Coin Sum

You are given a set of coins A. In how many ways you can make sum B. You can't take one denomination more than once.

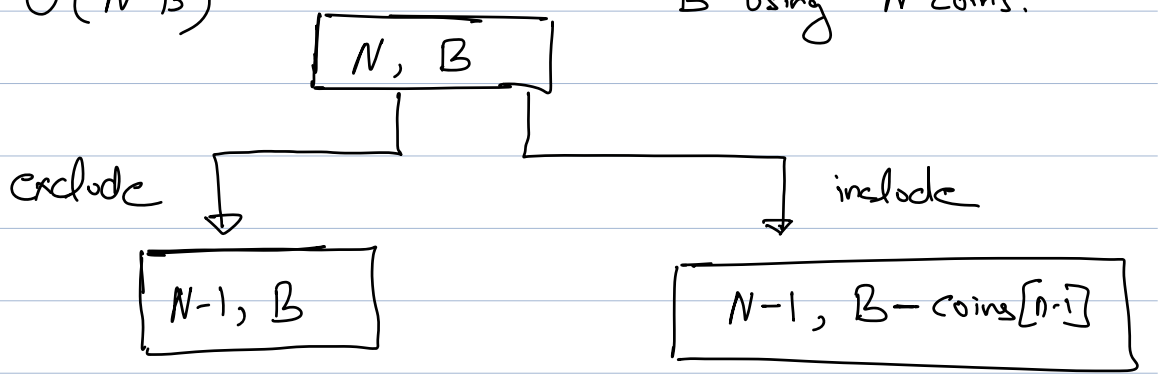
Ex1 coins[] = { 7 4 9 6 10 13 11 14 }

B = 22

⇒ 9, 13
⇒ 7, 9, 6 → ans = 2
⇒ 7, 4, 11

TC: O(NB)
SC: O(NB)

→ how many ways to make B using N coins.

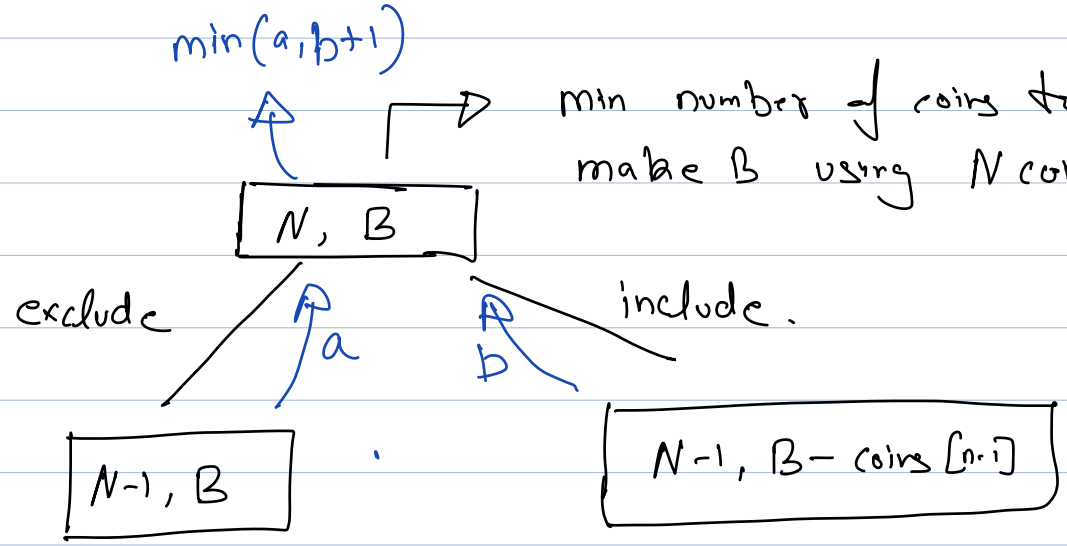


Base Case: 1) if (B == 0) return 1
else if (N == 0) return 0

$$dp[i][j] = \begin{cases} dp[i-1][j] + \\ dp[i-1][j - coins[i-1]] \end{cases}$$

↳ if (j ≥ coins[i-1])

Min number of coins to get sum B.



