[(1, 18) – 1
(18, 23) – 2
(15, 29) – 3
(4, 15) – 2
(2, 11) – 3
(5, 13)] – 4

o/p: 4

(1, 18)
(18, 23)

o/p: 2
(18, s) (18, e)
(1, 18) (18, 23)
(1, s) (18, e)

(1, 18) (2, 11) (4, 15) (5, 13) (15, 29) (18, 23)
0 1 2 3 4 5

Meeting-room: 1 18 | 2 11 (15, 29) | 4 15 | 5 13

⇒ TC: O(n^2)
AS: O(n)

s s s s e e s e s e e e
1 2 4 5 11 13 15 15 18 18 23 29
1 2 3 $\boxed{4}$ 3 2 3 2 3 2 1 0

TC: O(n log n)
AS: O(n)

s s e s e e
0 5 10 15 20 30
1 $\boxed{2}$ 1 $\boxed{2}$ 1 0

int → 18

's' 'e' (18, e) (18, s)

<

s c

Minimum Platforms

---

id	deadline	profit
1	4	20
2	1	10
~~3~~	1	40
→ 4	1	30

0-1 | 1-2
3 | 1

40 + 20 = 60

40

1	2	100
2	1	19
3	2	27
4	1	25
5	1	15

0-1 | 1-2
~~4~~ 3 | 1
~~25~~ 27 | 100 = ~~125~~ 127.

~~1~~	4	20
→ 2	1	10
→ ~~3~~	10	40
~~4~~	~~8~~ 1	30

time: 1

40

slots 0 1 2 3 4 5 6 7 8 9 10
[4 | | | | | | | | | 3 | 4 - . . .]
0-1 1-2 2-3 9-10

→ 90.

i → (i, i+1)

TC: O(n log n + n * min(n, maxx))

AS: O(maxx)

n → 1000
max → 10^7

~~(1, 2, 100)~~
~~(2, 1, 19)~~
~~(3, 2, 27)~~
~~(4, 1, 25)~~
~~(5, 1, 15)~~

Jobs Done = ~~2~~ 3
profit = ~~25~~ ~~125~~ 127

profits: (~~25~~, 100, 27)
Min-Heap

TC: O(n log n + n log n) = O(n log n)
AS: O(n)

MM TC: O(n)
AS: O(n) ←

[3, 1, 3, 2, 3, 4, 3, 3]

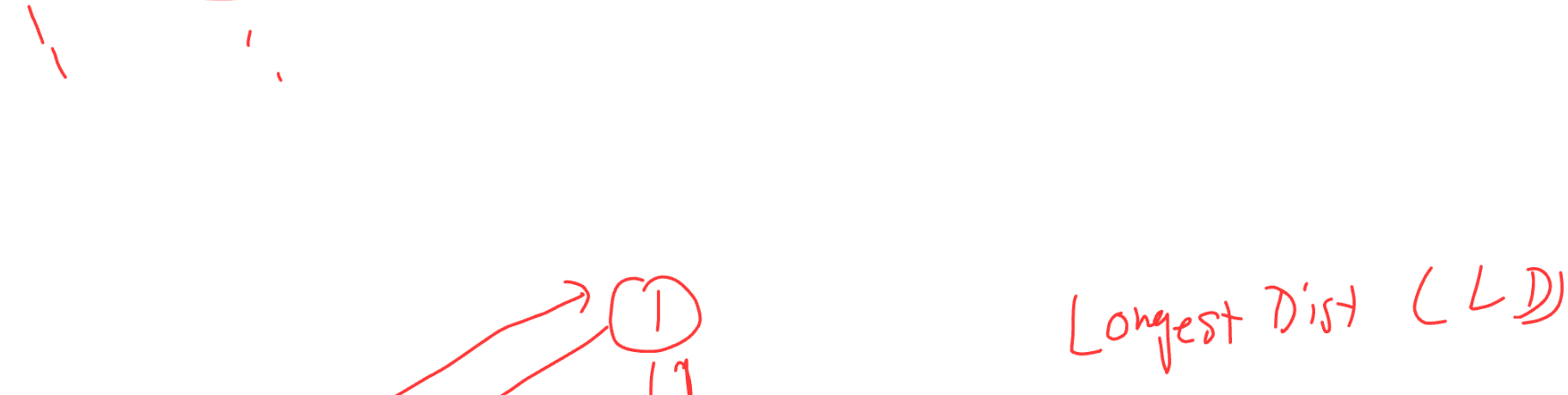
a = 3 cnt: 1 2 1 2 1 2 3 4 5
b = 4 cnt: 1 0 1 0 1

