Lecture ÷ DP-4

Agenda

- Rod cutting
- Coin sum infinite
  - Permutation
  - Combination
- 0-1 Knapsack -2

Qu1 **Rod cutting problem** (VVVI)

Given a rod of length n, and an array of length n.

A[i] ⟶ price of the ith rod

find max value we can obtain by selling rod.

Example: $n = 5$ [Rod length]

$$A[\,] = \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \\ \end{array}$$
$$A[\,] = [\,0 \quad 1 \quad 4 \quad 2 \quad 5 \quad 6\,]$$

- $2 + 3 \Rightarrow 4 + 2 = 6$
- $4 + 1 \Rightarrow 5 + 1 = 6$
- $2 + 2 + 1 \Rightarrow 4 + 4 + 1 = 9.$
- $3 + 1 + 1 \Rightarrow 2 + 2 + 1 = 4$
- $5 \Rightarrow 6$
- $2 + 1 + 1 + 1 \Rightarrow 4 + 1 + 1 + 1 = 7$

Similarity

Unbounded knapsack.

Capacity (K) = len of rod

wt[] — [0, 1, 2, 3, 4, 5]

val[] — arr[]

Max profit = Max value !!

Idea

dp[n+1]

↑ len of rod

dp[i] = Max profit we can get of rod = length i

Dry run:

n = 5

	1	2	3	4	5
A = [0	1	4	2	5	6]

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

dp[2]

[1]

1 1

1 + dp[1]

[2]

4 + dp[0]

max(2, 4) = 4

dp[3]

[1]

1 2

1 + dp[2]

(1 + 4 ,

[2]

4 + dp[1]

4 + 1 ,

[3]

2 + dp[0]

2 + 0)

dp[5]

[1] 1 + dp[4] → 9

[2] 4 + dp[3] → 9

[3] 2 + dp[2] → 6

[4] 5 + dp[1] → 6

[5] 6 + dp[0] → 6

Code

```
int rodcutting( arr[] )
        n = arr.length;
        len = n-1;
        dp[ len +1];
        dp[0] = 0;
        for( i=1; i<= len; i++) {
                max = -∞;
                for( j=1; j<=i; j++) {
                        max = Math.max( max, arr[j] +
                                                    dp[i-j]);
                }
                dp[i] = max;
        }
    return dp[len];
}
```

TC: $O(n^2)$

SC: $O(n)$

Qu2 **Coin change permutation.** $(x, y) \neq (y, x)$ (VVVVI)

Example: $K = 5$

$A = [3, 1, 4]$

- [1, 4]
- [4, 1]
- (1, 1, 1, 1, 1)
- (1, 1, 3)
- (1, 3, 1)
- (3, 1, 1)

6 Ans

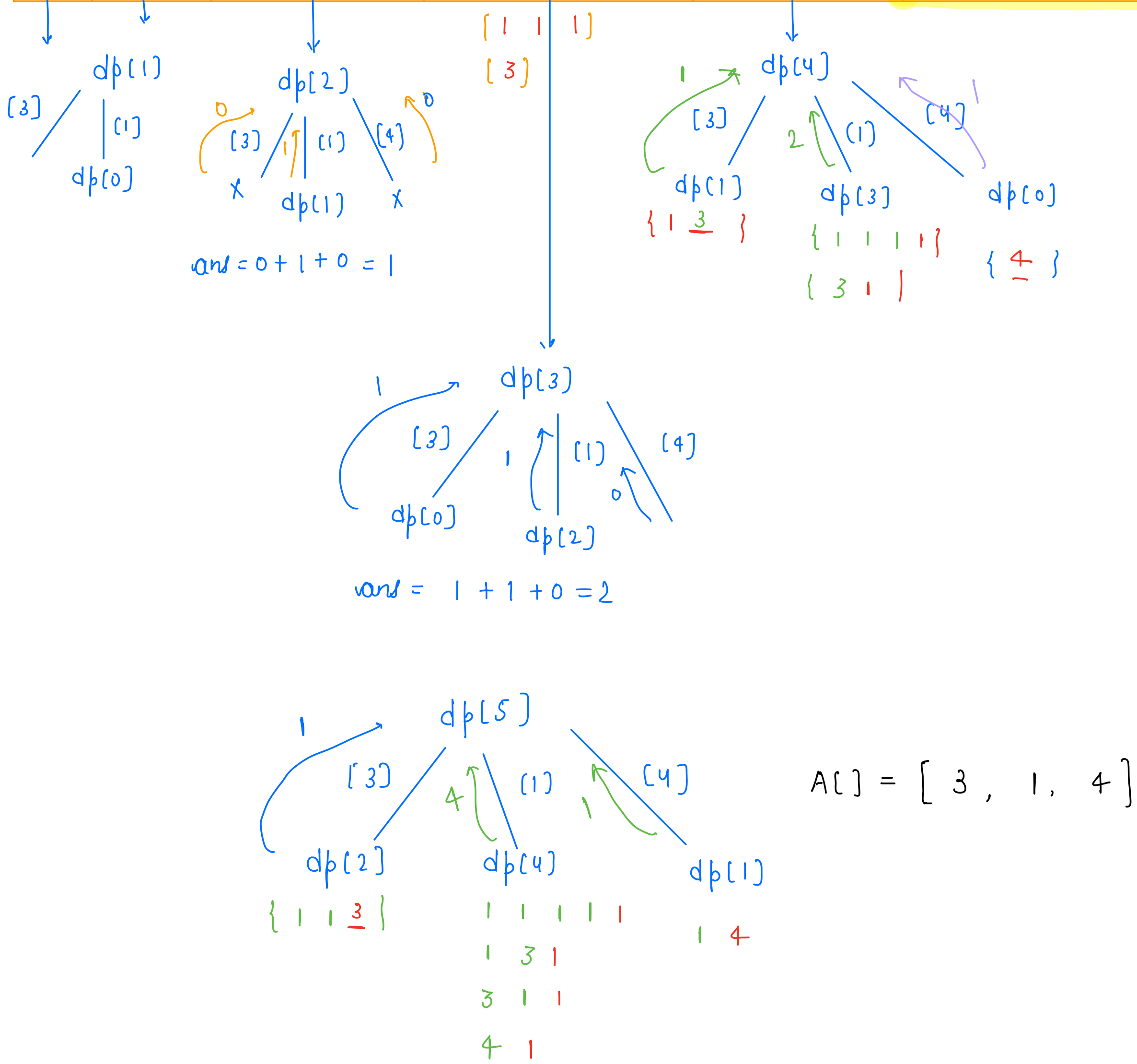
Idea: dp[k+1]