Base Case :

```
if (B==0)
    return 0
else if (N==0)
    return MAX.INTEGER
```

Coin Sum Infinite

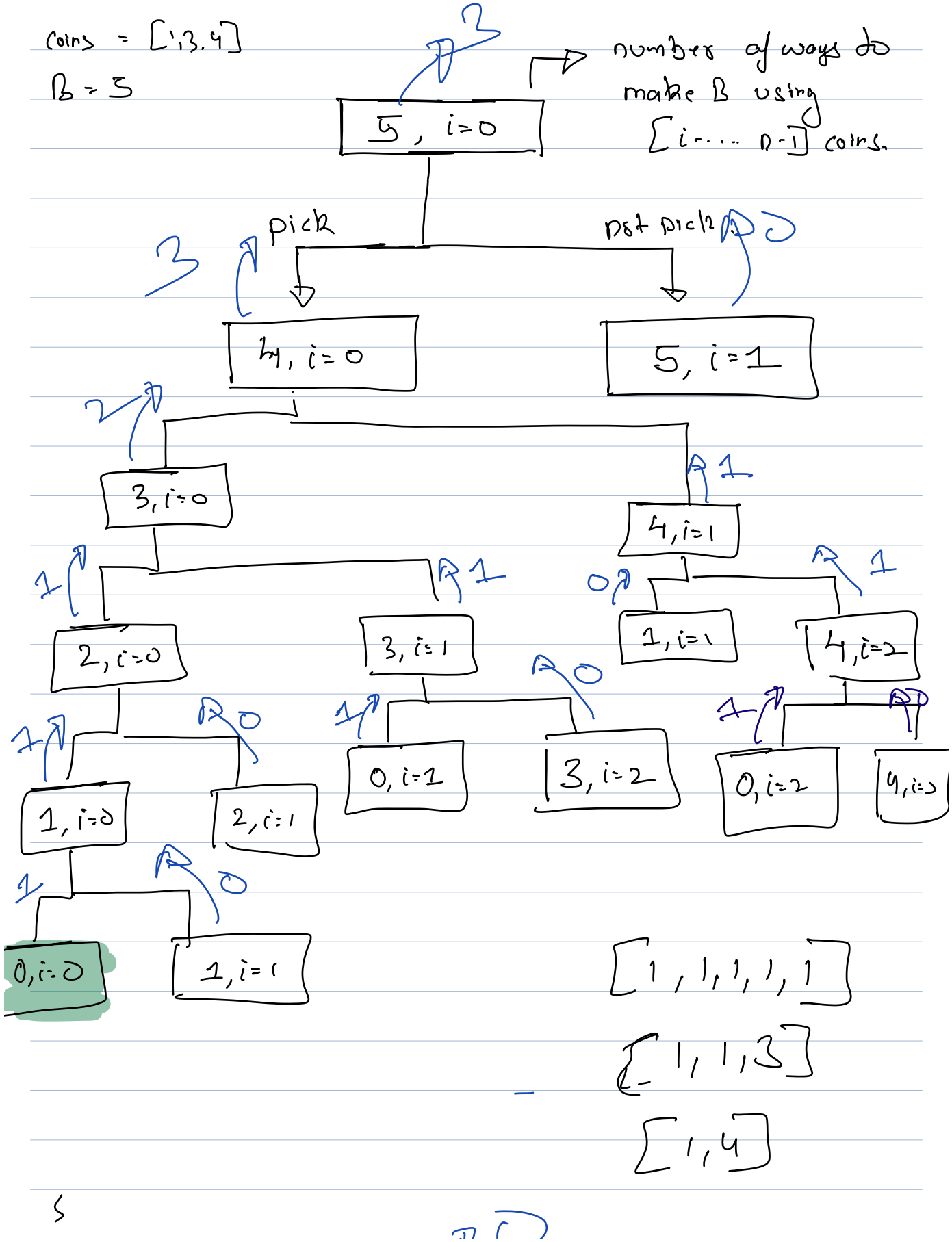
A coin can be taken infinite times.

