Starting 7:05

Today's Agenda:

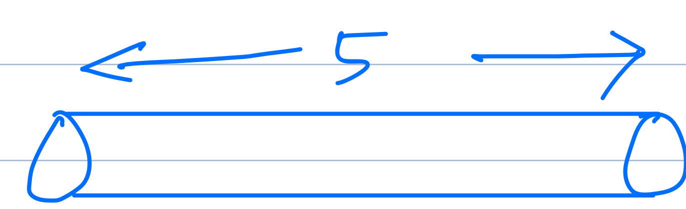
1. Rod Cutting
2. Coin Change — 2 variations
3. 0-1 Knapsack (Tighter Constraints)

Rod Cutting

Given a rod of length N & an array of length N.
arr[i] → price of i-length rod.
Find the max value that can be obtained by cutting the rod into 1 or more pieces & selling them.

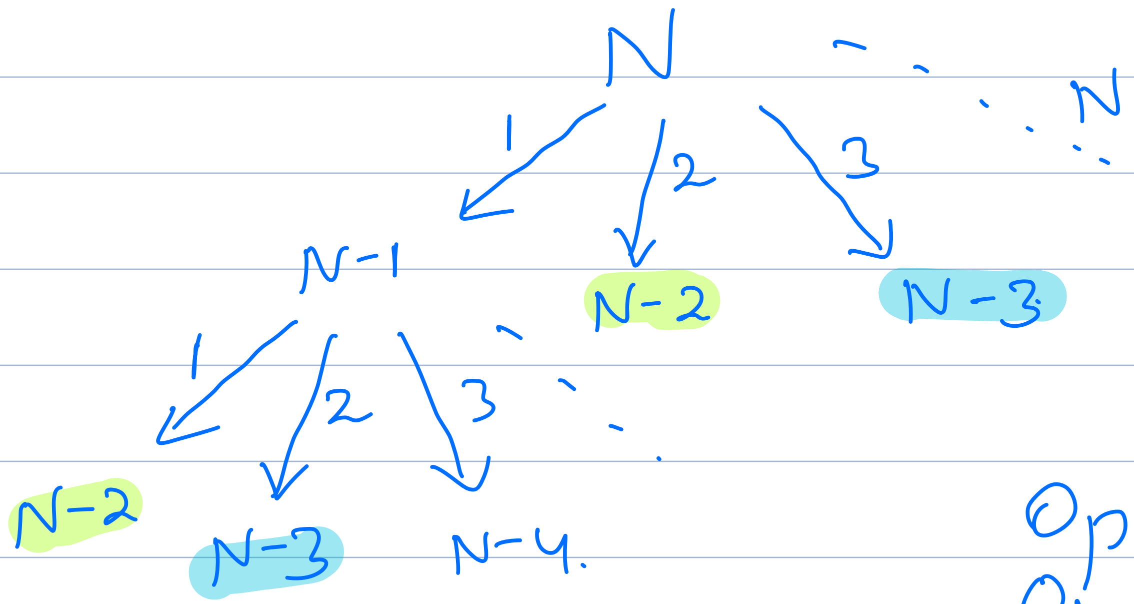
N = 5

arr → | 1 | 4 | 2 | 5 | 6 |
1 2 3 4 5



ans = 9.

Sold Length	Cost
5	6
4 + 1	6
3 + 2	6
2+2+1	4+4+1 = 9
2+1+1+1	7
1+1+1+1+1	5
3+1+1	4.



