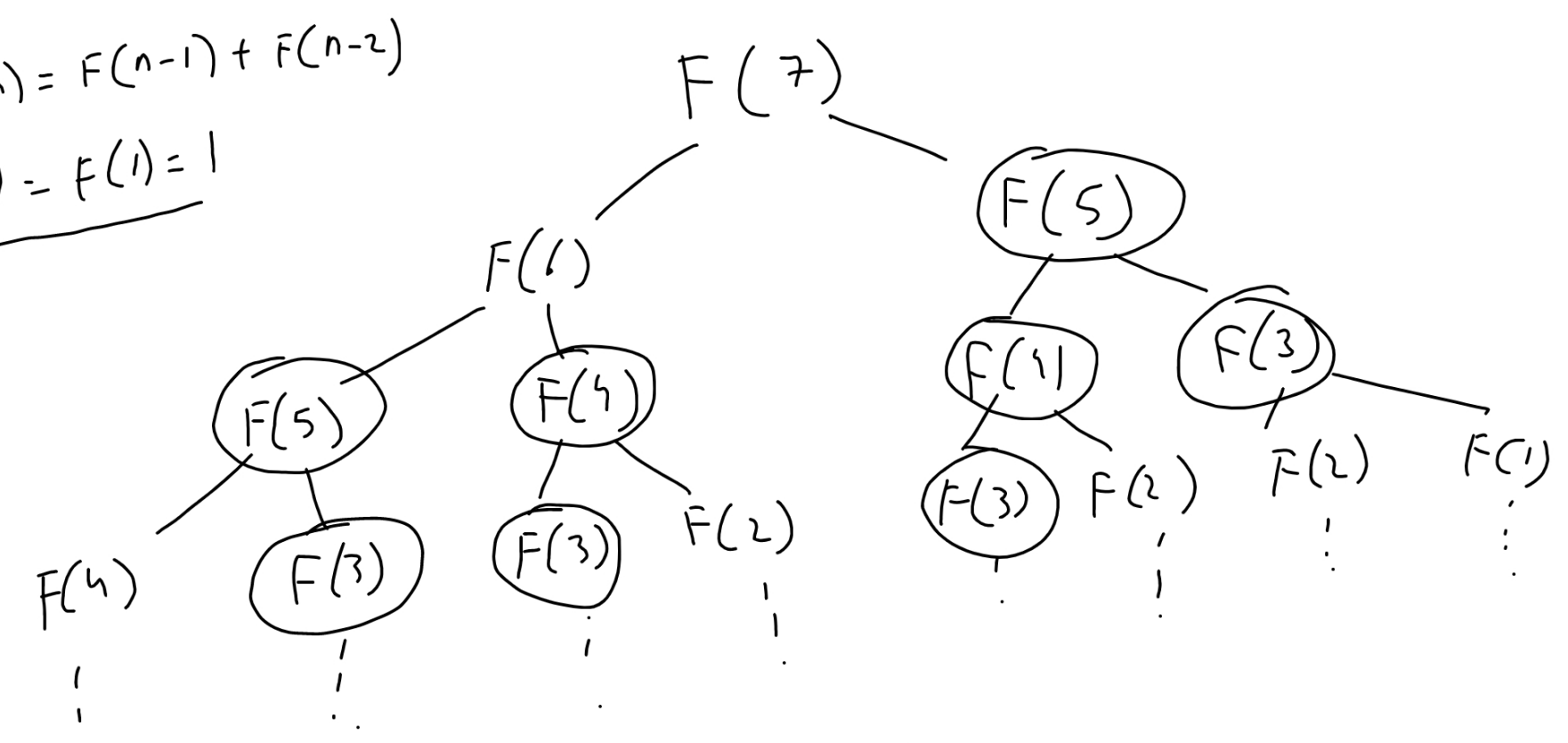
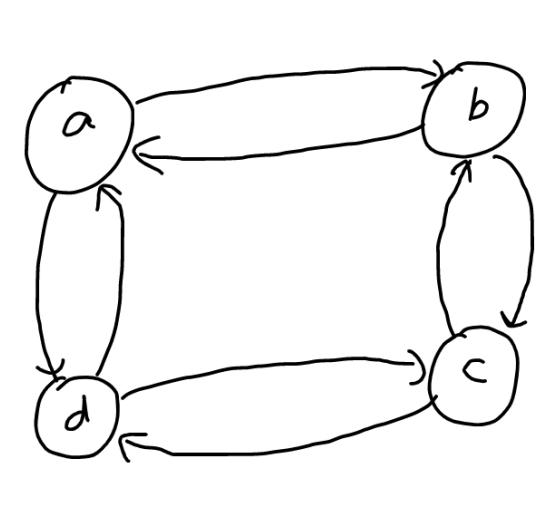
$F(n) = F(n-1) + F(n-2)$
$F(0) = F(1) = 1$