Ex1 coins[] = {1, 3, 4}
B = 5.

Order is not imp	Order is important
{1, 1, 1, 1, 1} {1, 1, 3} {1, 4}	{1, 1, 1, 1, 1} [(1 1 3), (1 3 1), (3, 1 1)] [{1, 4}, {4, 1}]
↓	↓
3	6

coins = [1,3,4]

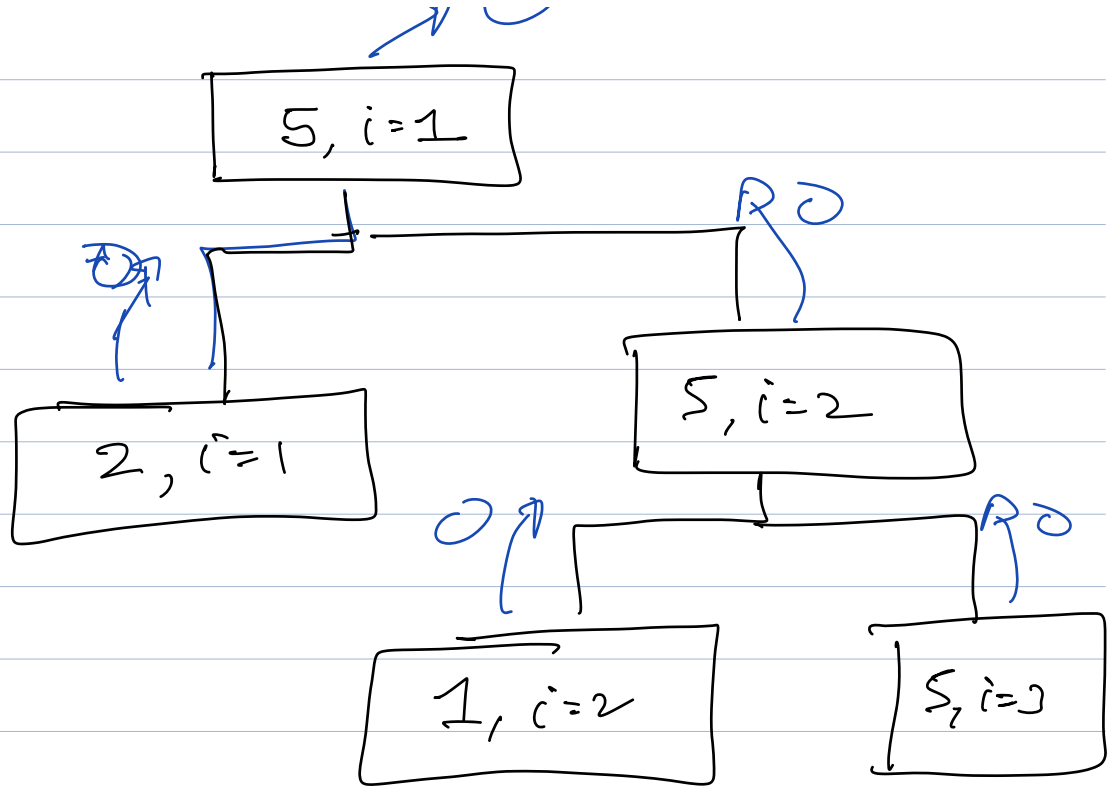
B = 5

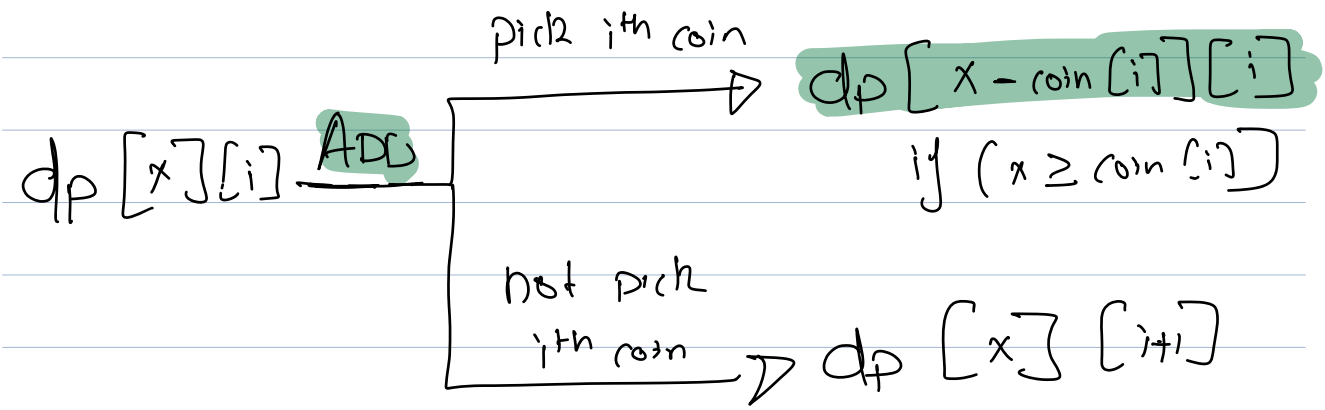


[1, 1, 1, 1, 1]

[1, 1, 3]

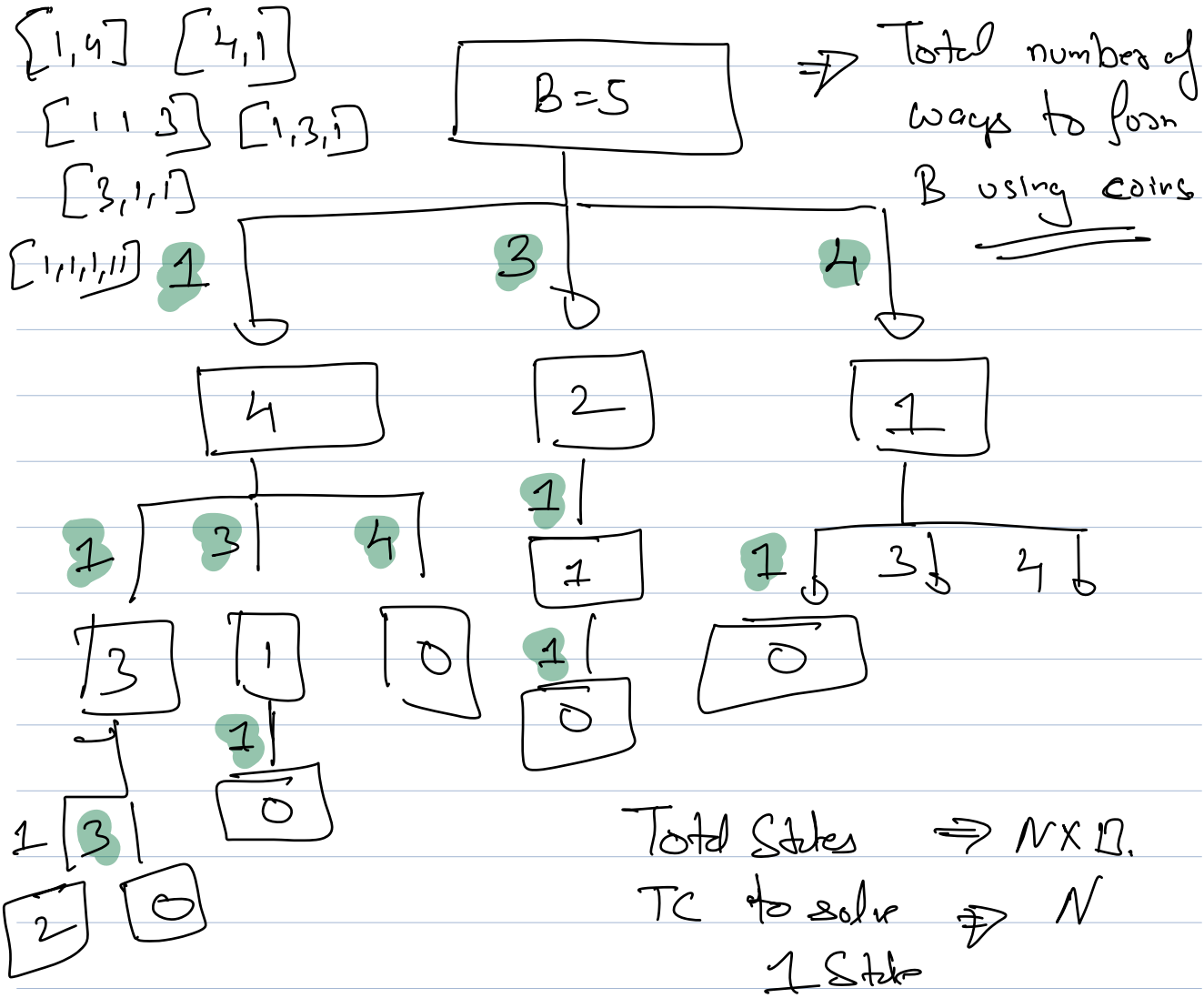
[1, 4]