1 + BA(4)
2 + BA(3)
3 + BA(2)
4 + BA(1)
5 + BA(0)

Optimal S.S. ✓
Overlapping S.P ✓

dp(i) → Best S.P. for rod of length i.

sp →

1	2	3	4	5
1	4	2	5	6

dp() →

0	1	2	3	4	5
0	1	4.	5	8	9.

dp[5]-

sp[1] + dp[4]
sp[2] + dp[3]
sp[3] + dp[2]
sp[4] + dp[1]
sp[5]

9, 9, 6, 6, 6

sp[1] + dp[2]
sp[2] + dp[1]
sp[3] + dp[0]

Code:

```
dp[n+1] , ∀i dp[i] = 0;
for (i=1; i <= N; i++) {
    for (cut = 1 ; cut <= i ; cut++) {
        dp[i] = max(dp[i],
                    sp[cut] + dp[i-cut]);
    }
}
return dp[N];
```

T.C - O(N²)
S.C — O(N)

Coin Change

N different denominations.
Total no. of ways to pay a given amount.

Any denomination any no of times.

(x, y) ≠ (y, x)

Amount = 5

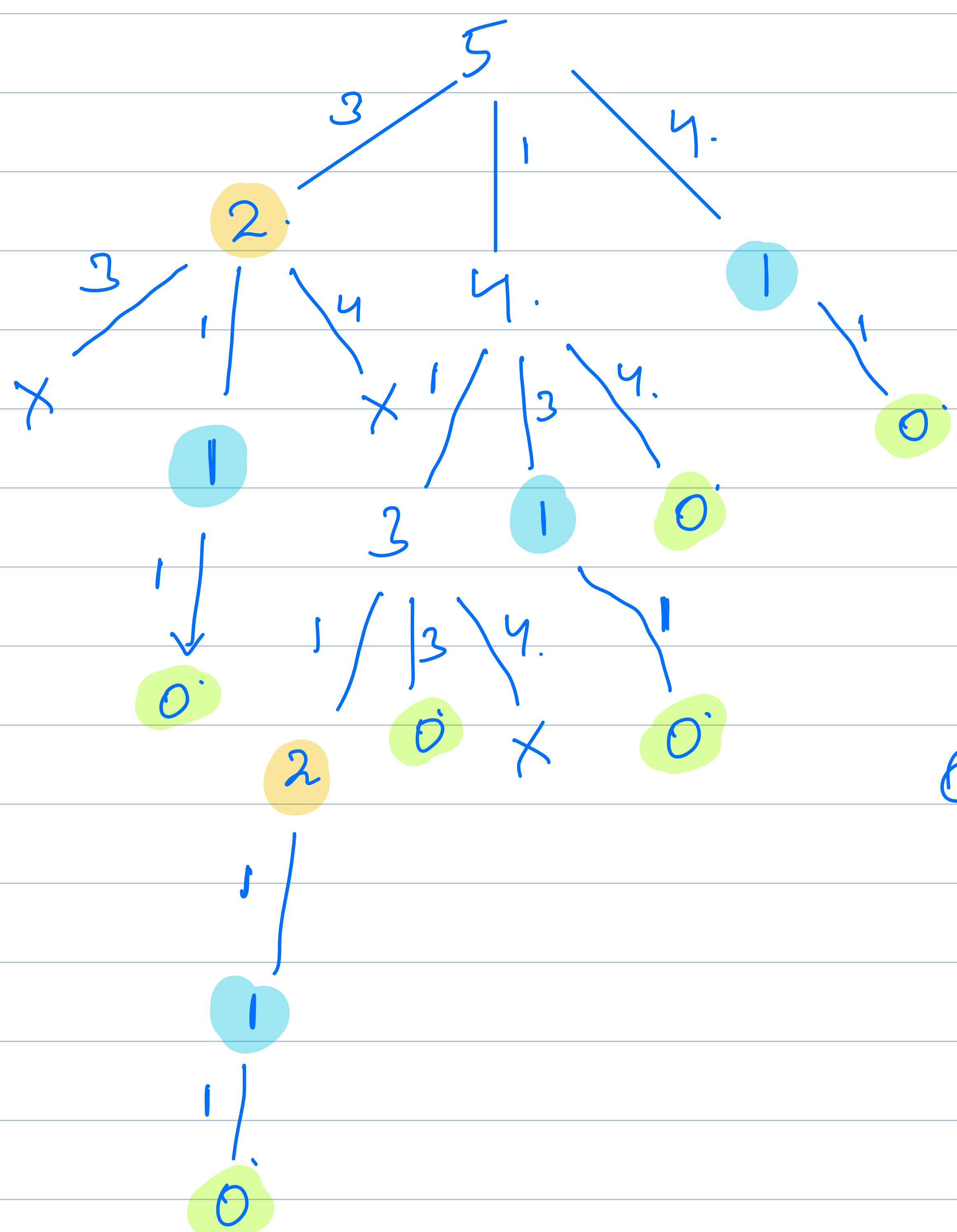
denoms → [3 1 4]