F(7)
F(6)
F(5)
F(5)
F(4)
F(4)
F(3)
F(4)
F(3)
F(3)
F(2)
F(3)
F(2)
F(2)
F(1)

## Overlapping Subproblems

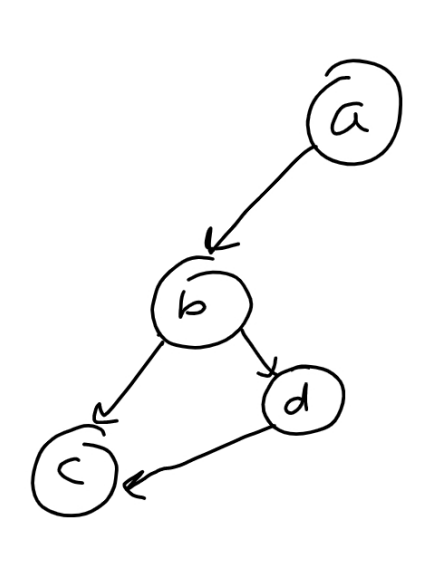


[Longest path b/w a & c:
a → b → c
a → d → c]

× ✓ 2/[4]   × ✓ 2/[5]

$$\left[\underbrace{LP(a,c)}_{=3} \neq \underbrace{LP(a,b)}_{=4} + \underbrace{LP(b,c)-1}_{4}\right]$$

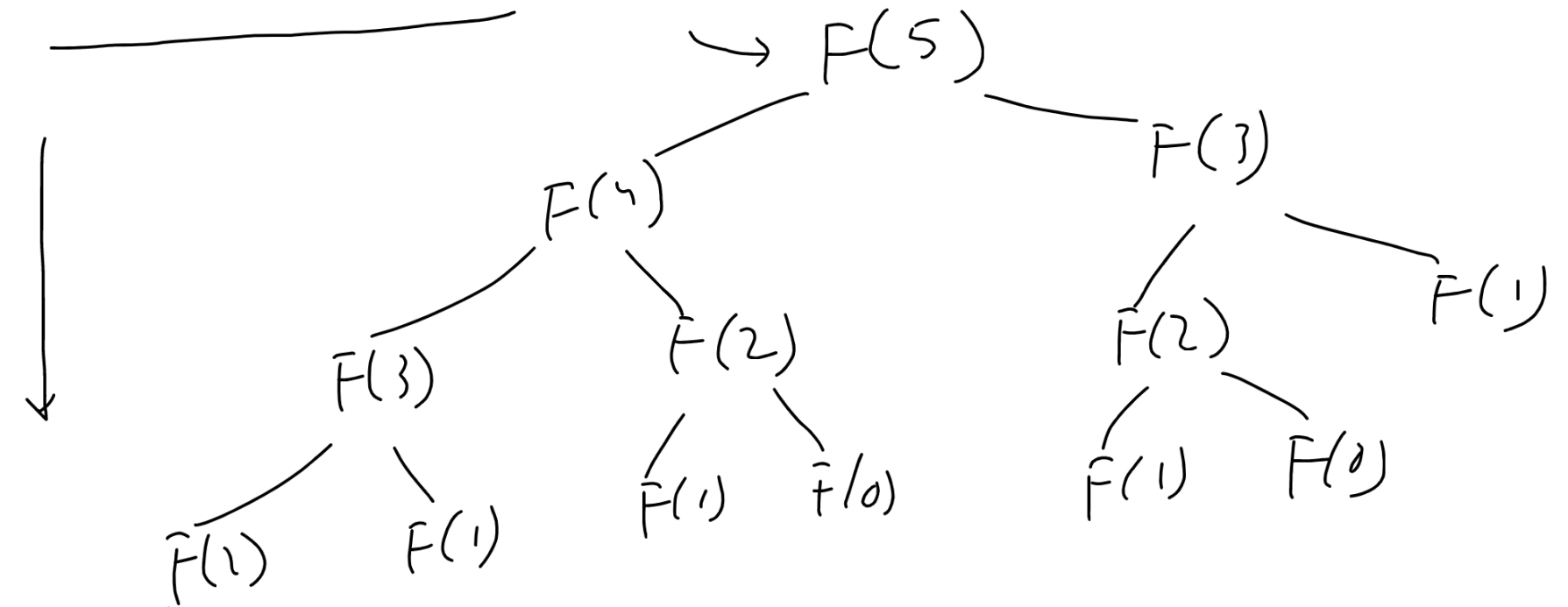
① Recurrence should contain overlapping subproblems
② Optimal soln to the subproblems should result into the optimal soln of the overall problem.



