abca

abca

abcb ✗

abca



X

abca
- a = abcaa = 11
- b = abcab = 5
- c = abcac = 0

165

abcab
- a = abcaba = 1
- b = abcabb = 0
- c = abcabc = 6

P_V

a

abcabca = 1

abcabcab = 5

abcabcac = 0

a a b a c b abc abca bca

Transition Table

$O((m+1) \cdot |\Sigma| \cdot (m+1) \cdot m)$

$O(m^3 \cdot |\Sigma|)$

String match complexity

$\Theta(n)$