(1, 4) (1, 1, 3) (1,1,1,1,1)

(4, 1) (1, 3, 1)

(3, 1, 1)

ans = 6.



Optimal S.S.

Overlapping S.P

$$N = \sum_{i=1}^{N} (N - d[i]).$$

different denomination

No. of ways to pay Rs. 0 using different denominations — 1. ✓

(not paying anything)

~~0~~

$dp(i) \rightarrow$ Total no. of ways to pay Rs i.

Amount = 5.

denoms → [3 1 4]

dp →

1	1	1	2	4	6
0	1	2	3	4	5

$$dp[5-1] + dp[5-3] + dp[5-4]$$

Code:

```
dp[amount + 1]    , ∀i dp[i] = 0;

    dp[0] = 1;
for (i = 1; i <= amount; i++){
    for (j = 0; j < denoms.length(); j++){
        if (i - denoms[j] >= 0) {
            dp[i] += dp[i - denoms[j]];
        }
    }
}
    return dp[amount];
```

T.C. → $O(N * amount)$.
S.C → $O(amount)$.

8:15.

Variation 2 :-

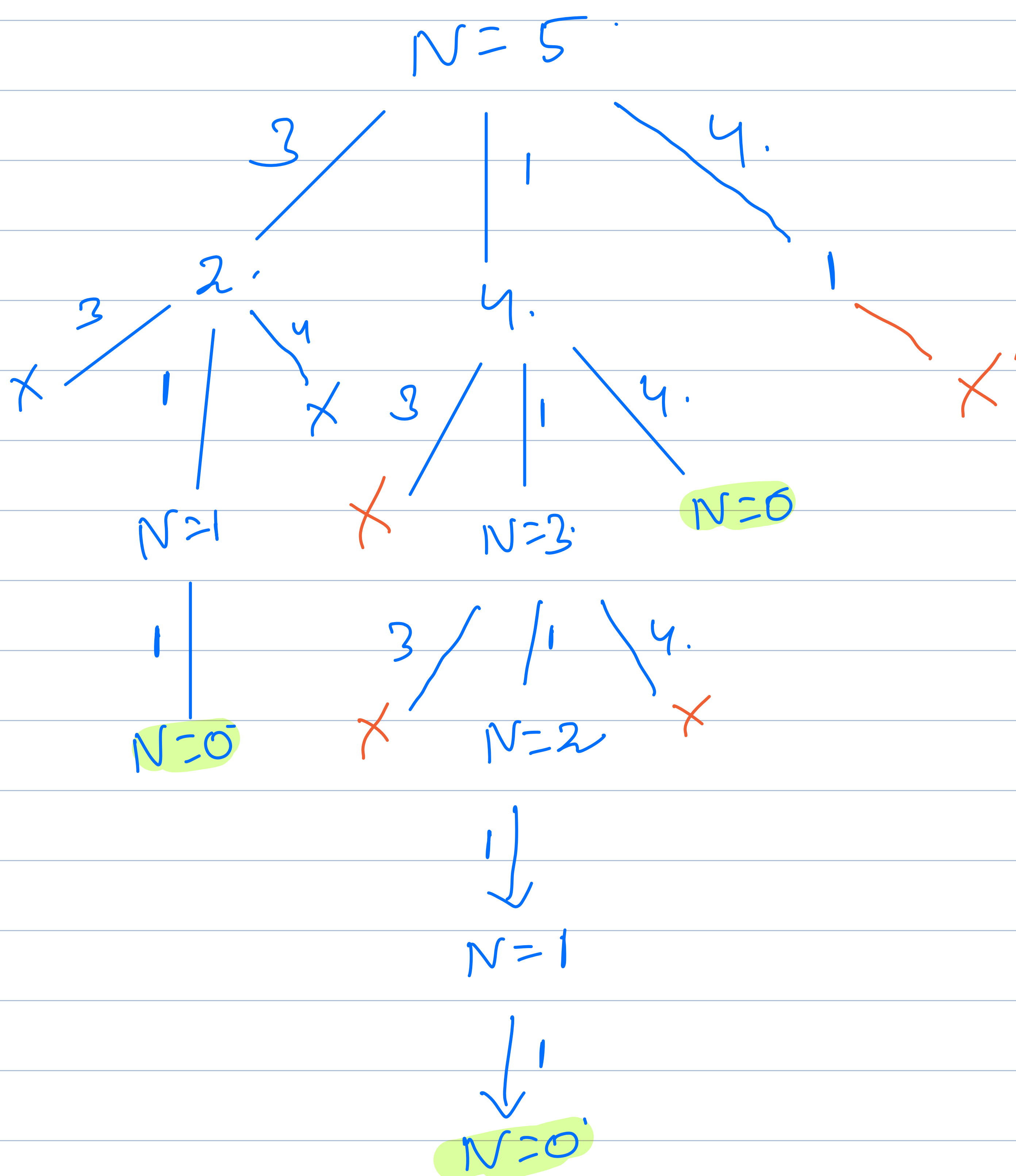
$$(x, y) = (y, x)$$

$N = 5$ denom $= \{3, 1, 4\}$.

$(3, 1, 1)$, $(1, 4)$, $(1, 1, 1, 1, 1)$

✗ $(1, 3, 1)$ ✗ $(4, 1)$

✗ $(1, 1, 3$



Once I used a denomination, I cannot go back to any previous denom.

denoms → [3 1 4].

dp →

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

→ After using 3.

dp[1 - 1]

dp →

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

dp[5 - 1]

↳ After using 3 & 1

3
1 1 1

3, 1
1, 1, 1, 1
1, 4.

3, 1, 1
1, 1, 1, 1, 1