$$\left[\underbrace{LP(a,c)}_{=4} = \underbrace{SP(a,b)}_{=2} + \underbrace{SP(b,c)}_{=2} - 1\right]$$

---

## Top-down DP ✓


F(5)
F(4)
F(3)
F(3)
F(2)
F(2)
F(1)
F(1)
F(1)
F(0)
F(1)
F(0)
F(1)
F(0)

```
int fib(int n) {
if (n <= 1) return 1;
return fib(n-1) + fib(n-2);
}
```

Top-down DP ≡ Solving DP problem recursively

## Bottom-up DP

```
int fib(int n) {
    vector<int> dp(n+1);
    dp[0] = dp[1] = 1;
    for (int i=2; i<=n; i++) {
        dp[i] = dp[i-1] + dp[i-2]
    }
    return dp[n];
}
```

Bottom-up DP ≡ Solving DP problem iteratively.

---

## LCS

$O(m \times 2^{n})$

0 1 2 3 4 5
S1 = ABCDGH
S2 = AEDFHR

Common subsequences: A, AD, $\underbrace{ADH}_{=3}$, AH, DH .......

$LCS(\overset{\downarrow}{A}BCD, \overset{\downarrow}{B}DH) = \max(LCS(BCD, BDH), LCS(ABCD, DH))$

$\underbrace{LCS(\overset{\downarrow}{B}CD, \overset{\downarrow}{B}DH)}_{1+1=2} = 1 + \underbrace{LCS(CD, DH)}_{=1}$

---

$LCS(i, j) = 1 + LCS(i+1, j+1)$ (if (s1[i] == s2[j]))
$= \max(LCS(i+1, j), LCS(i, j+1))$ (otherwise)

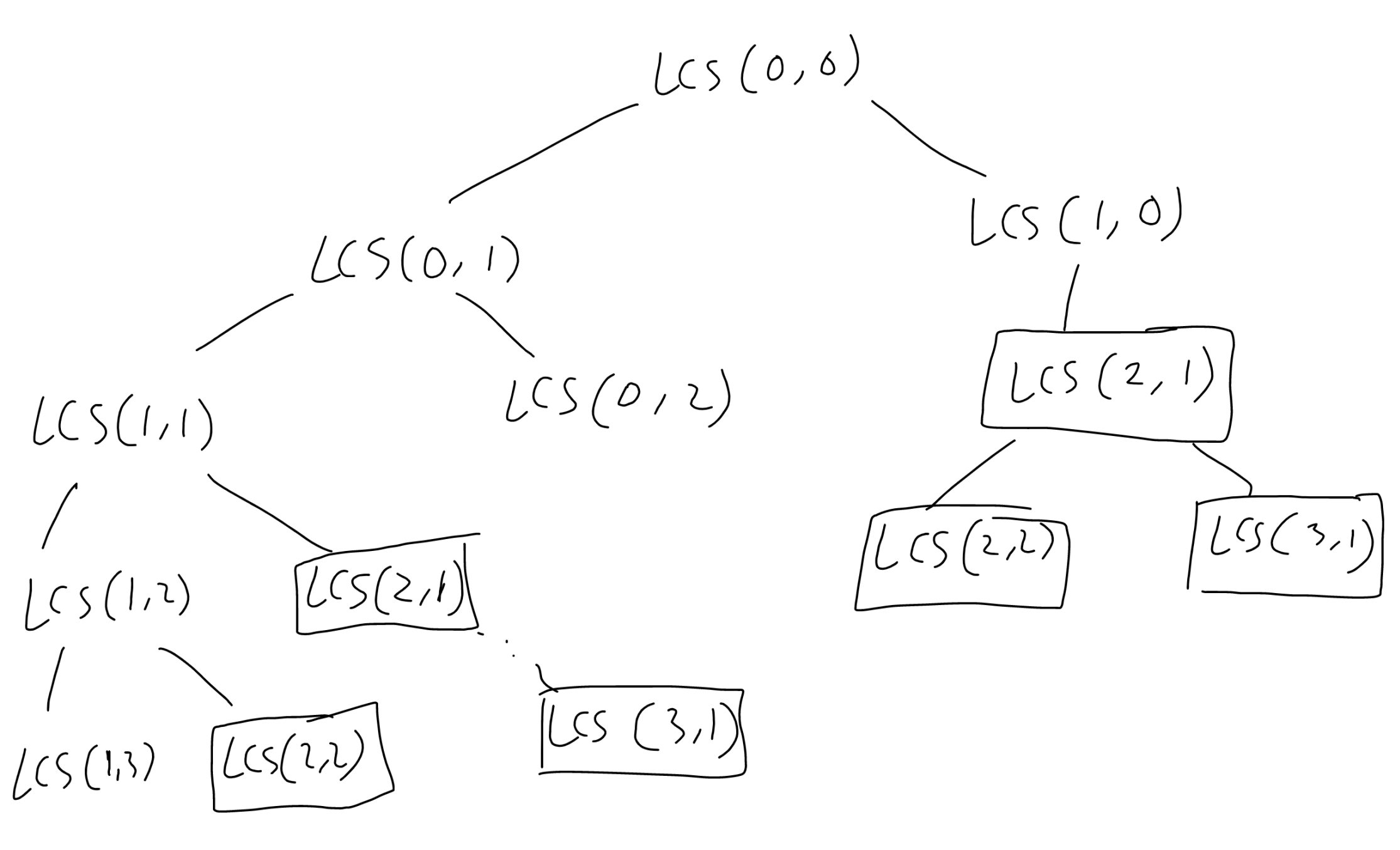
i: Starting index of s1
j: Starting index of s2