dp[i] = In how many permutation we can get ₹ i

Example: k = 5

A[] = [3, 1, 4]

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



Code:

```
int coinChangePermutation( arr[], k) {
        n = arr.length;
        dp[k+1];
        dp[0] = 1;

        for(i=0; i<=k; i++) {
            for(j=0; j<n; j++) {
                if( i-arr[j] >=0) {
                dp[i] = dp[i] + dp[i- arr[j]];
                }
            }
        }
    return dp[k];
}
```

TC: O(n*k)

SC: O(k)

Break: 8:33 AM

Qu: Coin change combination $[\,(x, y) = (y, x)\,]$

Example: $k = 5$

$$A = \begin{array}{ccc} 0 & 1 & 2 \\ \hline 3, & 1, & 4 \end{array}$$

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

3 ways

Idea:

$K = 7$

$A = [\,2,\ 3,\ 5\,]$

Iteration 1:

$K = 7$

$A = [\,2,\ 3,\ 5\,]$

0	1	2	3	4	5	6	7
1	0	1	0	1	0	1	0

dp[2] $\xrightarrow{[2]}$ dp[0]

dp[3] $\xrightarrow{[2]}$ dp[1]

dp[4] $\xrightarrow{[2]}$ dp[2]

Iteration 2

K = 7

A = [2, 3, 5]

0	1	2	3	4	5	6	7
1	0	1	0+1 1	1+0 1	0 + 1 1	1+1 2	0+1 1

dp[3]	dp[4]	dp[5]	dp[6]	dp[7]
[3]	[3]	[3]	[3]	[3]
dp[0]	dp[1]	dp[2	dp[3]	dp[4]

1

Iteration 3

K = 7

A = [2, 3, 5]

3 2 2

5 2

0	1	2	3	4	5	6	7
1	0	1	0+1	1+0	0+1 +1	1+1 +0	0+1 +1 2

dp[5] [5] dp[0]

dp[6] [5] dp[1]

dp[7] [5] dp[2]

Code

```
int coinChangeCombination (arr[], k) {
        n = arr.length;
        dp[k+1];
        dp[0] = 1;
        for(j=0; j<n; j++){
                for(i=0; i<=k; i++){
                        if( i-arr[j] >=0) {
                        dp[i] = dp[i] + dp[i- arr[j]];
                        }
                }
        }
    return dp[k];
}
```

TC: O(n*k)

SC: O(k)

Qu **0-1 knapsack 2**

Constraints:

$1 <= n <= 500$

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

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

$1 <= \underset{(K)}{capacity} <= 10^9$

Discussed algo

TC: $O(n * K)$

n → len(array)

K → capacity.

$n = 500$

$k = 10^9$

TC: $n * k = 500 * 10^9 > 10^8$

↓

TLE (Time Limit exceeded)

Discussed algo:

$dp[i][j]$ = Max value we can get in a bag of capacity j, such that we are choosing first i items.

$\downarrow$ $n+1$ $\quad$ $\downarrow$ $K+1$

New idea:

$dp[i][j]$ = Min weight required to get value j with first i items.

$\uparrow$ $n+1$ $\quad$ $\uparrow$ maxProfit + 1

H/W

$TC(n * maxProfit) \simeq 500 * 25000 = 25 * 5 * 10^5 = 125 * 10^5 < 10^8$

$1 <= n <= 500$

$1 <= val[i] <= 50$ $\longrightarrow$ Max profit = $500 * 50 = 25000$

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

$1 <= capacity <= 10^9$
(K)

0	1	2	- . . .			498	499
50	50	50	50			50	50

Thankyou ☺

Doubts

sum of array = s

$\sum$ flips < $\frac{s}{2}$ {sum = -ve}

$\boxed{|\text{capacity}| \longrightarrow \frac{s}{2}}$

$[\ ]^{n+1}$ $[\ ]^{\frac{s}{2}+1}$

___ ___ ___ ___

flip = ___ + dp ___

not = dp ___