Current PSP 52.15 → 55

Nov23_PSP_29Apr
Gobika K
Vijay V A
kameswarreddy Yeddula
Piyush Kumar
Sai Sharath
Manjunatha I
Harshil Dabhoya
Yash Malviya
Rajeev
Kevin Theodore E
Suraj Devraye
Mohammad Mateen

Nov23_PSP_29Apr
manikandan m
MD JASHIMUDDIN
sudhakar venkatachalam
Mayur Hadawale
Prashant Kumar Soni
Pradeep Kumar Chandra
Shaurya Srivastava
Vigneshwaran K
SABBAVARAPU KARTHIK
Mohammed Arshad
SIJU SAMSON
Pushkar Deshpande

Agenda

1. Cutting a rod
2. Coin sum Infinite
3. 0-1 Knapsack (modified)

Quiz 1 : T.C = O(N * C)

Q → Given a rod of length N, and an array A of length N

A[i] → price of i length rod

Find max value that can be obtained by cutting the rod into pieces and selling them

N = 5

A[i] =

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

S.No	cuts	value	ans
1	5	6	6
2	4+1	5+1	max(6,6) = 6
3	3+2	2+4	max(6,6) = 6
4	2+2+1	4+4+1	max(6,9) = 9
5	1+1+1+1+1	5	max(9,5) = 9

ans = 9

Ques 2: Unbounded Knapsack

DP Observation :

(we can keep using same cut multiple times)

DP state :

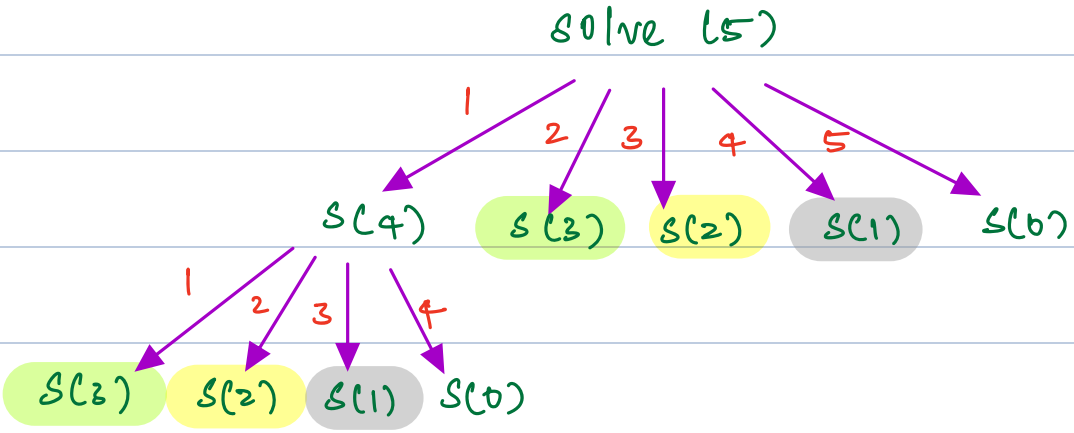
DP[N] ⟶ maximum value for length N

DP[0] = 0

N = 5

A[i] =

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



Optimal substructure ⟶ Yes

Overlapping subproblem ⟶ Yes

100% DP problem

pseudo code

```
// Initialize  dp[i] = 0

for (i = 1; i <= n; i++) {
    for (j = 1; j <= i; j++) {
        DP[i] = max (DP[i], A[j] + DP[i-j]);
    }
}
return DP[N];
```

// Dry Run

N = 5

A[i] =

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

DP[i] =

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

i	j	A[j] + DP[i-j]	DP[i]
1	1	1	1
2	1, 2	2, 4	4
3	1, 2, 3	5, 5, 2	5
4	1, 2, 3, 4	6, 8, 3, 5	8