---

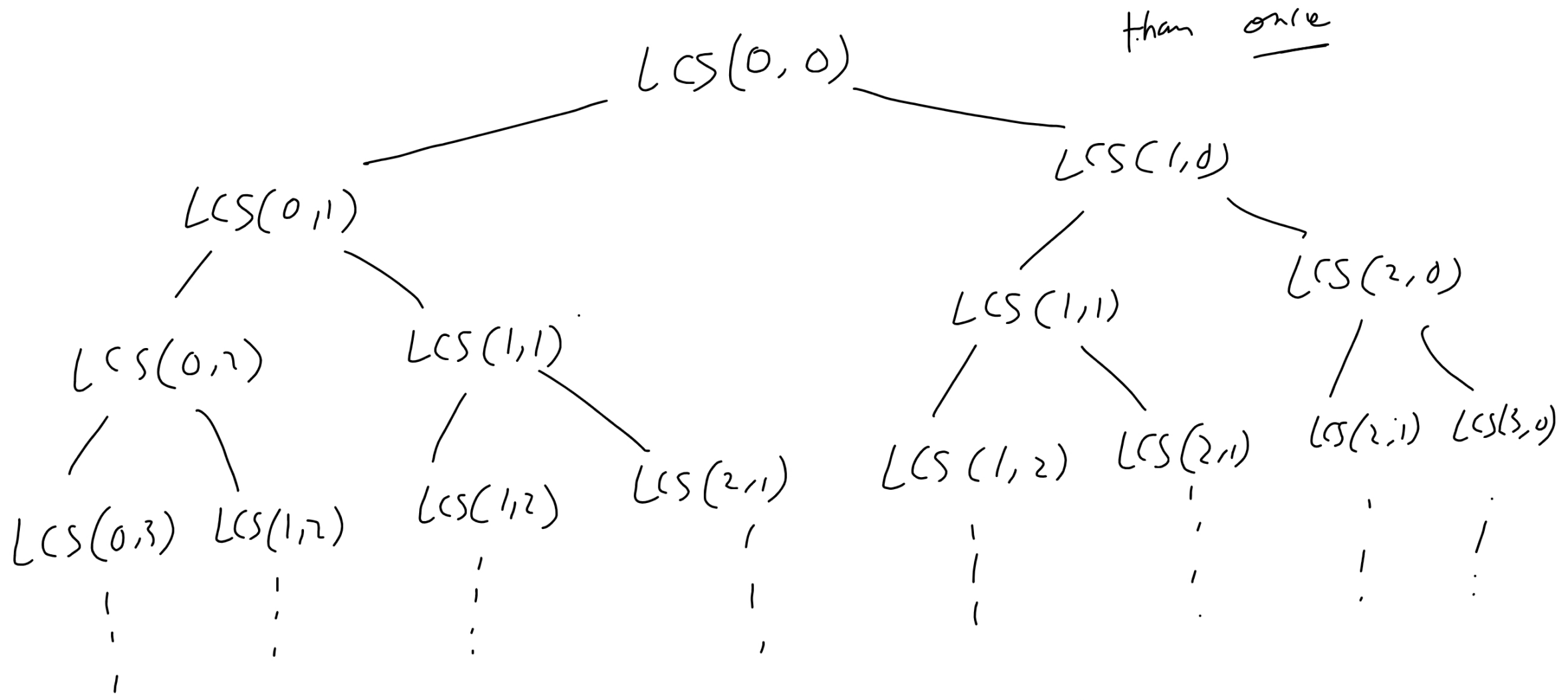
$\underbrace{LCS(\overset{\downarrow}{C}D, \overset{\downarrow}{D}H)}_{=1} = \max(\underbrace{LCS(D, DH)}_{=1}, \underbrace{LCS(CD, H)}_{=0})$