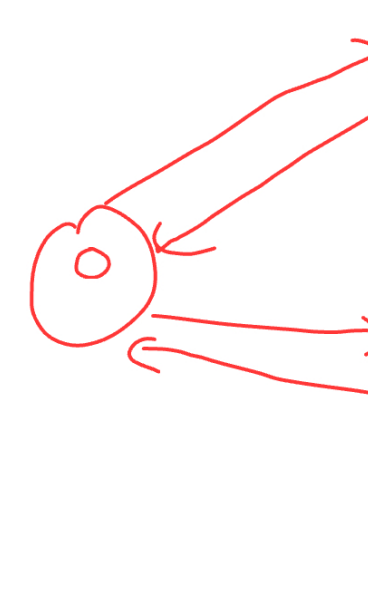
DP

2 "essential" conditions for a problem to be solved using DP:

1) Overlapping Subproblems.
IMP. 2) Optimal soln to the subproblems should lead you to the optimal soln of the overall problem.

F(n) = F(n-1) + F(n-2) 0, 1





Longest Dist (LD)

LD (0, 1) = 2
LD (1, 2) = 2
LD (0, 2) ≠ LD (0, 1) + LD (1, 2)
2 ≠ 2 + 2

LCS

ABCDGH → Num. of subsequences = 2^n

AB = "", A, B, AB

$n \times 2^n$
+
→ $n \times 2^n$
+
$2^n n$

$n \times 2^n$
$m \times 2^m$
2^m

~~$2^n \times 2^m$~~

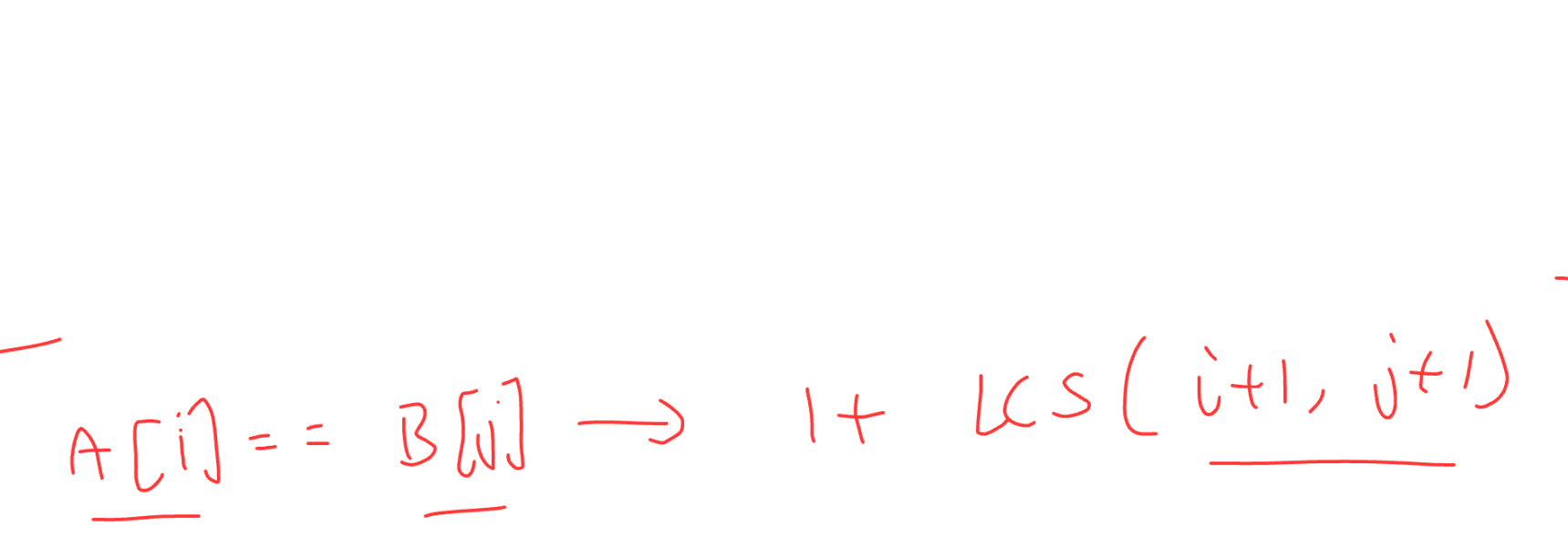
BF: O($n \times 2^n$)

LCS(ABCDGH / AEDFHR) = 1 + LCS(BCDGH / EDFHR)

max [LCS(CDGH / EDFHR), LCS(BCDGH / DFHR)]

A[i] == B[j] → 1 + LCS(i+1, j+1)
otherwise → max(LCS(i+1, j), LCS(i, j+1))

n, m.