B x N TC ⇒ O(NlogN + BN)

SC ⇒ O(BN)



Total States ⇒ N X B.

TC to solve 1 State ⇒ N

TC ⇒ $O(N \log N + N^2 B)$

↓

Sorting.

SC ⇒ $O(NB)$

Knapsack

$n = 5$

$C = 8$

	0	1	2	3	4
Value	12	20	15	6	10
Weight	3	6	5	2	4

Value	Weight		0	1	2	3	4	5	6	7	8
		0	0	0	0	0	0	0	0	0	0
12	3	1	0	0	0	12	12	12	12	12	12
20	6	2	0	0	0	12	12	12	20	20	20
15	5	3	0	0	0	12	12	15	20	20	27
6	2	4	0	0	6	12	12	18	20	21	27
10	4	5	0	0	6	12	12	18	20	22	27

func(int i, int j)

max

- did not pick ith item → func(i-1, j)
- pick ith item → func(i-1, j - wt[i-1]) + val[i-1]

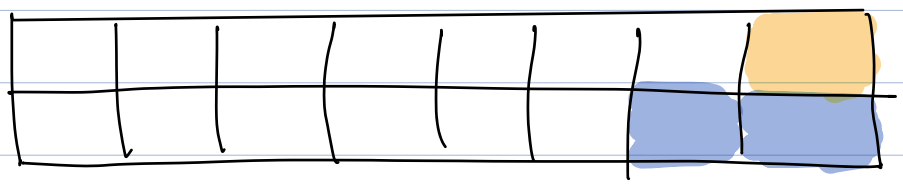
```
i = N, j = C
while (i ≥ 0 && j ≥ 0) {
  if (dp[i][j] == dp[i-1][j]) {
      i--;
  } else {
      item.insert(i);
      i--;
      j = j - wt[i-1];
  }
}
```

TC : $O(N \times C)$

SC : $O(N \times C)$

$N = 500$

$C \Rightarrow 10^5$



func (i-1, j) func (i-1, j - wt[i-1])

```
for (int i=1 ; i<=N ; i++) {
    for( int j = 1 ; j<= C ; j++) {
        if ( j >= wt[i-1]) {
            dp[j] = max ( dp[j],
                dp[j - wt[i-1]]
                + val[i-1];
            )
        }
    }
}
```

$\Rightarrow$ $O(C)$