Lecture :- Applications of Knapsack

Agenda

- Rod cutting
- Coin change permutation
- Coin change combination
- 0-1 Knapsack 2.

Qu Given a rod of length = n and A[n]   VVI

A[i] = price of ith rod

find max value obtained by selling rod.

n = 5 (Rod length)

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

→ 2 + 3 ⇒ 4 + 2 = 6

→ 4 + 1 ⇒ 5 + 1 = 6

→ 2 + 2 + 1 ⇒ 4 + 4 + 1 = 9

→ 2 + 1 + 1 + 1 ⇒ 4 + 1 + 1 + 1 = 7

Similarity Unbounded knapsack

capacity (k) = len of rod

wt[] = [indices of given array]

val[] = arr[]

Max profit = Max value.

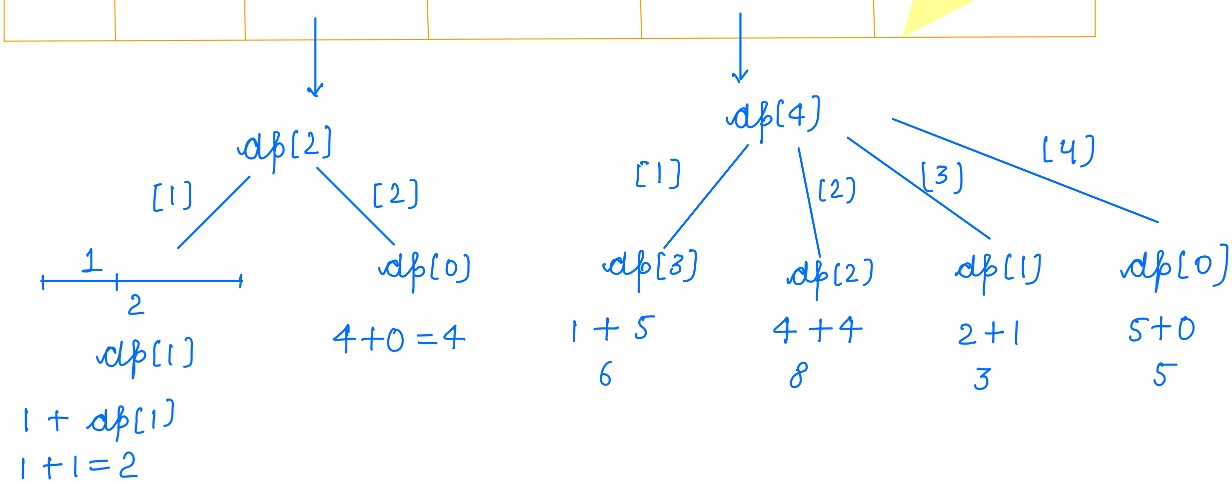
Idea dp[n+1]

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

Dry run n = 5 (Rod length)

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

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



Pseudocode

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

TC: $O(n^2)$

SC: $O(n)$

Qu coin change permutation $(x, y) \neq (y, x)$

$K = 5$

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] = no of permutations to get val = i

Dry run K = 5

3	1	4

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

dp[2]

$\xrightarrow{3}$ dp[-1] X 0

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

$\xrightarrow{4}$ dp[-2] X 0

0 + 1 + 0 = 1

dp[4]

$\xrightarrow{3}$ dp[1] → 1 { 1 3 }

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

$\xrightarrow{4}$ dp[0] = 1 { 4 }

dp[5]

3	1	4

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

$\xrightarrow{1}$ dp[4] 4 { 1 1 1 1 1 } { 3 1 1 } { 1 3 1 } { 4 1 }

$\xrightarrow{4}$ dp[1] 1 { 1 4 }

Pseudocode

```
int coinChangePerm( A[], k) {
    n = arr.length;

    dp[k+1];

    dp[0] = 1;

    for(i=1; i<=k; i++) {
        sum=0;
        for(j=0; j<n; j++) {
            if (i - A[j]) >=0) {
            val = dp[i - A[j]]
            sum += val;
        } }
        dp[i] = sum;
    }
    return dp[k];
}
```

TC: O(n*k)
SC: O(k)

Break: 8:11 – 8:21

Qu coin change combination $(x, y) = (y, x)$

$K = 5$

3	1	4

(1, 4)

(3, 1, 1)

(1, 1, 1, 1, 1)

ans = 3

Idea

$K = 7$

2	3	5

Iteration1

$K = 7$

2	3	5

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

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

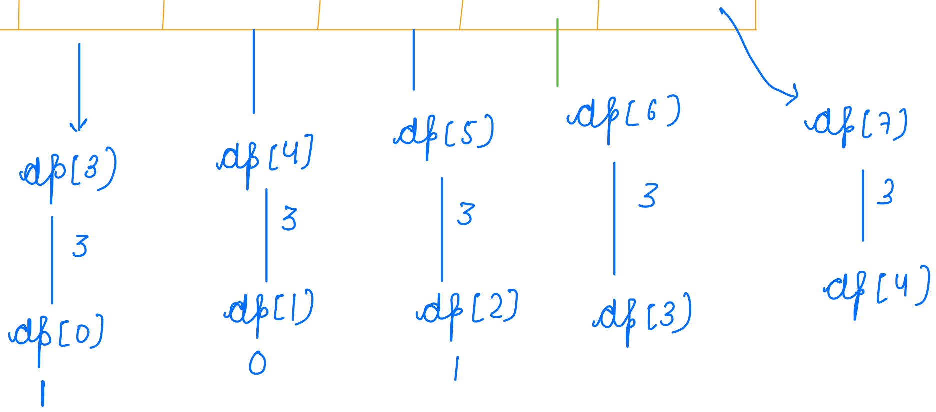
$dp[4] \xrightarrow{2} dp[2]$

Iteration2

K = 7

2	3	5

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



Iteration 3

K = 7

2	3	5

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

$dp[5] \xrightarrow{5} dp[0]$

$dp[6] \xrightarrow{5} dp[1]$

$dp[7] \xrightarrow{5} dp[2]$

Pseudocode

```
int coinChangeComb(A[], K) {
    n = arr.length;
    dp[K+1];
    dp[0] = 1;
    for(j=0; j<n; j++) {
        for(i=0; i<=K; i++) {
            if (i - A[j] >= 0) {
            dp[i] = dp[i] + dp[i - A[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

$T.C \Rightarrow O(n * K)$

len(arr) capacity

$n = 500$

$K = 10^9$

$n * K = 5 * 10^{11} > 10^8$

TLE

Discussed algo $dp[n+1][K+1]$

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

New idea $dp[n+1][maxProfit]$

$dp[i][j] \Rightarrow$ Min weight required to get profit j with first i items [Hint]

$1 <= n <= 500$

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

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

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

TC: $n * maxProfit$

$500 * 50 = 25*10^3$

500

$500 * 25*10^3$

$125 * 10^5 < 10^8$

Thankyou ☺