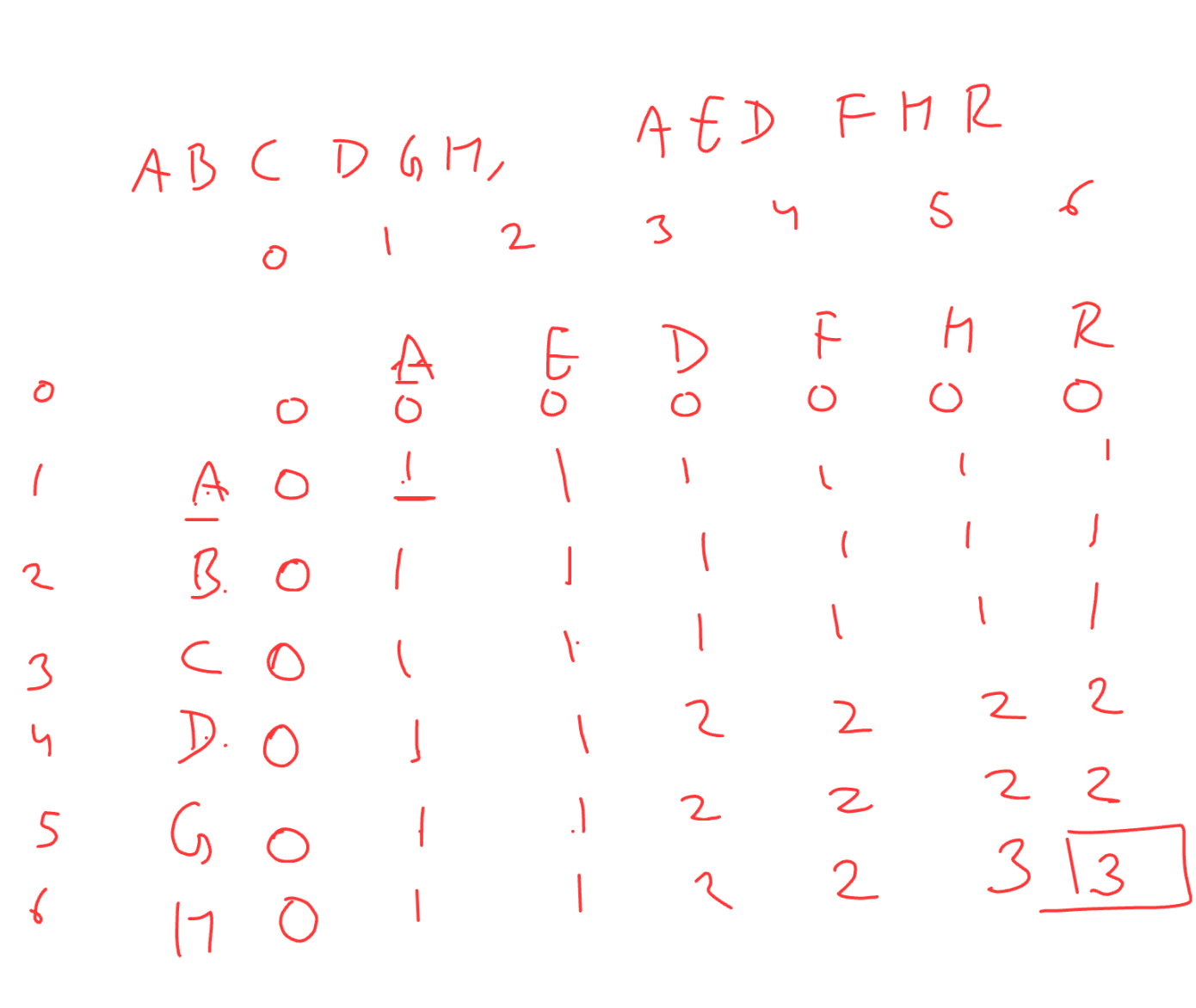


M = max(n, m).

TC = O($2^{\max(n, m)}$)
AS = O(max(n, m)).

TC = No. of rec. calls × work done in 1 rec. call
No. of distinct states of rec. that you are computing.

TC = nm × 1 = O(nm).



ABCDGH, AEDFHR

		""	A	E	D	F	H	R
		0	1	2	3	4	5	6
0		0	0	0	0	0	0	0
1	A	0	1	1	1	1	1	1
2	B	0	1	1	1	1	1	1
3	C	0	1	1	1	1	1	1
4	D	0	1	1	2	2	2	2
5	G	0	1	1	2	2	2	2
6	H	0	1	1	2	2	3	$\boxed{3}$

dp[i][j] ⇒ if s1[i-1] == s2[j-1] → 1 + dp[i-1][j-1]
otherwise. max(dp[i-1][j], dp[i][j-1]).

(A, AE) = max(("", AE), (A, A)).
dp[i-1][j] dp[i][j-1].