$\underbrace{LCS(\overset{\downarrow}{D}, \overset{\downarrow}{D}H)}_{=1} = 1 + \underbrace{LCS("", H)}_{=0}$

---


LCS(0,0)
LCS(0,1)
LCS(1,0)
LCS(1,1)
LCS(0,2)
LCS(2,1)
LCS(1,2)
LCS(2,1)
LCS(2,2)
LCS(3,1)
LCS(1,3)
LCS(2,2)
LCS(3,1)

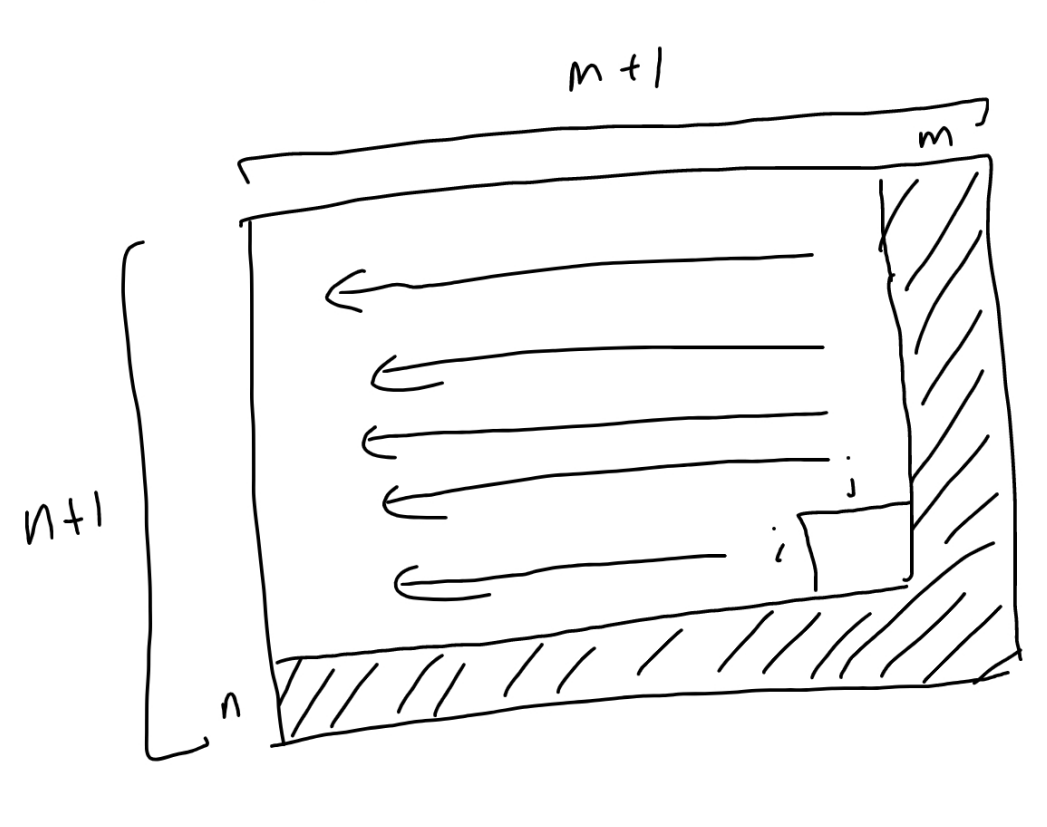
No. of distinct recursive states = O(n * m)
work done by each rec. state = O(1)
$TC = O(n*m)$ ×   wrong bcz a lot of rec. states are overlapping. i.e. being computed more than once