1, 1, 4.

dp →

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

final state.

Code:

```
dp[amount + 1];   ∀i dp[i] = 0;
   dp[0] = 1;

for (j = 0;  j < denoms.size();  j++) {
    for (i = denoms[j];  i <= amount ;  i++) {
        dp[i] += dp[i - denoms[j]];
    }
}
    return dp[amount];
```

0-1 Knapsack

We are given N toys with their happiness & weight. Find max total happiness that can be kept in a bag with capacity W.

Toys can't be divided.

Constraint -

$dp[N][W]$

$N \times W$

↓

W

$1 \le N \le 500$

$1 \le W \le 10^9$

$1 \le wt[i] \le 10^9$

$1 \le value[i] \le 50$

$500 \times 10^9 = 5 \times 10^{11}$

TLE

$10^7 \sim 10^8$

A person going to buy a car.

Type 1

20 lakhs

Type 2

$L_1 \quad L_2 \quad L_3 \quad L_4$

$\downarrow \quad \downarrow \quad \downarrow \quad \downarrow$

15L 25L 19L 50L

→ Max value that can be generated using first i elements & capacity of my bag equals to j.

Type 1

→ Min weight of my bag if I need to generate a value j with first i elements.

Type 2 :

N * V.

→ dp[i][j] → Min cost required to get value j with first i elements.

Max. No. of Items * Max Value.

500   50 * 500

T.C. — 500 × 50 × 500
= 1.25 * 10^7. (Not TLE).

4, 5, 6.

dp[i][0] → 0.
dp[0][j]

dp[i]