Q → In how many ways can the sum be equal to N by using coins given in the array ? One coin can be used multiple times

a) Ordered Selection → $(x, y) \neq (y, x)$

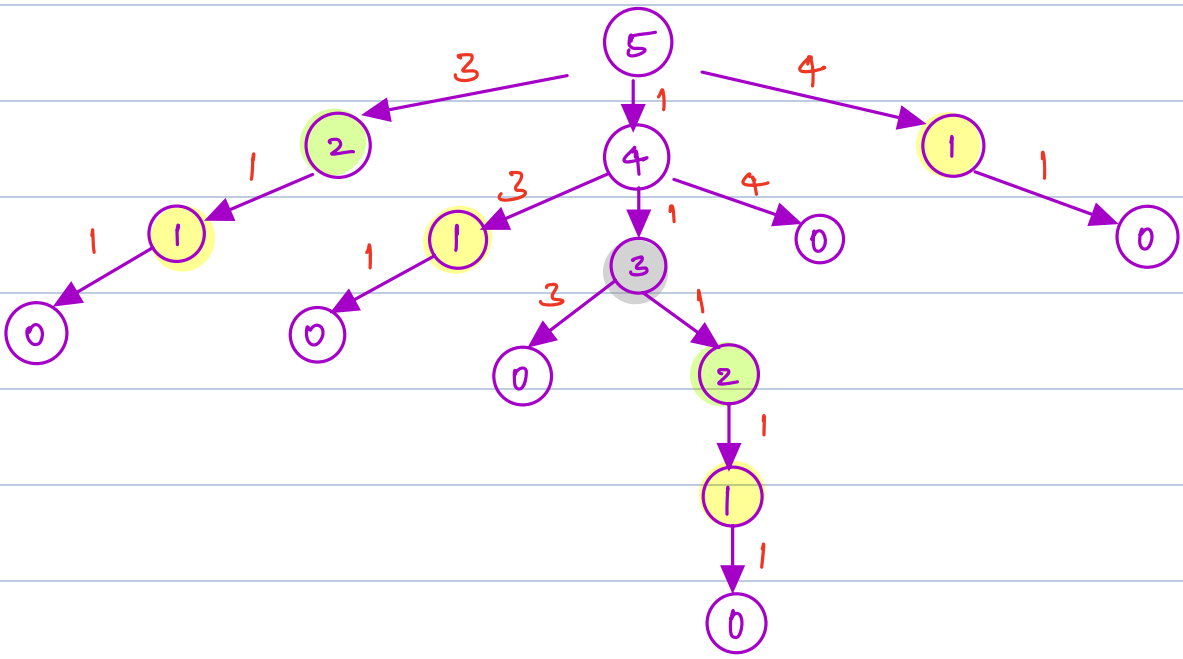
N = 5

A = | 3 | 1 | 4 |
    | 0 | 1 | 2 |

1 + 4	3 + 1 + 1	1 + 1 + 1 + 1 + 1
4 + 1	1 + 3 + 1	
	1 + 1 + 3	

Observation

Unbounded Knapsack DP[N] → ways of coins to get sum N



Optimal Substructure → 100% Yes

Overlapping subproblem → 100%. Yes

DP

DP state

DP[N] = count of ways to form sum N using all coins

Ques 3

Number of ways of getting DP[0] ?

DP[0] = 1 (don't pick any coin)

psendo code

```
// Initialize DP[i] = 0  ∀ i
DP[0] = 1 ;
for (i=1; i<=n; i++) {  // Get count of ways for i
    for (j = 0; j < A.length ; j++) {
        if (A[j] <= i) {
            DP[i] = DP[i] + DP[i-A[j]]
        }
    }
}
return DP[N];
```

T.C = (n * A.length)

S.C = O(n)

Dry Run

N = 5

A =

3	1	4
0	1	2

DP[] =

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

i	j	DP[i - A[j]]	DP[i]
1	0, 1, 2	~~0~~, ✓1, ~~2~~	1
2	0, 1, 2	~~0~~, 1, ~~2~~	1
3	0, 1, 2	0, 1, ~~2~~	2
4	0, 1, 2	0, 1, 2	4
5	0, 1, 2	0, 1, 2	6