LCS(0,0)
LCS(0,1)
LCS(1,0)
LCS(0,2)
LCS(1,1)
LCS(1,1)
LCS(2,0)
LCS(0,3)
LCS(1,2)
LCS(1,2)
LCS(2,1)
LCS(1,2)
LCS(2,1)
LCS(2,1)
LCS(3,0)

| Level | # of rec. states |
|---|---|
| 0 | 1 |
| 1 | 2 |
| 2 | 4 |
| 3 | 8 |
| ⋮ | |
| i | $2^i$ |

ht. of rec. tree = max(n, m)

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



For computing dp[i][j], we will need:
dp[i+1][j+1],
dp[i+1][j],
dp[i][j+1]

---

## 0-1 KNAPSACK

0 1 2
Values: [60, 100, 120]
wts: [10, 20, 30]
W = 50

o/p = 220

120 + 100 = 220
w = ~~20~~ 0

(0, 1) → 110
(0, 2) → 180
(1, 2) → $\boxed{220}$

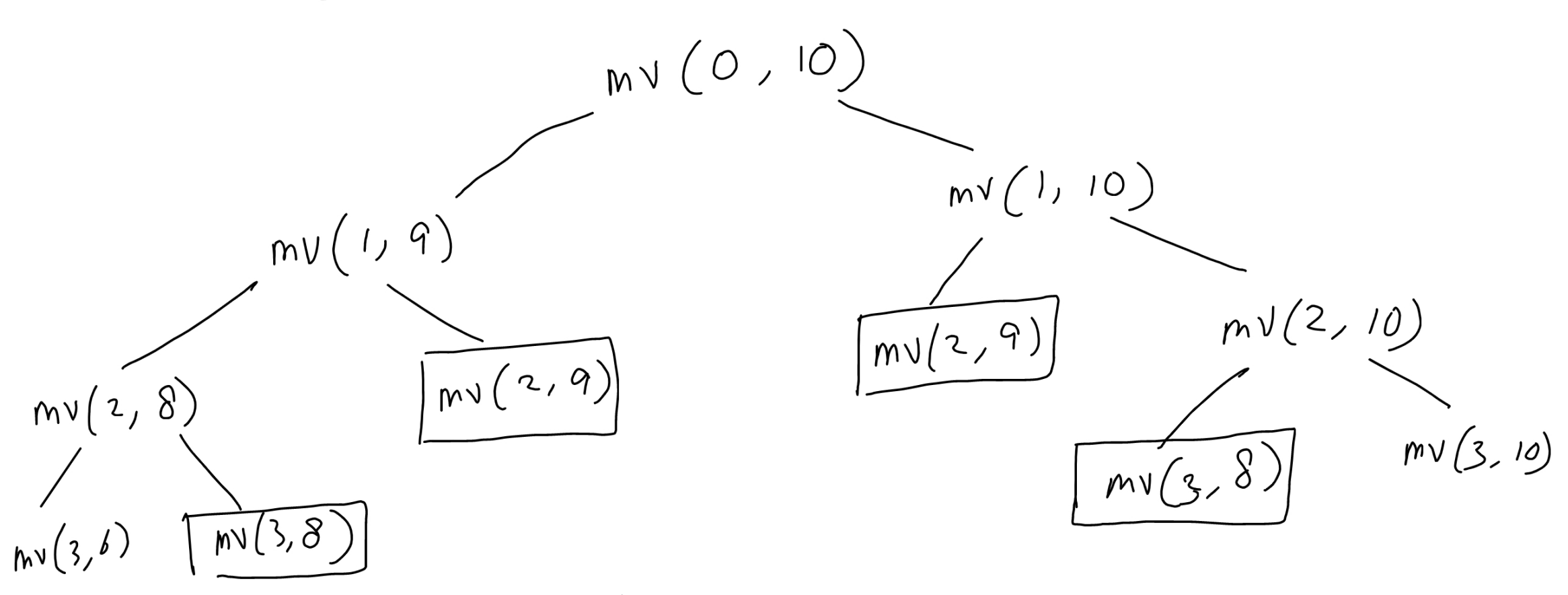
[index of the starting item ← (0, 50) → capacity available]