B → Unordered Selection

$$(i, j) = (j, i)$$

N = 5

A =

3	1	4
0	1	2

1 + 4	3 + 1 + 1	1 + 1 + 1 + 1 + 1
4 + 1	1 + 3 + 1	
	1 + 1 + 3	

3 ways

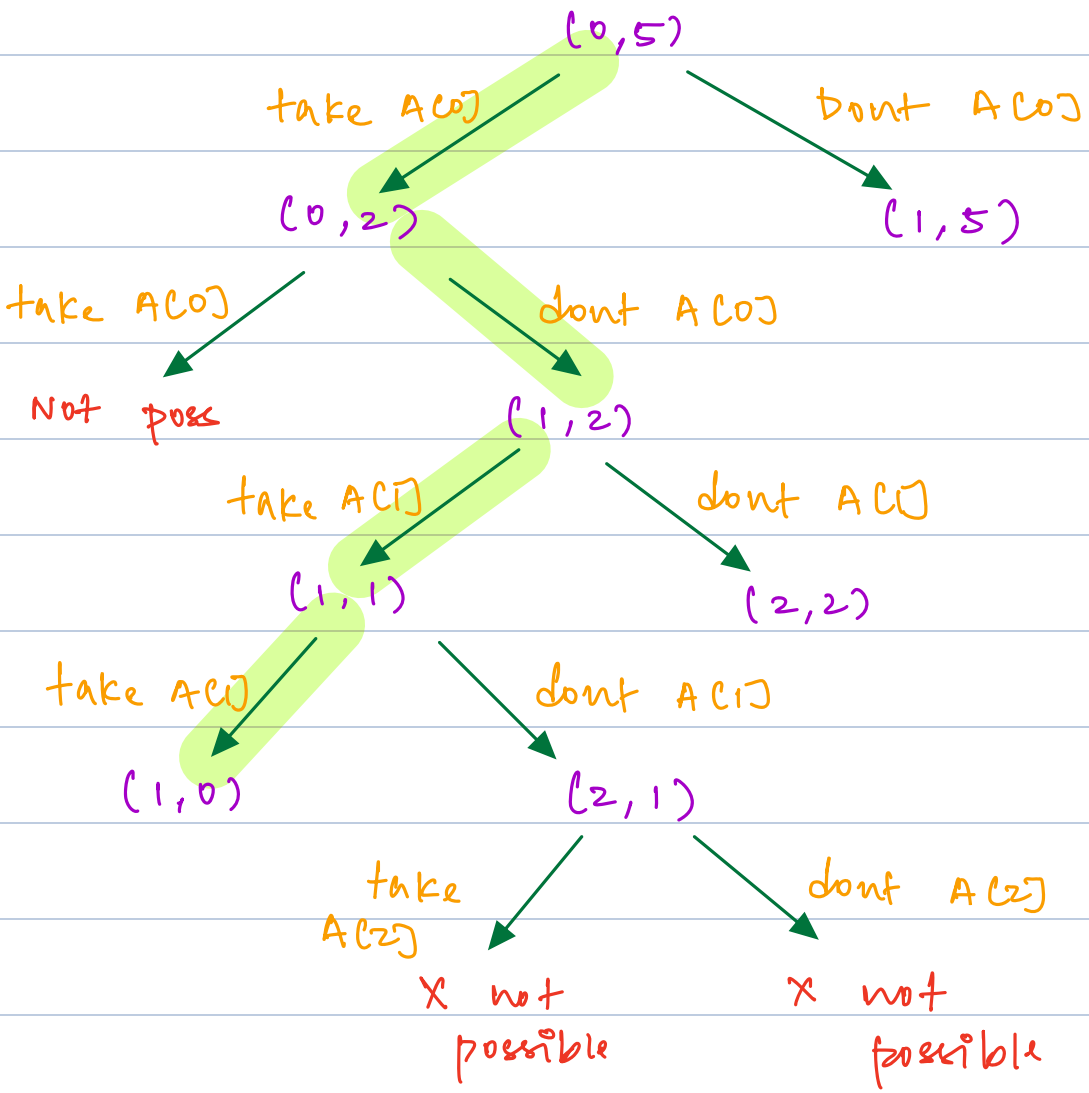
Observation

Unbounded Knapsack

DP state

(Index, total) ⟶ using coins from (0 - i)

how many ways we can make total ?