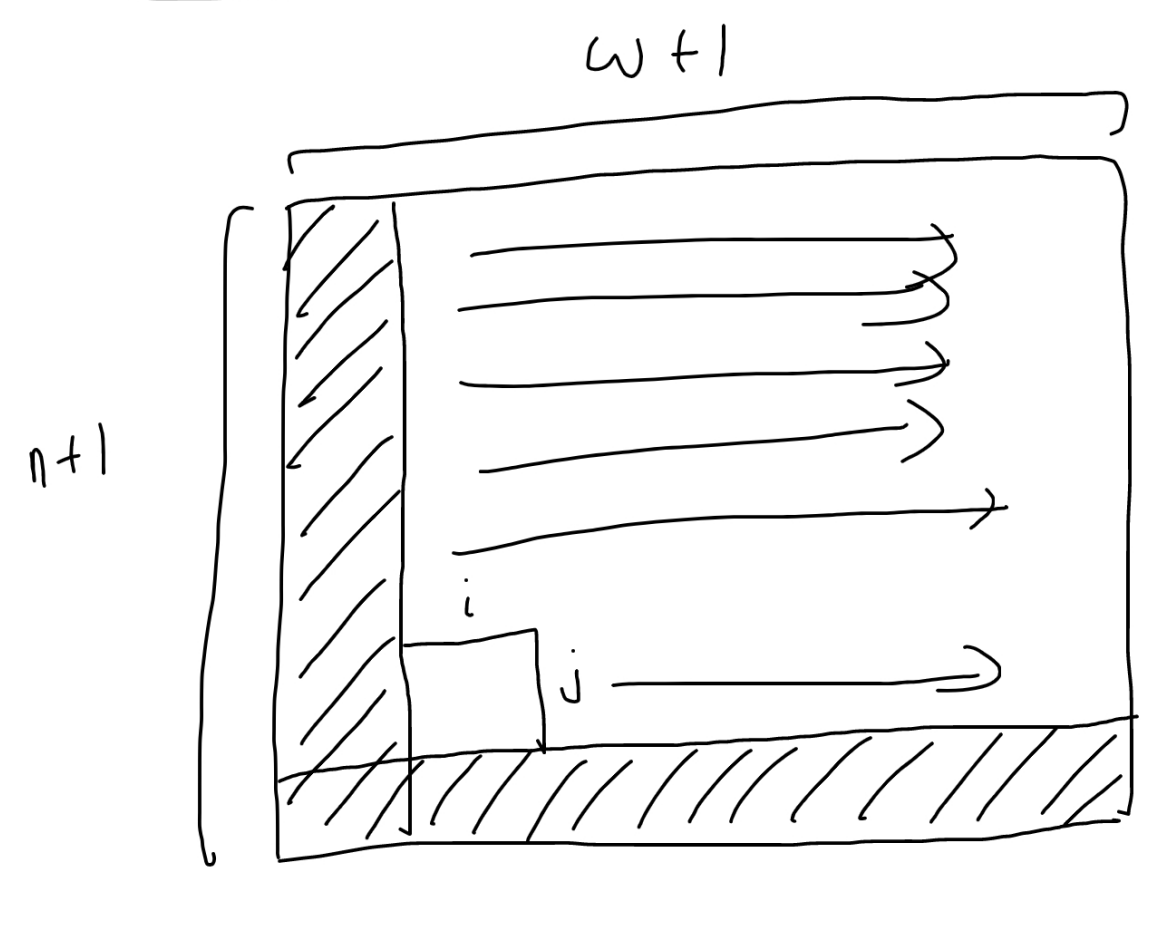
GOAL: compute max_value(0, W)

$\overset{mv}{\text{max\_value}}(i, W) =$
if (wts[i] <= W):
⇒ max(values[i] + mv(i+1, W - wts[i]), mv(i+1, W))
else: mv(i+1, W)

0 1 2 3
Values: [1, 2, 3, 4]
wts: [1, 1, 2, 3]

W = 10


mv(0, 10)
mv(1, 9)
mv(1, 10)
mv(2, 8)
mv(2, 9)
mv(2, 9)
mv(2, 10)
mv(3, 6)
mv(3, 8)
mv(3, 8)
mv(3, 10)



For dp[i][j], we will need:
dp[i+1][j]
dp[i+1][j - wts[i]]

---

$${}^nC_r = \frac{n!}{r!\,(n-r)!}$$

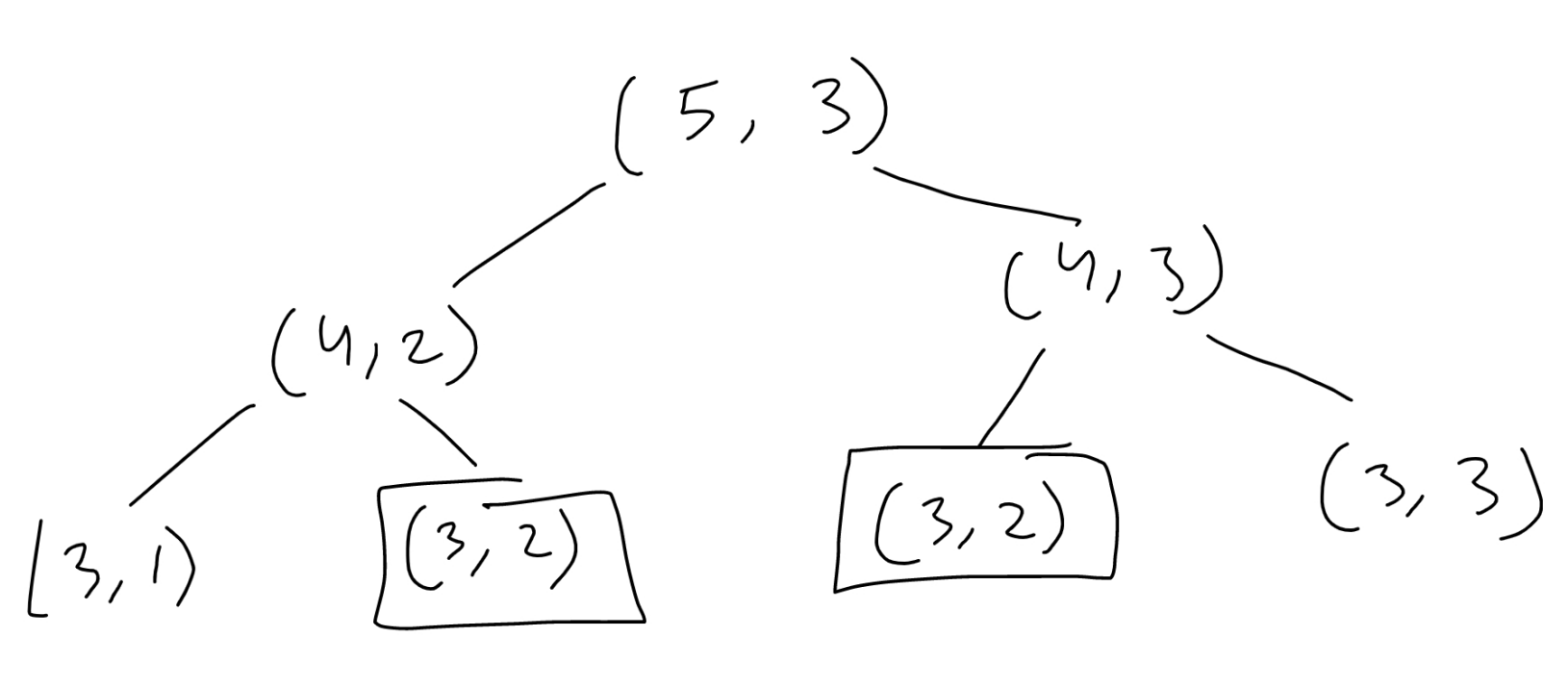
$$\left[{}^nC_r = {}^{n-1}C_{r-1} + {}^{n-1}C_r\right] \#$$

$$\frac{(n-1)!}{(r-1)!\,(n-r)!} + \frac{(n-1)!}{r!\,(n-r-1)!}$$

$$= \frac{(n-1)!}{(r-1)!\,(n-r-1)!}\left[\frac{1}{(n-r)} + \frac{1}{r}\right]$$

$$= \frac{(n-1)!}{(r-1)!\,(n-r-1)!}\left[\frac{\cancel{r} + n - \cancel{r}}{r(n-r)}\right]$$

$$= \frac{n!}{r!\,(n-r)!} = {}^nC_r$$


(5, 3)
(4, 2)
(4, 3)
(3, 1)
(3, 2)
(3, 2)
(3, 3)