pseudo code

N-1 T

```
int solve ( index, target) {
    if (target == 0)  return 1;
    if (target < 0)   return 0; // target -ve
    if (index < 0)    return 0; // no more coins
    // DP state
    string key = index + "-" + target
reuse // if (hm.contains(key)) return hm[key];
    dont = solve (index-1, target)
    take = solve (index, target - A[index]);
    ways = dont + take;
    hm[key] = ways; // store
    return ways;
}
```

Break : 10:45 pm

Quiz 4

0-1 Knapsack

weight = | 3 | 6 | 5 | 2 | 4 |

value = | 12 | 20 | 15 | 6 | 10 |

0 1 2 3 4

Capacity = 8

MH = 12 + 15 = 27

0-1 Knapsack updated

Given N toys with their happiness and weight, find max total happiness that can be kept in a bag with capacity = C & toys cannot be divided ☺

Constraints

$1 <= n <= 500$

$1 <= H[i] <= 50$

$1 <= Wt[i] <= 10^9$

$1 <= C <= 10^9$

T.C = N * C

$= 500 * 10^9$

Does not work

Normal 0-1 Knapsack

DP state

DP[N][C] ⟶ max Happiness

Interchange

DP[N] [Max Happiness] ⟶ Min Capacity

sum[H[]]

min capacity needed from index 0 to N-1 to get max Happiness MH

weight =	3	6	5	2	4	Capacity = 8
happiness =	12	20	15	6	10	
	0	1	2	3	4	

pseudo code

N-1 sum[H[]]

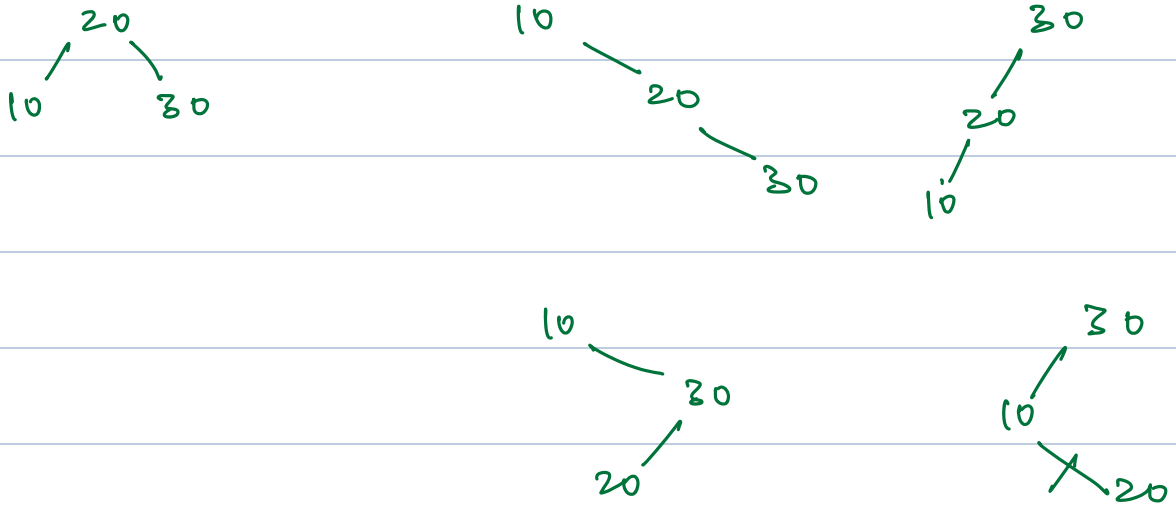
```
int solve ( index, MH) {
    if (MH == 0) return 0;
    if (index < 0) return INT_MAX;
    dont = solve ( index-1, MH);
    if (MH >= H[index])
        take = W[index] + solve (index-1,
                                 MH - H[index])
    else
        take = INT_MAX
    min_capacity = min (dont, take)
    return min_capacity;
}
```

H.W ⟶ Try memoization

Doubts Session

Count of unique BST

N = 3 [10, 20, 30]




```
C[0] = 1    C[1] = 1;
for (i = 2; i <= n; i++) {
    for (j = 0; j < i; j++) {
        C[i] += C[j] * C[n-1-j]
    }
}
```